AIAA JOURNAL
Vol. 34, No. 4, April 1996

Adaptive Prismatic-Tetrahedral Grid Refinement and Redistribution for Viscous Flows

V. Parthasarathy* and Y. Kallinderis†
University of Texas at Austin, Austin, Texas 78712-1085

A hybrid grid adaptive algorithm that combines grid refinement and redistribution suitable for three-dimensional viscous flow simulations is presented. The flow domain is discretized with both prismatic and tetrahedral elements. The prismatic region comprises successive layers of semistructured prisms that encompass regions close to the wall where the viscous effects are dominant. The grid outside the prismatic region is tessellated with tetrahedra. A hybrid grid adaptation scheme that implements local refinement and redistribution strategies is developed to provide optimum meshes for viscous flow computations. Grid refinement on a hybrid grid is a dual adaptation scheme that couples isotropic division of tetrahedra and directional division of prisms. The directional division of prisms is essentially a two-dimensional grid refinement scheme that results in a significant reduction in required computing resources. The grid adaptive solver yields accurate results as compared with a globally refined grid with reduced computing resources.

I. Introduction

UNSTRUCTURED grids have been extensively used for three-dimensional inviscid flow modeling, and a wide range of Euler solvers have been developed and validated in the recent years. However, substantial progress is yet to be made in large-scale viscous computations on complex three-dimensional geometries. Using a fully unstructured grid results in a very large number of cells. Furthermore, it is difficult to generate thin tetrahedra in the regions close to the wall. The methodology proposed in Refs. 1–3 provides an alternative approach to generate suitable grids for Navier–Stokes computations by using prismatic elements. The prisms provide the option to use sufficient grid clustering in the normal direction as well as flexibility in geometric modeling by using unstructured tesselation in the lateral directions. Data structures used for prisms also reduce the storage requirements considerably. Prisms are used only in the regions close to the wall to capture the strongly directional viscous stresses. The rest of the domain can be very irregular, which makes the use of the more flexible tetrahedra beneficial. The tetrahedra are generated starting from the triangulation on the outermost surface of the prismatic region. The size and number of tetrahedral cells in the outer region are partly governed by the requirement that there be a gradual transition in grid sizes at the interface between prisms and tetrahedra.[3]

Unstructured grid adaptive algorithms that employ local refinement methods have gained a lot of momentum in recent years. This is because a grid refinement scheme in conjunction with a solver provides a great deal of flexibility and efficiency in obtaining numerical solutions. Some of the contributions made in this area in recent years are presented in Refs. 4–6. However, there have not been any such algorithms developed for hybrid grids since this area of research is still in its infancy. In the present work, a hybrid grid adaptation algorithm is developed that couples tetrahedral and prismatic grid refinement strategies. Prisms are refined directionally by altering only the unstructured tesselation in the lateral direction while retaining the structure in the normal direction. The directional adaptation of prisms does not enable the scheme to resolve the flow features that are aligned in the normal-to-the-surface direction. However, a grid redistribution algorithm is also employed to recluster the nodes in the normal direction so as to better resolve the viscous stresses. The tetrahedral cells are refined to resolve features that are dominant in the inviscid region of the flow domain. The tetrahedral division methods are similar to those developed in the dynamic adaptation algorithm of Ref. 5.

The Navier–Stokes algorithm developed in the present work is an extension of a node-centered, finite volume Euler scheme described in Ref. 7. Two important issues that need to be addressed regarding viscous computations are odd-even mode suppression and storage requirements. Depending on whether the scheme is node centered or cell centered, the choice of the right location at which to evaluate the derivative terms is crucial to eliminate odd-even modes. In the present work, the semistructure of prismatic grids is exploited to reduce the required memory storage.

Unstructured grids typically lay substantial demands on computer storage requirement because of indirect addressing and arbitrariness of nodal distribution that may entail the need for extensive connectivity information. Hence, it is imperative that a numerical scheme developed be storage efficient. Typical viscous flow simulations on a fully unstructured grid would require cells of the order of 10^6. Most existing unstructured Navier–Stokes methods need nothing short of a supercomputer for modeling viscous flows on complex three-dimensional geometries. One of the prime objectives of the present work has been to implement a scheme that could fit viscous simulations in a workstation. A storage-efficient data structure is employed for both the solver and adapter, utilizing the semistructure of prismatic grid, and hence the memory overheads are reduced considerably.

II. Finite Volume Discretization

The Navier–Stokes equations for viscous fluid flow are written in the differential form as follows:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{\mathcal{F}} = \nabla \cdot \boldsymbol{\mathcal{R}} \tag{1}$$

where $\boldsymbol{U}$ is the state vector; $\boldsymbol{\mathcal{F}}$ comprises the convective flux vector components $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$; and $\boldsymbol{\mathcal{R}}$ comprises the viscous flux vector components $\boldsymbol{R}$, $\boldsymbol{S}$, and $\boldsymbol{T}$. The state vector and the convective and viscous flux vectors are defined in terms of primitive variables.[7]

The solution at any node N, at time level $n+1$ can be expressed in terms of the solution at time level n using a Taylor series expansion:

$$\boldsymbol{U}_N^{n+1} = \boldsymbol{U}_N^n + \delta \boldsymbol{U}_N^n$$

$$\delta \boldsymbol{U}_N^n = \boldsymbol{U}_N^{n+1} - \boldsymbol{U}_N^n = \left(\frac{\partial \boldsymbol{U}}{\partial t}\right)_N^n \Delta t + \left(\frac{\partial^2 \boldsymbol{U}}{\partial t^2}\right)_N^n \frac{\Delta t^2}{2} + \mathcal{O}(\Delta t^3) \tag{2}$$

Received March 25, 1995; revision received Sept. 28, 1995; accepted for publication Oct. 3, 1995.

*Postdoctoral Research Associate, Department of Aerospace Engineering and Engineering Mechanics; currently Research Engineer, CFD Research Corp., Huntsville, AL 35805. Member AIAA.

†Associate Professor, Department of Aerospace Engineering and Engineering Mechanics. Senior Member AIAA.

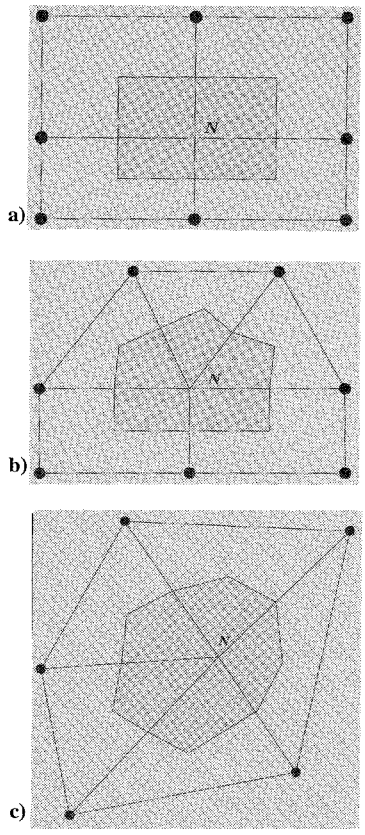


Fig. 1 Dual volume construction for mixed-element topology. Two-dimensional analogies for dual mesh around a node in the a) prismatic region, b) tetrahedral-prismatic interface, and c) tetrahedral region.

The temporal derivatives in the preceding expression are evaluated in terms of spatial derivatives using the governing equations according to the Lax–Wendroff approach. The finite-volume method evaluates the integral averages of the temporal derivative terms in Eq. (2) over the control volume Ω_N associated with node N.[7]

A. Spatial Discretization with Mixed Elements

The spatial discretization proceeds by constructing around each node N a dual cell that represents the control volume over which the integral averages of the temporal derivatives are evaluated. The two-dimensional analogy of defining dual cells for different configurations in a triangular-quadrilateral hybrid mesh is illustrated in Fig. 1. The duals are defined by connecting the midpoints of the edges and centroids of the triangular and/or quadrilateral faces that share the node. Dual cells for a three-dimensional hybrid grid are constructed along similar lines using the centroids of faces and cells with which each node is associated.

The integral average of the first-order temporal derivatives associated with the node N is written in discrete form following the governing equation (1):

$$\left(\frac{\partial U}{\partial t}\right)_N = \frac{-1}{\Omega_N}\sum_f (\mathcal{F}-\mathcal{R})_f \cdot \hat{n}_f S_f \tag{3}$$

where the summation f is over all of the discrete faces of the dual mesh that constitute $\partial\Omega_N$. It is shown in Ref. 7 that the summation in Eq. (3) can be alternatively computed on an edgewise basis as

$$\left(\frac{\partial U}{\partial t}\right)_N = \frac{-1}{\Omega_N}\sum_e (\mathcal{F}-\mathcal{R})_e \cdot \hat{n}_e S_e \tag{4}$$

where the summation e is over all of the edges that share the node N. The term S_e represents the dual-face area associated with each edge, and $\hat{n}_e$ is the unit normal vector of the dual-face area S_e. The areas S_e are computed using the dual mesh construction of Fig. 1, by accumulating the areas of each dual-mesh face that shares the edge e. The finite volume scheme then proceeds by computing $\delta U s$ at the nodes by a global sweep over the edges and is thus transparent to whether a node lies in the tetrahedral region, in the prismatic region, or at the interface.

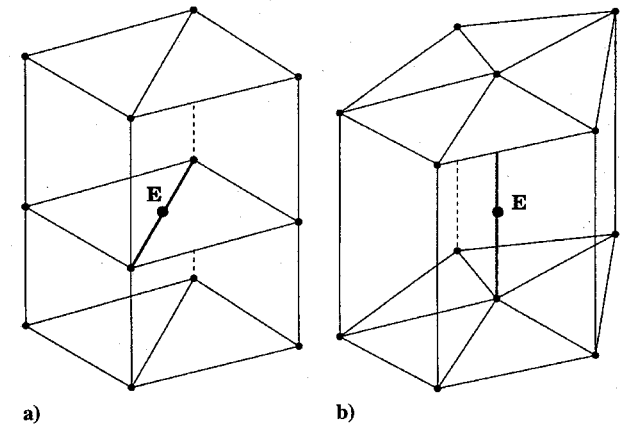


Fig. 2 Edge-dual volumes defined around the prismatic edges for computing the gradients of primitive variables at the edge centers: a) lateral edge and b) vertical edge.

The second-order temporal derivatives are evaluated along similar lines. The expression for the second-order derivative at node N is given from Ref. 7 as follows:

$$\left(\frac{\partial^2 U}{\partial t^2}\right)_N = \frac{-1}{\Omega_N}\int_{\partial\Omega_N} (\tilde{A}n_x + \tilde{B}n_y + \tilde{C}n_z)\frac{\partial U}{\partial t}\,\mathrm{d}S \tag{5}$$

where $\tilde{A} = \partial E/\partial U$, $\tilde{B} = \partial G/\partial U$, and $\tilde{C} = \partial H/\partial U$ are the Jacobians of convective flux vectors. The Jacobians of flux vectors need to be computed to evaluate the second-order derivatives. However, only the convective flux vectors are considered in this step as the Jacobians of viscous flux vectors are too expensive to compute. Therefore, discretization of the viscous terms is first-order accurate in time and second-order accurate in space.

B. Viscous Stresses Calculation Using Edge-Dual Volumes

The viscous flux vectors $\boldsymbol{R}$, $\boldsymbol{S}$, and $\boldsymbol{T}$ comprise viscous stress and heat flux terms. Hence, the derivatives of the primitive variables u, v, w, and E need to be evaluated. Different approaches may be employed to compute the derivatives depending on whether a scheme is node centered or cell centered. In the present work, the scheme being node centered and edge based, the alternatives are either to compute the derivatives directly at the edges or to compute them at the nodes and average them at each midedge point. This choice is crucial in regard to suppressing odd-even modes that frequently appear in numerical solutions. It is well substantiated that the latter strategy is not capable of eliminating the odd-even modes.[8,9] Hence, derivatives are evaluated directly at the edges. The gradient of a scalar ϕ is computed using Green's theorem as follows:

$$(\nabla\phi)_E = \int_{\Omega_E} (\nabla\phi)\,\mathrm{d}\Omega = \int_{\partial\Omega_E} \phi\hat{n}\,\mathrm{d}S \tag{6}$$

To evaluate the gradients, dual volumes are defined around each edge. The dual volume for edge E required to calculate the preceding integral is illustrated in Figs. 2a and 2b for lateral and vertical prismatic edges, respectively. The edge dual is defined by the union of all of its neighboring prisms. The gradients of u, v, w, and E are evaluated by looping once through the quadrilateral and triangular faces that constitute the prismatic grid and computing the flux balances an in Eq. (6).

III. Hybrid Grid Adaptation

A dynamic grid adaptation algorithm has previously been developed for three-dimensional unstructured grids.[5] The algorithm is capable of simultaneously unrefining and refining appropriate regions of the flow domain. This method is extended to the present

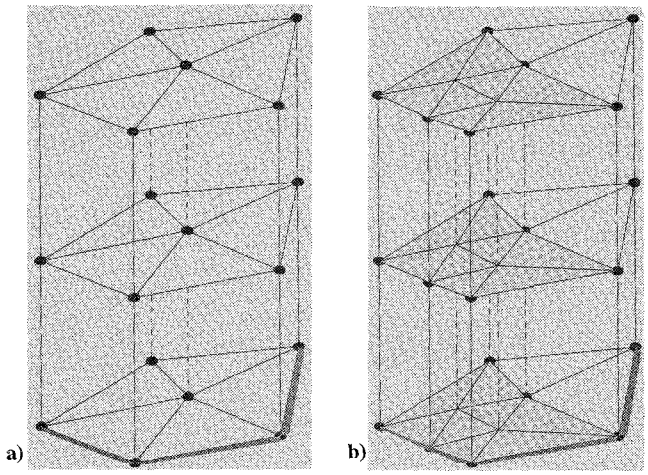


Fig. 3 Directional refinement of prisms: a) initial unembedded grid and b) embedded grid obtained after quadtree and binary divisions of the triangular faces.

work and is coupled with prismatic grid adaptation to implement a hybrid adaptation method. The adaptive algorithm is guided by a feature detector that senses the regions with significant changes in flow properties. Velocity differences and velocity gradients are used as detection parameters. The details of the feature detector used in this work are given in Ref. 10.

A. Directional Division of Prisms

The prisms are refined directionally to preserve the structure of the mesh along the normal-to-surface direction. The prismatic grid refinement proceeds by dividing only the lateral edges that lie on the wall surface and hence the wall faces. The faces on the wall are divided into two or four subfaces. The resulting surface triangulation is replicated in each successive layer of the prismatic grid. This results in all of the prisms that belong to the same stack (namely, the group of cells that originate from the same triangular face on the wall surface) getting divided alike as illustrated in Fig. 3. The prismatic grid refinement preserves the structure of the initial grid in the direction normal to the surface. This is important if the adapted grid is also to retain the two dimensionality of the data structure. Another advantage of using such an adaptive algorithm for prisms is that the data structures needed for its implementation are essentially as simple as that for refining a two-dimensional triangular grid.

The directional division of the prisms does not increase resolution in the direction that is normal to the wall surface. Therefore, a grid redistribution algorithm can be employed to recluster nodes in the normal direction so as to better resolve the viscous stresses.[2,10]

B. Tetrahedral Grid Refinement

Tetrahedra are divided into two, four, or eight subcells as shown in Fig. 4. As the feature detector flags only the edges for division, there will arise situations where the flagged edges of a tetrahedron may not straight away allow one of these three types of division shown in Fig. 4. To circumvent this problem and to simplify the refinement algorithm, such cases are constrained by flagging additional edges in the tetrahedron so that the cell conforms to the refinement types delineated in Fig. 4.

The tetrahedral cells created by 1:8 octree division have an aspect ratio that is comparable to that of the parent cell whereas 1:2 binary and 1:4 quadtree divisions result in skewed cells. Hence, to avoid excessive grid skewness, repeated binary and quadtree divisions of tetrahedra that originated from such a refinement in a previous pass are avoided. Furthermore, to avoid sudden changes in grid size, the grid refinement algorithm also limits the maximum difference in embedding level between neighboring cells to two. All of these aforementioned refinement-constraining rules are enforced in an equivalent two-dimensional sense to the prismatic adaptation scheme as well.

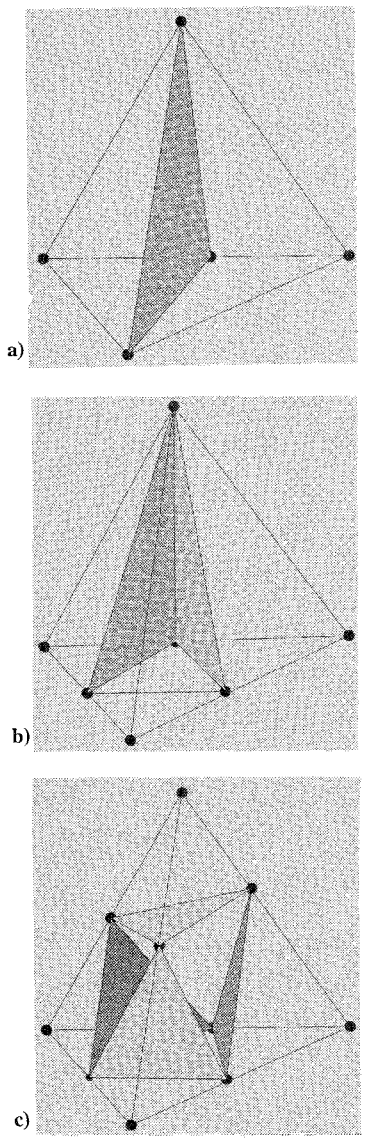


Fig. 4 Refinement strategies for a tetrahedron: a) binary (1:2), b) quadtree (1:4), and c) octree (1:8) divisions.

C. Coupling of Prismatic-Tetrahedral Adaptation

The hybrid adaptation is completed by splicing the refined prismatic and tetrahedral grids together at the interface. The feature detectors that flag the edges in the prismatic and tetrahedral regions function independently. This may cause the following discrepancies at the interface between prisms and tetrahedra: 1) an edge on the wall surface may be flagged for refinement, but its counterpart in the tetrahedral region may not have been flagged by the tetrahedral feature detector; and 2) a tetrahedral edge may be flagged for refinement, but its footprint on the wall may not have been flagged by the prismatic feature detector.

This is illustrated in Fig. 5, which shows a portion of a hybrid grid at the interface. The tetrahedral and prismatic regions are shown staggered. Referring to the figure, nodes $t1$ and $t2$ are to be added at the midpoints of the boundary edges that correspond to the interface edges where tetrahedral nodes $T1$ and $T2$ appear. Likewise, nodes $p1$ and $p2$ need to be introduced at the interface edges that correspond to the boundary edges where prismatic nodes $P1$ and $P2$ appear. Likewise, these discrepancies are eliminated before refining the prisms and tetrahedra so that the adapted grids in the two regions can be coupled together to create a hybrid grid without any hanging nodes on the interface.

D. Redistribution of Prisms

The redistribution algorithm increases local grid resolution by clustering existing grid points in regions of interest. Since the

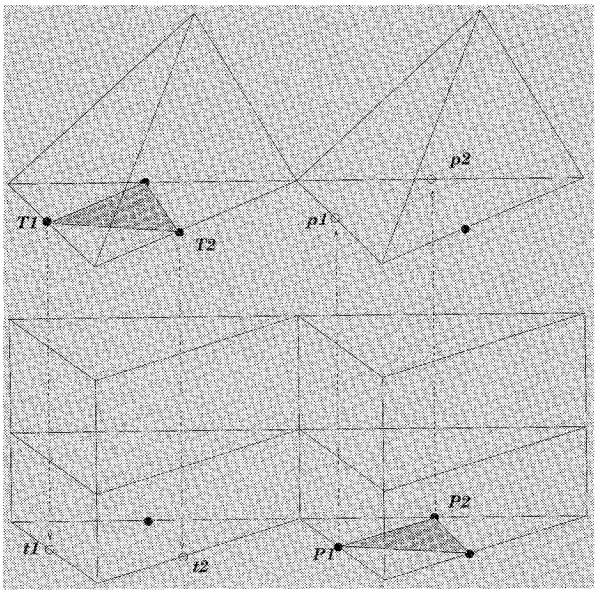


Fig. 5 Coupling of prismatic-tetrahedral adaptation at the interface: •,—midedge nodes introduced by the feature detector and ○,—midedge nodes to be introduced to eliminate hanging nodes at the interface.

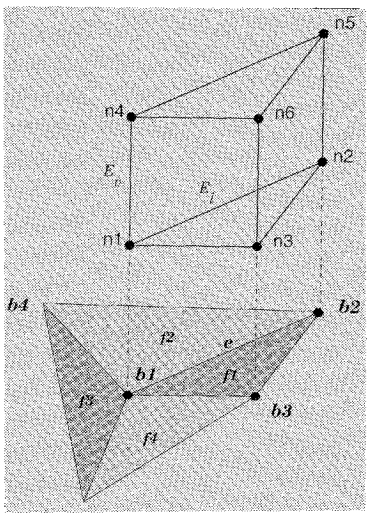


Fig. 6 Two-dimensional pointers of surface triangulation define a prismatic grid. Nodes $b1, b2, b3$, and $b4$ are on the boundary, and nodes $n1, n2$, and $n3$ are on layer j and $n4, n5$, and $n6$ are on layer $(j+1)$.

number of grid points is fixed, reclustering in one region may result in less resolution elsewhere when the initial grid does not include a sufficient number of points. Nevertheless, redistribution can be advantageous when the number of nodes is sufficient. Even in cases where the initial grid has adequate resolution for viscous stresses, the nodes may be misaligned with the emerging flow features and some realignment freedom is beneficial. The redistribution scheme complements directional refinement of prisms as the latter does not alter the grid resolution along the normal directions.

A measure of the grid resolution required normal to the no-slip wall could be the values of $y^+ = u_\tau y/\nu$, with $u_\tau = \sqrt{(|\tau_{\text{wall}}|/\rho_{\text{wall}})}$ being the wall friction velocity.[10] A criterion based on the values of y^+ at the wall is employed to either attract nodes towards the wall or repel them away from the surface so that a specific value of y^+ is attained at all of the wall nodes. This procedure in essence determines a new value for the spacing δ_{wall} of the first node off the wall at all locations on the wall surface. The nodes in the prismatic region are then reclustered along the marching lines emanating from the corresponding wall node, in accordance with the new value of δ_{wall}. Specifically, the redistribution algorithm is as follows:

1) Integrate on the initial mesh for a certain number of iterations.

2) Calculate the values of y^+ at the wall nodes. Then evaluate both the average and standard deviation of the y^+ distribution. The specified value of y^+ is obtained as

$$y^+_{\text{specified}} = y^+_{\text{ave}} - \alpha y^+_{\text{sdev}}$$

A typical value of α is 0.4.

3) Move the nodes so that $y^+_{\text{new}} = y^+_{\text{specified}}$. Apply a smoothing procedure to ensure that the values of y^+_{new} do not change too drastically between neighboring wall nodes. Finally, rescale the distances by which nodes are to be moved along the marching lines so that the maximum distance is less than a certain value. This is useful in regions like stagnation and separation points wherein a limited number of nodes may move by relatively large distances compared with the rest of the nodes.

IV. Memory Requirements

The scope of this section is to 1) present details as to how a set of two-dimensional pointers is used to define a prismatic grid and 2) to compute the reduced storage requirements for a hybrid grid as opposed to using a fully unstructured grid.

A. Two-Dimensional Prismatic Grid Pointers

Consider a triangulation on the wall boundary surface that comprises NBnodes nodes, NBedges edges and NBfaces faces. A prismatic grid comprising Nlayers layers is generated with this triangulation.

The data structures employed to define the surface triangulation include face-to-node, face-to-edge, edge-to-node, and edge-to-face pointers. These pointers require very little memory storage. An illustration of using two-dimensional pointers to define a prismatic grid is presented in Fig. 6.

Referring to the figure, the addresses of nodes defining the prism are obtained using the nodes of their footprint on the boundary surface (i.e., the nodes $b1$, $b2$, and $b3$ of face $f1$), along with the layer index j, as follows:

$$n1 = b1 + (j-1) * NB\text{nodes} \qquad n4 = b1 + j * NB\text{nodes}$$

$$n2 = b2 + (j-1) * NB\text{nodes} \qquad n5 = b2 + j * NB\text{nodes}$$

$$n3 = b3 + (j-1) * NB\text{nodes} \qquad n6 = b3 + j * NB\text{nodes}$$

The four neighboring prisms (c_i) that define the control volume around the lateral prismatic edge E_l are obtained using the two faces $(f1, f2)$ that share the footprint of edge E_l on the wall, namely, edge e:

$$c1 = f1 + (j-1) * NB\text{faces} \qquad c3 = f1 + (j-2) * NB\text{faces}$$

$$c2 = f2 + (j-1) * NB\text{faces} \qquad c4 = f2 + (j-2) * NB\text{faces}$$

The neighboring prisms that define the control volume around a vertical prismatic edge E_v are obtained using the faces $(f1, f2, f3, f4)$ that share the wall boundary node $b1$:

$$c1 = f1 + (j-1) * NB\text{faces} \qquad c3 = f3 + (j-1) * NB\text{faces}$$

$$c2 = f2 + (j-1) * NB\text{faces} \qquad c4 = f4 + (j-1) * NB\text{faces}$$

The semistructured nature of prisms can be used in conjunction with the two-dimensional pointers to define all sorts of grid connectivities in the prismatic region. If a fully unstructured grid is used in the viscous region, such pointers would impose very high storage requirements. The use of a prismatic grid thus greatly alleviates the demand on memory.

B. Memory Storage for Solver and Adapter

The number of cells, faces, edges, and nodes in a tetrahedral grid are related by the following formula:

$$N\text{faces}/2 \simeq NT\text{cells} \simeq NT\text{edges} \simeq 5 \times NT\text{nodes}$$

whereas for the boundary mesh the formula is

$$\tfrac{2}{3} \times NB\text{edges} \simeq NB\text{faces} \simeq 2 \times NB\text{nodes}$$

Further, using the surface triangulation parameters and the number of prismatic layers, the number of cells, faces, edges, and nodes in the prismatic region (NPcells, NPfaces, NPedges, and NPnodes) can be expressed as follows:

$$NP\text{cells} = NB\text{faces} \times N\text{layers}$$

$$NP\text{faces} = NB\text{faces} \times (N\text{layers} + 1) + NB\text{edges} \times N\text{layers}$$

$$NP\text{edges} = NB\text{edges} \times (N\text{layers} + 1) + NB\text{nodes} \times N\text{layers}$$

$$NP\text{nodes} = NB\text{nodes} \times (N\text{layers} + 1)$$

Edge arrays for the solver comprise edge-to-node pointers and dual-area projections associated with each edge. Evaluation of derivatives at the prismatic edges require additional storage of terms such as the area projections of all of the triangular and quadrilateral faces in the prismatic region as well as storage of edge-dual volumes and edge-centered gradients of the primitive variables u, v, w, and E. The node arrays include storage of nodal state vectors, coordinates, volumes, and node-based time steps. Comprehensively, using the relations between different parameters as expressed previously, the memory requirement of the tetrahedral region (M_T) and of the prismatic region (M_P) of the hybrid solver is

$$M_T = 50 \times NT\text{nodes}$$

$$M_P = 100 \times NB\text{nodes} \times NT\text{layers} \simeq 100 \times NP\text{nodes}$$

The data structures employed for prismatic adaptation are again a set of two-dimensional pointers that comprise face-to-node, face-to-edge, edge-to-node, and edge-to-face connectivities for the boundary surface triangulation. Arrays for the tetrahedral grid include cell-to-node, edge-to-node pointers that would suffice to perform grid refinement. Additional storage is required for the wall boundary faces and the tetrahedra to store the parent-child pointers that constitute the embedded-tree structure. Comprehensively, the memory requirements (M_T, M_P) for the adapter are given as

$$M_T = 70 \times NT\text{nodes}$$

$$M_P = 25 \times NB\text{nodes}$$

Therefore, the total memory storage for the hybrid adapter is approximately $70 \times NT$nodes.

For a hybrid grid comprising 5000 wall boundary nodes, 20 layers of prisms, and 250,000 tetrahedra, the preceding expressions are used to compute the actual computer memory required. This is approximately 12 Mwords for the solver and 3 Mwords for the adapter.

V. Results

A number of test cases are employed to provide an assessment of accuracy and robustness of the developed adaptive Navier–Stokes method with hybrid meshes. The test cases include flow over a cylindrical bump in a channel, as well as flow around a semispherical body. The cases considered involve flows with both shear layers and shock waves present. Accuracy of the numerical results obtained with the adaptive hybrid solver is validated by comparison with two-dimensional numerical results as well as by comparison of results obtained with different types of grid. Convergence of the time-marching process is addressed in Refs. 11 and 12.

A. Flow Past a Cylindrical Bump in a Channel—Hybrid Grid vs Prismatic Grid

Laminar flow past a circular bump in a two-dimensional channel has been used extensively as a benchmark for validating two-dimensional Navier–Stokes schemes. The flow past a cylindrical bump in a three-dimensional channel is quasi-two-dimensional, and hence the results can be compared directly with the two-dimensional numerical results. The nondimensional flow parameters are freestream Mach number $M_\infty = 1.4$ and Reynolds number $Re = 1.6 \times 10^4$. Characteristic boundary conditions are imposed on the inlet and outlet planes of the channel. No-slip boundary conditions are imposed on the lower wall surface of the channel downstream of the leading edge of the bump. Inviscid wall boundary conditions are imposed on all of the rest of the boundary surfaces.

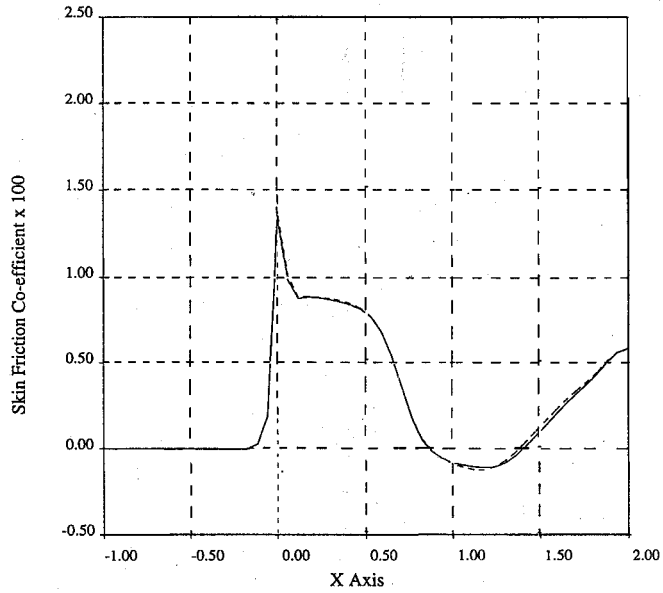


Fig. 7 Comparison of skin-friction coefficients obtained on the hybrid grid and the all-prismatic grid for supersonic, laminar flow at $M_\infty = 1.4$ and $Re = 1.6 \times 10^4$ through a channel with a 4% bump: ——, solution on the hybrid grid and – – –, solution on the all-prismatic grid.

Numerical solution is computed using two types of grid, viz., an all-prismatic grid and a hybrid grid. The Blasius solution for flow over a flat plate relates the thickness of boundary layer at any location x to the Reynolds number as $\delta(x) \propto 1/\sqrt{Re}$. Using this as an approximate guiding rule, a grid spacing of $\delta_w \sim 10^{-3}$ is chosen at the wall a priori for the prismatic grid. A stretching factor of 1.1 is used to cluster the prisms near the wall. The prismatic grid comprises ~500 wall boundary nodes and 64 layers of prisms. The hybrid grid is generated by dividing each prism in the outer 32 layers into three tetrahedra. This ensures that the nodes are collocated in both grids, which permits a direct comparison between them. The tetrahedral region of the hybrid grid comprises ~85,000 cells. A comparison of the values of skin-friction coefficient $C_f = 2\tau_w/\rho_\infty U_\infty^2$ between solutions computed on the two grids is shown in Fig. 7. It is observed that the values are in good agreement. The comparison shows that a hybrid grid can yield solutions of the same accuracy as with an all-prismatic grid.

The first hybrid grid chosen has a wall surface triangulation that has ~500 nodes and comprises 25 layers of prisms. The tetrahedral region comprises ~4000 nodes and ~15,000 tetrahedra. The tetrahedral region of the hybrid grid is fully unstructured unlike the previous case where it was generated by dividing the prisms. The tetrahedral grid is generated by tessellating the region using the advancing front method. The second hybrid mesh chosen has the same wall surface triangulation but comprises 32 layers of prisms. The tetrahedral region comprises ~2500 nodes and ~10,000 tetrahedra.

The grids are adaptively embedded in two successive passes of adaptation based on velocity differences and velocity gradients as the detection parameters. The twice-embedded mesh resulting from the first hybrid grid is shown in Fig. 8. The figure shows the isometric view of the tessellation on the boundary surfaces of the adapted grid. Local embedding is observed in a predominant portion of the region downstream of the bump after the first pass of adaptation. The figure clearly shows the dual nature of the hybrid grid adaptation scheme involving isotropic refinement of the tetrahedra with two-dimensional refinement of the surface triangulation that is directionally extended all of the way till the prism–tetrahedra

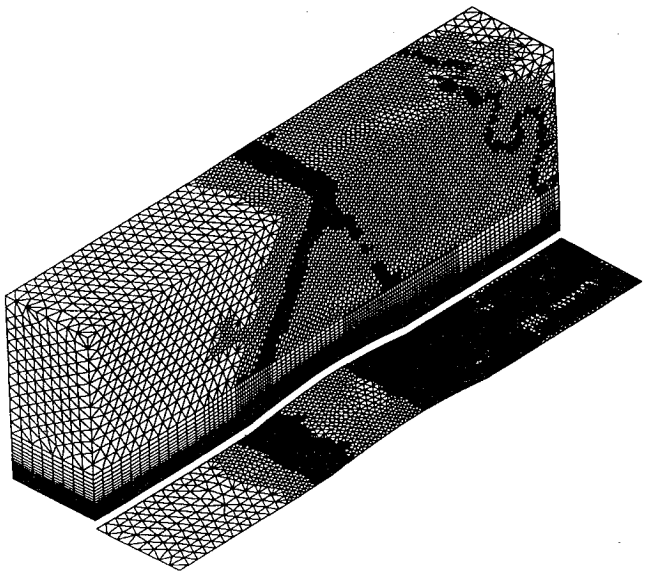
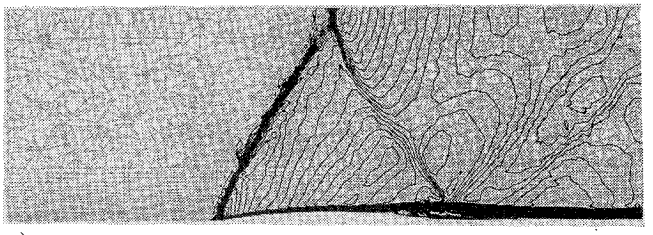
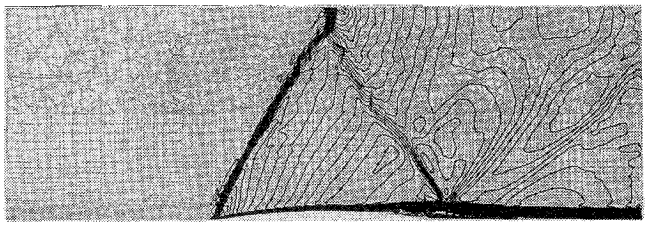


Fig. 8 Isometric view of the twice-embedded grid with 25 layers of prisms. Tessellations on the boundary surfaces are shown. The lower viscous wall boundary surface is shown staggered for clarity. The initial unembedded grid comprises a semistructured prismatic region and a tetrahedral region which is fully unstructured and tessellated with the advancing front method.



a)



b)

Fig. 9 Mach number contour lines superimposed on the mesh at a section cutting through the interior of the hybrid grid at midspan. Solution was adaptively obtained on embedded grids for supersonic, laminar flow at $M_\infty = 1.4$ and $Re = 1.6 \times 10^4$ through a channel with a 4% bump ($M_{\min} = 0$, $M_{\max} = 1.6$, and $\Delta M = 0.05$): a) hybrid grid with 25 layers of prisms and b) hybrid grid with 32 layers of prisms.

interface. The embedding in the second pass of adaptation is seen to be focused in the shock regions. The twice-embedded mesh shown in Fig. 8 comprises 4165 wall boundary nodes and ~171,000 tetrahedra.

B. Adaptive Hybrid Grids

Two adaptive hybrid grids with different prism–tetrahedra interface locations are employed. The prism–tetrahedra interface is placed at different locations in the two grids to study the influence of the grid interface on the numerical solution obtained using the hybrid approach. Solutions are obtained adaptively on grids using local grid refinement, and the computed results are compared with the corresponding two-dimensional numerical solution.

Mach number contour lines of the final solution computed on each embedded grid are shown in Figs. 9a and 9b, which show a section passing through the interior of the unstructured grid superimposed

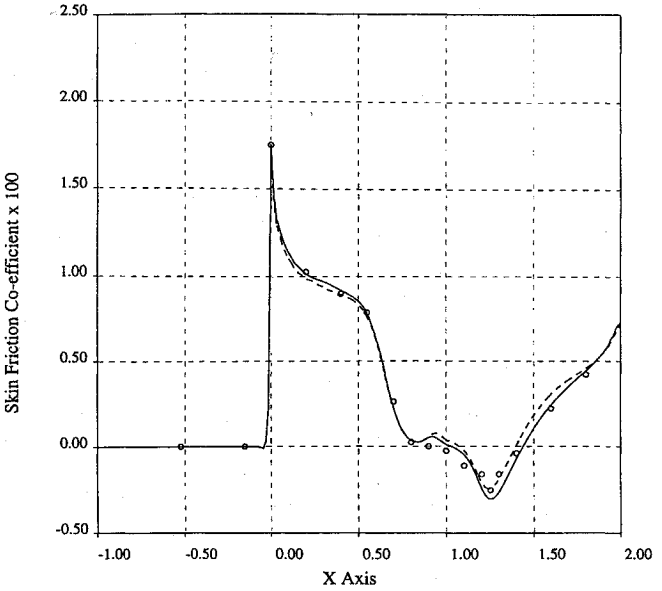


Fig. 10 Comparison of skin-friction coefficients computed at the midspan of the channel along the lower viscous wall: ——, solution on the twice-adapted 25 layered hybrid grid; – – –, solution on the twice-adapted 32 layered hybrid grid; and o o o, two-dimensional numerical result.[15]

on the solution. Redistribution of prisms was not employed for this case. Qualitative assessment of the computed solutions in each case shows that switching to a different topology at the interface does not affect the solution adversely. Further, it is also seen that the solver is robust enough to deal with the drastic changes in topology at the interface, as is seen from Figs. 9a and 9b. Comparison of the values of skin-friction coefficients computed for the respective cases with that of the two-dimensional numerical results is shown in Fig. 10. The computed values are seen to be in good agreement with two-dimensional results shown in the figure, which substantiates the validity of the adaptive, hybrid grid approach for viscous computations. Furthermore, it is observed that placing the prism–tetrahedra interface at different locations did not cause any disparity in the solution, and the results are in good agreement with each other.

C. Flow Past a Sphere

An external flow problem that involves supersonic flow past a semisphere at a freestream Mach number of 1.4 and a Reynolds number of 1×10^3 (based on the radius) is examined. A symmetry plane is used. The far-field boundary is placed approximately at a distance of 50 radii of the sphere.

The flow is characterized by both inviscid and viscous flow features such as shock waves and boundary-layer separation. The Navier–Stokes solver combined with hybrid grid refinement and grid redistribution is employed to show the efficacy of the grid adaptation methods in adaptively obtaining accurate solutions. Numerical solutions are computed for two different cases: 1) a hybrid grid adaptively refined using the dual adapter and 2) a redistributed hybrid grid. Comparisons are made with the numerical results computed using a globally fine grid. The values of skin-friction coefficients are computed at the junction of the spherical surface and an equatorial plane normal to the symmetry plane.

Dual Adaptation of Hybrid Grid

A hybrid grid comprising ~1400 wall boundary nodes and ~100,000 tetrahedra is used as the initial grid. The prismatic region is constituted by 20 layers of prisms that are clustered with a grid spacing of $\delta_w = 10^{-2}$ at the wall and a stretching factor of 1.1. The adapted grid obtained after local refinement based on an initial solution is shown in Fig. 11. The figure shows the tessellation on the wall surface, on the symmetry plane, as well as on an equatorial plane, normal to the symmetry plane, cutting through the interior of the grid. Mach number contour lines of the solution obtained on the adapted grid are also seen in Fig. 11,

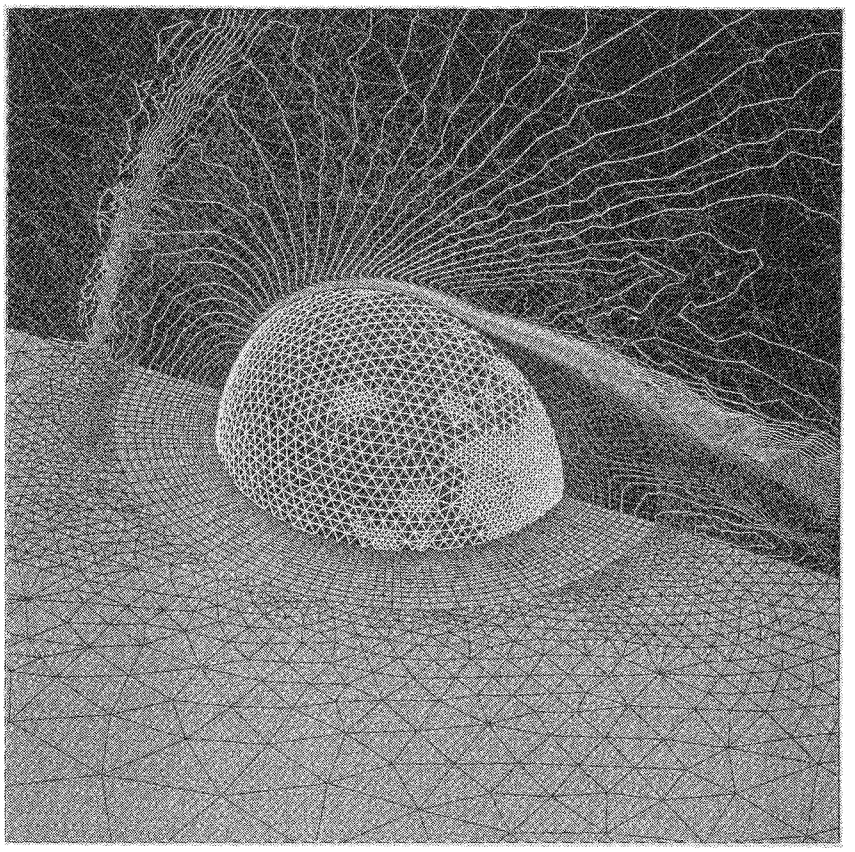


Fig. 11 Isometric view of the tessellation on the wall surface, symmetry plane, and an equatorial plane cutting through the interior of the hybrid grid. The hybrid grid, embedded isotropically in the tetrahedral region and directionally in the prismatic region, comprises ~2500 wall boundary nodes and ~275,000 tetrahedra. Mach number contour lines of the solution are also shown for the solution obtained on the embedded hybrid grid for supersonic, laminar flow at $M_\infty = 1.4$ and $Re = 1 \times 10^3$ ($M_{min} = 0$, $M_{max} = 2$, and $\Delta M = 0.05$).

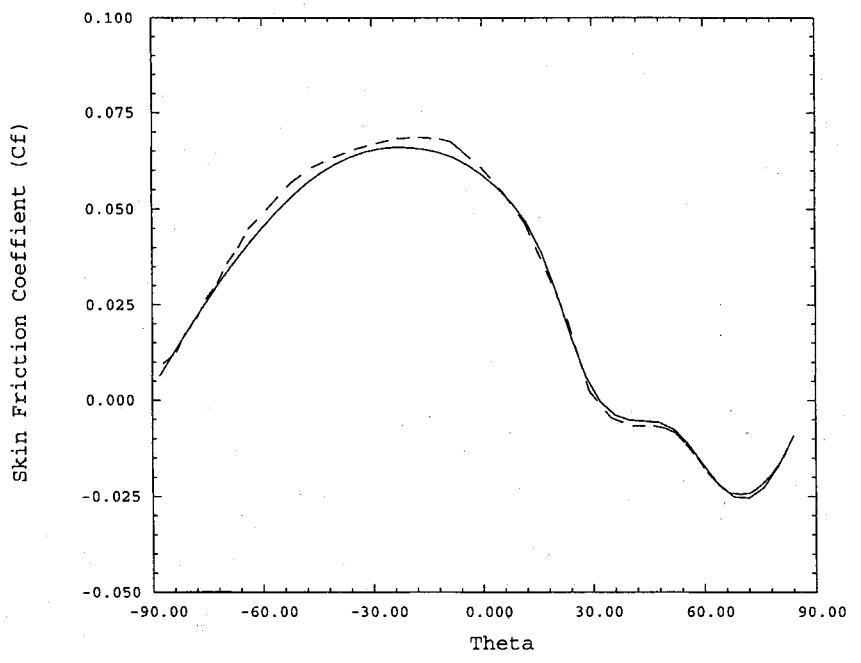


Fig. 12 Comparison of skin-friction coefficients at the wall, on the equatorial plane normal to the symmetry plane: ——, adaptive hybrid grid solution and – – –, globally refined hybrid grid.

superimposed on the grid. A spherical bow shock is observed upstream of the blunt body. The three-dimensional boundary layer becomes thicker on the aft section of the body, and it eventually separates and forms a wake. It is clearly seen that embedding in the tetrahedral region is focused near the shock location just outside of the prismatic-tetrahedral interface. The prismatic region is also directionally refined near the upstream and downstream sections of the body. This is due to the flow upstream accelerating rapidly from the upstream stagnation point and the flow downstream seperating that causes flow gradients in the lateral directions that are detected by the directional adaptive algorithm. The embedded hybrid grid comprises ~2500 wall boundary nodes and ~275,000 tetrahedra. Numerical solution is also computed on a grid obtained by globally embedding the initial mesh. This is done

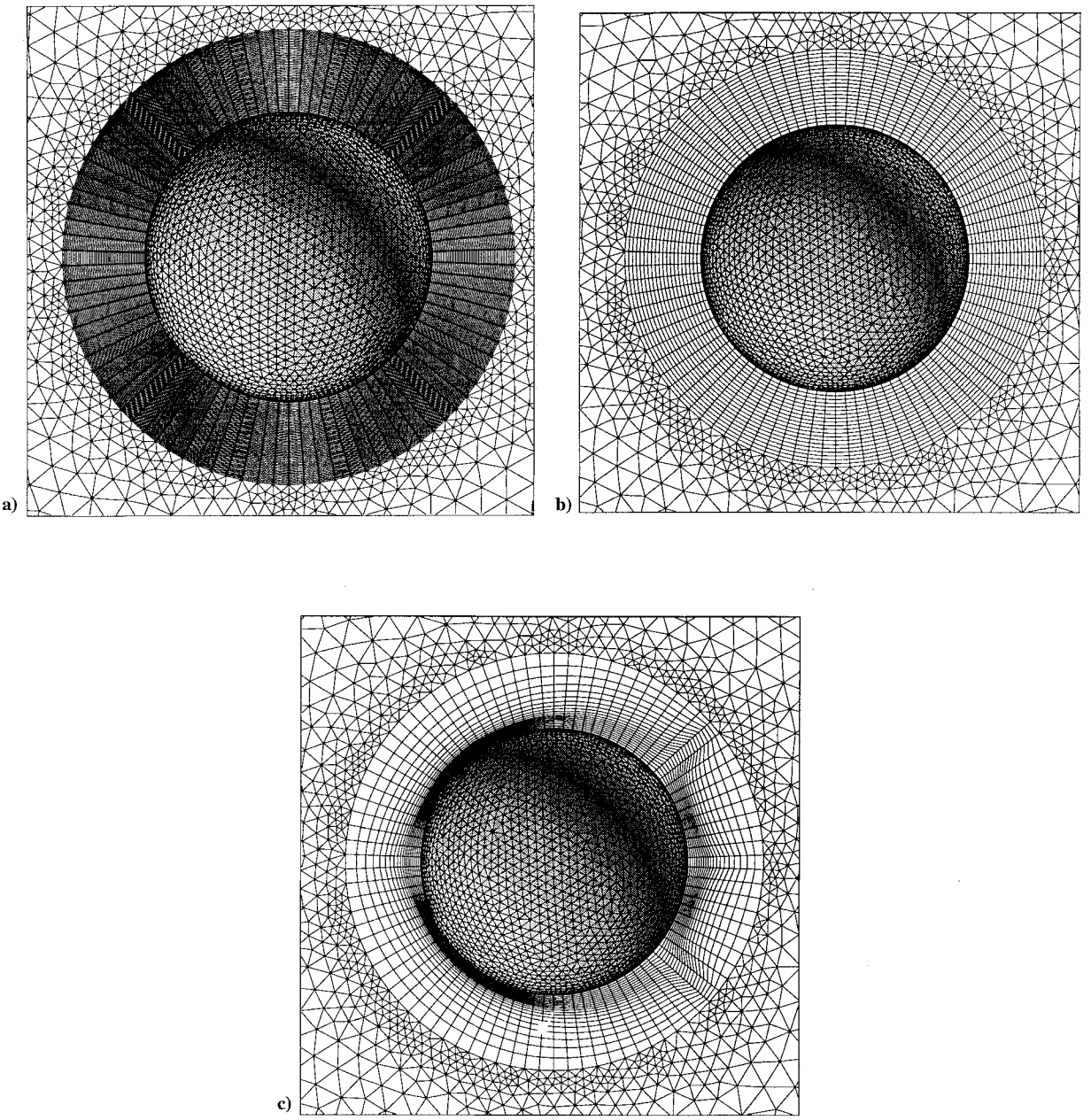


Fig. 13 Tessellation on the symmetry plane showing the clustering of prismatic layers: a) a hybrid grid with a fine prismatic mesh comprising 60 layers of equispaced prisms, b) a hybrid grid with a coarse prismatic mesh comprising 20 layers of equispaced prisms, and c) the hybrid grid with the coarse prismatic mesh obtained after redistribution of the prismatic layers.

by refining each tetrahedron isotropically into eight subcells and each prism directionally into four subcells. This globally embedded grid comprises ~5700 wall boundary nodes and ~800,000 tetrahedra. A comparison of the skin-friction and pressure coefficients computed at the wall for the adaptively embedded case and the globally embedded case is shown in Fig. 12. The local grid embedding strategy obtains essentially the same results as the globally fine grid while significantly minimizing the number of grid points and cells.

Redistribution of Prisms

In the previous cases studied, the grid points were clustered near the wall using a grid spacing chosen a priori so as to provide a suitable resolution in the normal-to-the-surface direction. The effect of grid redistribution in the viscous region is now shown by selecting an initial grid that has a much larger wall spacing ($\delta_w = 10^{-2}$) than is optimally required, and the prism layers are equispaced as shown in Fig. 13a. This grid has ~1400 wall nodes and ~100,000 tetrahedra. Based on an initial solution obtained on this grid, the redistribution algorithm is used to recompute the values of δ_w at all of the wall boundary nodes, using y^+ as the detection parameter. The hybrid grid with the redistributed prismatic region is shown in Fig. 13b. Observing the grid in the fore section, it is seen that the redistribution algorithm reclusters the grid substantially by attracting the nodes very close to the wall to resolve the large gradients in the normal direction. In the aft region, the boundary layer thickens substantially and separates, and the algorithm is seen to push the nodes away from the wall. A hybrid grid with the same surface triangulation but comprising a fine mesh in the prismatic region with 60 equispaced layers of prisms and a wall spacing of $\delta_w = 10^{-3}$ is shown in Fig. 13c. A comparison of the values of C_f and C_p computed for the three cases is shown in Fig. 14. The figure clearly shows that the initial grid performs poorly in the fore section due to lack of sufficient resolution to resolve the large viscous stresses, whereas it performs better in the aft region where the gradients are small due to separation of the boundary layer. The solution obtained on the redistributed grid, however, compares well with the solution obtained on the hybrid grid with the globally

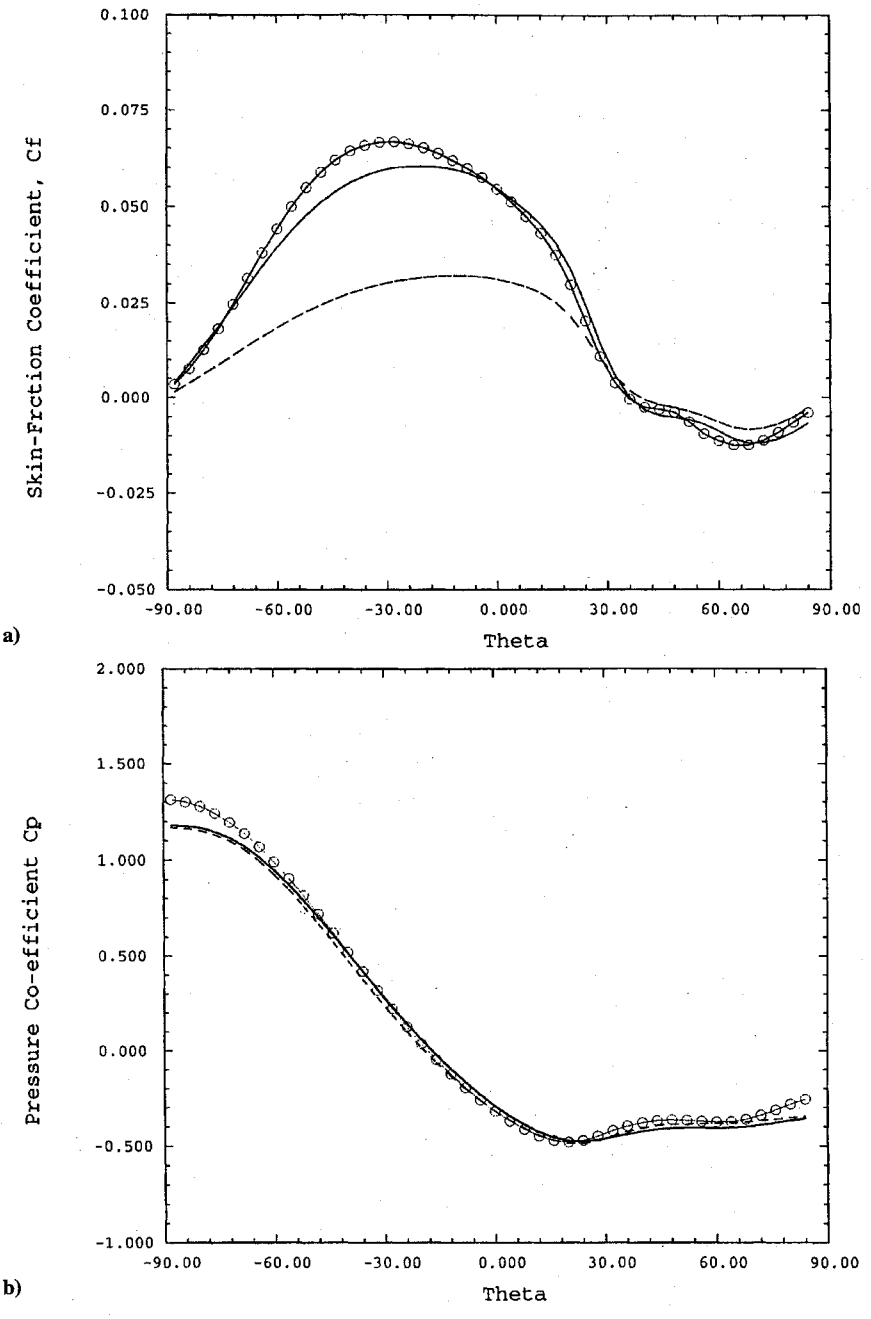


Fig. 14 Comparison of a) skin-friction coefficients and b) pressure coefficients at the wall, on the equatorial plane normal to the symmetry plane: – – –, initial coarse grid; ——, redistributed coarse grid; and o – o – o, fine prismatic grid.

fine prismatic region. The redistribution algorithm thus improves the capability to better capture the directional viscous stresses by attracting or repelling the nodes along the normal marching directions while at the same time minimizing the number of prismatic layers.

VI. Conclusions

The edge-based solver was transparent to the type of element topology in the grid. The directional refinement of prisms provided a fast approach with low memory requirement as well as preserved the structure of the prismatic grid as it was essentially a two-dimensional refinement scheme. The two-dimensional data structures employed for the prisms reduced the memory requirement of the hybrid solver. The dual grid adaptation did not require any special algorithm for coupling the prisms and tetrahedra at the interface. The hybrid approach to Navier–Stokes computations yielded the same accuracy as that of an equivalent case employing an all-prismatic mesh. Locally adapted hybrid grid yielded the same accuracy as that of a globally embedded mesh. It was also observed that the sudden change in grid topology at the prism–tetrahedra interface did not affect the accuracy of the solution. Furthermore, the numerical solution was not dependent on the relative location of the prism–tetrahedra interface. Adaptive redistribution of prismatic layers was demonstrated as an efficient strategy to complement the directional division of prisms and improved the solution accuracy by better resolving the directional viscous stresses.

Acknowledgments

This work was supported by NASA Grant NAG1-1459, Texas Advanced Technology Program Grant 003658-413, Advanced Research Projects Agency Grant DABT 63-92-0042, and National Science Foundation Grant ASC-9357677 (National Young Investigator program). Supercomputing time was provided by the Numerical Aerodynamic Simulation Division of the NASA Ames Research Center as well as by the High Performance Computing Facility of the University of Texas at Austin.

References

[1]Nakahashi, K., "Optimum Spacing Control of the Marching Grid Generation," AIAA Paper 91-0103, Jan. 1991.

[2]Kallinderis, Y., and Ward, S., "Prismatic Grid Generation for 3-D Complex Geometries," *AIAA Journal*, Vol. 31, No. 10, 1993, pp. 1850–1856.

[3]Kallinderis, Y., Khawaja, A., and McMorris, H., "Hybrid Prismatic/Tetrahedral Grid Generation for Complex Geometries," AIAA Paper 95-0211, Jan. 1995 and AIAA Paper 95-1685, June 1995.

[4]Lohner, R., and Baum, J., "Numerical Simulation of Shock Interaction with Complex Geometry Three-Dimensional Structures Using a New Adaptive H-Refinement Scheme on Unstructured Grids," AIAA Paper 90-0700, Jan. 1990.

[5]Kallinderis, Y., and Vijayan, P., "An Adaptive Refinement-Coarsening Scheme for 3-D Unstructured Meshes," *AIAA Journal*, Vol. 31, No. 8, 1993, pp. 1440–1447.

[6]Biswas, R., and Strawn, R., "A New Procedure for Dynamic Adaptation of Three Dimensional Unstructured Grids," AIAA Paper 93-0672, Jan. 1993.

[7]Vijayan, P., and Kallinderis, Y., "A 3-D Finite-Volume Scheme for the Euler Equations on Adaptive Tetrahedral Grids," *Journal of Computational Physics*, Vol. 113, Aug. 1994, pp. 249–267.

[8]Kallinderis, Y., "A Finite-Volume Navier–Stokes Algorithm for Adaptive Grids," *International Journal of Numerical Methods in Fluids*, Vol. 15, No. 2, 1992, pp. 193–217.

[9]Holmes, D. G., and Connell, S. D., "Solution of 2D Navier–Stokes Equations on Unstructured Adaptive Grids," AIAA Paper 89-1932, July 1989.

[10]Kallinderis, Y., and Baron, J. R., "A New Adaptive Algorithm for Turbulent Flows," *Computers and Fluids Journal*, Vol. 21, No. 1, 1992, pp. 77–96.

[11]Parthasarathy, V., and Kallinderis, Y., "A New Multigrid Approach for Three Dimensional Unstructured Adaptive Grids," *AIAA Journal*, Vol. 32, No. 5, 1994, pp. 956–963; see also AIAA Paper 94-0078, Jan. 1994.

[12]Parthasarathy, V., and Kallinderis, Y., "Directional Viscous Multigrid Using Adaptive Prismatic Meshes," *AIAA Journal*, Vol. 33, No. 1, 1995, pp. 69–78.