An Overset Adaptive Cartesian/Prism Grid Method for Moving Boundary Flow Problems

Z.J. Wang[*] and Ravishekar Kannan[†]
Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824

The use of overset grids in CFD started more than two decades ago, and has achieved tremendous success in handling complex geometries. In particular, overset grids have the advantage of avoiding grid remeshing when dealing with moving boundary flow problems. Traditionally overset grids were mainly used with structured grids to simplify the grid generation process, because a complex computational domain can be more easily meshed after it is partitioned into sub-domains with overset interfaces than with patched interfaces. More recently, unstructured grids are also used in the overset grid system to further simplify grid generation for each sub-domain. In this paper, two particular unstructured grids are advocated for moving boundary flow simulation, i.e., the use of overset adaptive Cartesian/prism grids. Semi-structured prism grids are generated around solid walls. These prism grids then overlap a single adaptive Cartesian background grid. With the adaptive Cartesian grid, the mesh resolution of the prism grid near the outer boundary can easily match that of the oversetting Cartesian grid cells. In addition, the tree-based data structure of the Cartesian grid can be used efficiently in hole-cutting and donor cell identification. The overset adaptive Cartesian/prism grid method is tested for both steady and unsteady flow computation. It is demonstrated that moving boundary flow computations can be automated with minimum user interferences.

Nomenclature

A_j	=	Area of the triangle j
c_f	=	Skin friction coefficient, $\tau_w/(0.5\rho U_\infty^2)$
c_p	=	Static Pressure Recovery Coefficient
Cd	=	Drag Coefficient
x	=	Streamwise distance from the center of the sphere
D	=	Diameter of the sphere
F^i	=	Inviscid flux vector
F^v	=	Viscous flux vector
$\mathbf{M}_i$	=	Marching vector at a node i.
$\theta_{\max}$	=	Maximum angle between the marching vector and the face normals of its node-manifold
$\boldsymbol{m}$	=	Area weighted normal vector of a triangle
Q	=	Vector of conserved variables
$\boldsymbol{r}$	=	Position vector
$\boldsymbol{r}_j^c$	=	Position vector of the centroid of triangle j
Re	=	Reynolds number
$\mathbf{v}_g$	=	Grid velocity
v_{gn}	=	Surface normal grid velocity component
V_i	=	Volume of control volume i

Introduction

The use of unstructured grids in computational fluid dynamics (CFD) has become widespread during the last two decades due to their ability to discretize arbitrarily complex geometries and the flexibility in supporting solution-

[*]Currently Associate Professor of Aerospace Engineering, Iowa State University, Associate Fellow of AIAA
[†]Graduate Research Assistant, 4110 Engineering Building

based grid adaptations to enhance the solution accuracy and efficiency.[1-7] In the early days of unstructured grid development, triangular/tetrahedral grids were employed primarily in dealing with complex geometries. Recently, mixed or hybrid grids including many different cell types have gained popularity because of the improved efficiency and accuracy over pure tetrahedral grids. For example, hybrid prism/tetrahedral grids,[8] mixed grids including tetrahedral/prism/pyramid/hexahedral cells,[9] and adaptive Cartesian grid methods[10-17] have been used in many applications with complex configurations. In addition, solution algorithms for computing steady flows on unstructured and hybrid grids have evolved to a high degree of sophistication. The state-of-the-art spatial discretization algorithm is probably the second-order Godunov-type finite volume method.[18] For time integration, explicit algorithms such as multi-stage Runge-Kutta schemes are the easiest to implement. Convergence acceleration techniques such as local time-stepping and implicit residual smoothing[1] have also been employed in this context. However, for large-scale problems and especially for the solution of viscous turbulent flows, implicit schemes[19-25] are required to speed up the convergence rate. The success demonstrated by unstructured grids for steady flow problems has prompted their applications to unsteady moving boundary flow problems. For a moving boundary flow problem, the computational grids must move with the moving boundaries. The most straightforward approach is to deform the computational grid locally using a spring-analogy type algorithm to follow the motion of the moving boundaries.[26] The approach is very efficient because it does not require solution interpolation. A disadvantage of the approach is that the grid integrity can be destroyed by large motions or shear-type of boundary motions. To remedy this drawback, local remeshing can be applied whenever the grid becomes too skewed. With local remeshing, solution interpolations from the old to the new grid become necessary. The hybrid approach of combining grid deformation with grid local remeshing seems to be the state-of-the-art in handling moving boundary problems, and has been used successfully for a variety of applications.[27,17]

Another powerful approach for moving boundary flow problems is the overset Chimera grid method.[28] Originally, the Chimera grid method was used to simplify domain decomposition for complex geometries using structured grids. The method is particularly useful for moving boundary flow simulations since grid remeshing can be avoided.[29] However, frequent hole-cutting and donor cell searching may be necessary to facilitate communications between the moving Chimera grids. With continuous improvement over the last one and half decades, the Chimera grid method has achieved tremendous success in handling very complex moving boundary flow problems. More recently, in order to further simplify the grid generation process, unstructured grids are also used in a Chimera grid system for moving boundary flow computations, making the approach even more flexible in handling complex geometries.[30]

In this paper, we advocate the use of an overset adaptive Cartesian/prism grid method for moving boundary flow computations. The method combines the advantage of adaptive Cartesian/prism grid in geometry flexibility with that of Chimera approach in tackling moving boundary flow without grid remeshing. There are several reasons why an adaptive Cartesian grid is used for moving boundary problems: 1. Cartesian cells are more efficient in filling space given a certain length scale than triangular/tetrahedral cells; 2. Searching operations can be performed very efficiently with the Octree data structure; 3. Solution based and geometry-based grid adaptations are straightforward to carry out. The grid generation process is as follows. Body-fitted prism grids are generated first near solid bodies to resolve viscous boundary layers. An adaptive Cartesian grid is then generated to cover the outer domain and serve as the background grid for bridging the "gaps" between the prism grids. The prism grids are used to generate holes in the adaptive Cartesian grid to facilitate data communications. If the bodies move, the prism grids move with the bodies, while the Cartesian grid remains stationary. After a few (tens of) time steps, new holes are cut out of the Cartesian grids, and new donor cells are also identified. Solution fields are interpolated from the old Cartesian grid to the new grid using cell-wise linear reconstruction.

The paper is organized as follows. In the next section, the overset adaptive Cartesian/prism grid generation approach will be presented, together with illustration examples. After that, a finite volume, Godunov-type second-order dual time-stepping method for dynamic grids is described. In Section 4, several steady and unsteady moving boundary problems are computed. Grid refinement studies are performed to ensure the computational solutions are grid independent. Computational results are compared with experimental data and other simulations whenever possible. Finally conclusions from this study are summarized in Section 5.

Generation of Overset Cartesian/Prism Mesh

A. Prism Grid Generation

Since we do not address geometry modeling issues in this paper, it is assumed that water-tight surface grids are already generated with other packages, and serve as inputs to the present Cartesian/prism grid generator. The generation of prismatic grids follows the basic idea of many similar approaches, i.e., through surface extrusion in the approximate surface normal direction.[31-33] Even after many years of development, we are still searching for a "fool-proof" prism grid generator, which is capable of handling arbitrarily complex surface shapes. The current algorithm is still not "fool-proof", and we plan to continuously improve its robustness and efficiency. It does borrow many ideas already developed, and an idea to determine the optimum direction for a given surface grid node seems to be new, and is implemented. The steps employed to generate the prism grid are outlined next.

1. Obtaining the Marching Vectors

This is the quintessential aspect of prism grid generation. Kallinderis[31] defined the term node-manifold as the list of faces confining the node to be marched. Common sense tells us that the marching vector at any node should not make an angle greater than 90^o with the face normals of its manifold. If the above criterion is violated, the tip of the marching vector is not visible from all the faces of the manifold. This results in intersections of the prism grid cells. So the paramount objective here is to ensure that the marching vectors satisfy the visibility criterion. The secondary objective is to impose orthogonality. Strict orthogonality can be achieved if the marching vectors are identical to the outer normal of the manifold. In any case, the maximum of the angles between the marching vector at a node and the face normals of its node-manifold is obtained. This angle, $\theta_{\max}$, needs to be as small as possible (if $\theta_{max} = 0$, the marching vector is perpendicular to its node-manifold). The optimal orientation for the marching vector can be obtained iteratively. An angle based weighting is used to obtain the initial guess for the marching vector. The marching vector is then refined locally to reduce the maximum of the angles it makes with the face normals of its node-manifold. An optimal orientation for the marching vectors needs to be obtained in order to fulfill the paramount objective i.e. ensuring visibility. In many real life geometries, the angle based weighting scheme yields a marching vector which is invisible from some of the nodes in its node-manifold. Examples of the above include the trailing edge of an airfoil, the tip of the nose and the tail of a store and in the nacelles of aircrafts. The algorithm for obtaining the optimal marching vector is discussed below.

For each marching vector $\mathbf{M}_i$, a set of vectors $\{\mathbf{S}\}$ which make a small angle δ (about 1^o) with $\mathbf{M}_i$ is obtained. For each of the vectors in the above set, the maximum of the angles made with the face normals of the node-manifold in consideration is obtained. Thus a set of maximum angles is obtained. The minimum value in the above set is determined**.** If the minimum value is smaller than $\theta_{\max}$ of $\mathbf{M}_i$, then the vector associated with the minimum value is the new marching vector $\mathbf{M}_i$. This process is repeated till the marching vector remains the same**.**

An inverse distance based smoothing given by Kallinderis was used to further smooth the marching vectors, i.e.,

$$\mathbf{M}_i = \alpha \mathbf{M}_i + \frac{\sum_j (1-\alpha)\frac{\mathbf{M}_j}{d_{ij}}}{\sum_j \frac{1}{d_{ij}}}, \tag{2.1}$$

where $\alpha = 1 - \cos(\theta_{max})$, node j is a neighboring node of node i, d_{ij} is the distance between node i and node j. The summation is from 1 to number of neighbors of i. Thus if node i is a critical node, then the orientation of $\mathbf{M}_i$ is virtually unaltered.

2. Marching Step Based on the Curvature

Once the marching vectors are generated, the nodes need to be positioned at the next layer. One of the many traits of a good body conforming grid is that the curvature of the front needs to decrease from one layer to the next layer. It would be unwise to maintain a constant layer thickness at all nodes in a particular layer. It could be figured intuitively that the marching vectors at concave nodes need to be marched faster and the marching vectors at the convex nodes need to be marched slower. The ratio of the marching steps between 2 adjacent nodes needs to lie between 0.5 and 2.0. The above is carried out to ensure smooth transition. The average thickness increases exponentially with increasing layers. The average thickness is the marching step when the front in consideration is a planar surface i.e. the marching vector at a node is the same as any of the face normals of its

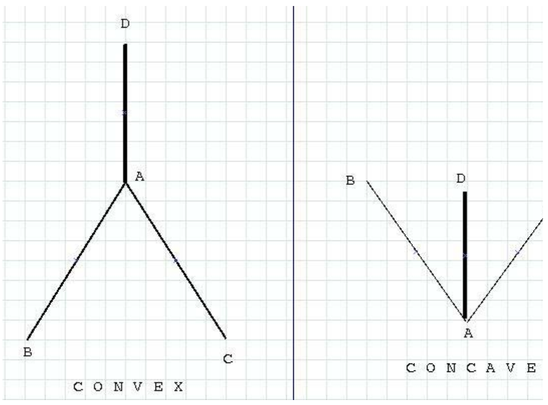

Figure 1. Two Dimensional Concave and Convex models

manifold.

A new scheme was devised to estimate the surface curvature. For better understanding, let us start with a 2 dimensional model. Figure 1 depicts the marching vectors for a two dimensional case. AD is the un-smoothed marching vector in Figure 1. For the convex case, the angle between AD and AC is greater then 90^o. Similarly the angle between AD and AB is greater then 90^o. For the concave case, the angle between AD and AC is less than 90^o. Similarly the angle between AD and AB is less than 90^o. This idea can be extended to three dimensions. In the three dimensional case, the angles between the un-smoothed marching vector and the edges connecting the node in consideration are determined. If each of the above angles is greater than 90^o, the surface is convex. If each of the above angles is lesser than 90^o, the surface is concave. In reality some saddle points occur. For such a scenario, the average of the angles between the un-smoothed marching vector and the edges connecting the node in consideration is determined. If this average is greater than 90^o, the surface is treated as a convex surface else it is treated as a concave surface.

3. Mean Filter Smoothing Algorithm

This filter[34] is employed to smooth the nodes in the current layer. Each iteration of this filter consists of the following steps

a. For each triangle *i*, compute the area weighted averaging normal:

$$\mathbf{m}_i = \frac{\sum_j A_j \mathbf{n}_j}{\sum_j A_j}. \tag{2.2}$$

b. Normalize the averaged normals;

c. For each mesh vertex, perform the following vertex updating procedure:

$$\mathbf{r}_{new} = \mathbf{r}_{old} + \frac{\sum_j A_j (\mathbf{m}_j \bullet \mathbf{r}_j^c)\mathbf{m}_j}{\sum_j A_j}, \tag{2.3}$$

However the position of the new node is updated with the new value if

a. The new value of θ_{max} is less than some threshold value;

b. The thickness of the layer obtained using the modified position is greater than a critical value.

4. Checking for Intersections

Most of the times, the efficiency and the accuracy of the flow solver depend on the grid generated. Even extremely robust solvers deliver erroneous results if the grid is invalid. A grid is said to be invalid if intersections exist. Intersections give rise to negative volumes and hence forcing the solver to diverge. So an important issue in prism grid generation is to identify intersections. As all the prisms need to be checked for intersections, a fast scheme was developed to determine intersections. Figure 2 shows a prism with no intersections. Triangle ABC is the triangular face at the current layer. Triangle DEF is the triangle at the next layer.

Intersections occur when either of the below occurs

1. Nodes A and D do not lie on the same side of the planes BCEB, BCFB, EFBE and EFCE
2. Nodes B and E do not lie on the same side of the planes ACFA, ACDA, DFAD and DFCD
3. Nodes C and F do not lie on the same side of the planes ABEA, ABDA, EDAE and EDBE

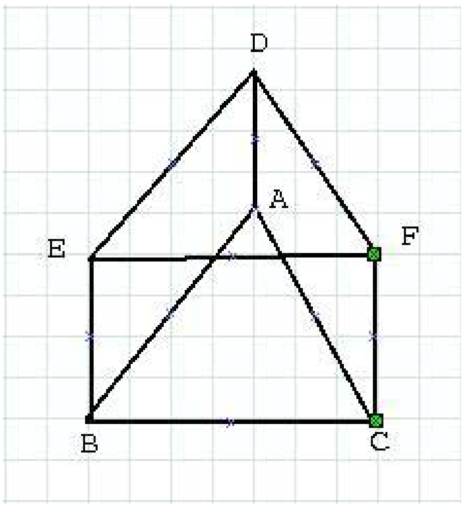

Figure 2. An example of a prism with no intersections

B. Generation of an adaptive Cartesian grid

After the prism grid generation, an adaptive Cartesian grid was generated automatically matching the grid resolution near the outer boundaries of the prismatic grids. In order to support arbitrary local grid adaptations, the Octree data structure was used. The following steps were employed to generate the initial grid:

1. Generate a single root node based on the domain size;
2. Recursively subdivide the root node until all cells are smaller than the specified maximum cell size;
3. Identify all Cartesian cells intersecting the outer boundaries of the prismatic grids;
4. Recursively refine the intersected cells until all the cells intersecting the interfaces match the grid resolution of the prismatic cells;

The final adaptive Cartesian grid was smoothed so that the length scales between 2 neighboring cells do not differ by a factor more than 2 in any coordinate direction. In addition, several buffer layers with the same grid resolution near the outer boundaries of the prismatic grids were used to minimize the local discretization error.

C. Automated Hole Cutting and Donor Cell Identification

The use of overset adaptive Cartesian and prismatic grids has the potential of handling moving boundary problems without any user interferences. A critical element in achieving this level of automation is an automated hole cutting algorithm, in which invalid Cartesian grid cells (cells inside the solid boundary) are excluded from the calculation, and donor cells are identified for the hole boundary cells (inner boundary Cartesian cells) and the prism outer boundary cells. The efficiency of the hole cutting algorithm is critical since hole-cutting is performed many times in a moving boundary flow simulation. To achieve the maximum efficiency, search trees were used extensively. The hole cutting algorithm consists of the following steps

1. Blank all the Cartesian cells which are inside the solid boundary;
2. Use the Alternating Digital Tree (ADT) to find the prismatic cells, which bound the centroids of the hole boundary cells. These prismatic cells are the prismatic donor cells;
3. Generate a list of outer boundary cells and use the Octree tree to identify the Cartesian Cells which bound the cell centroids of the last layer cells of the prism grids. These Cartesian cells are the Cartesian donor cells.

The overset Cartesian-prism grid generation algorithm was tested for several real life geometries including a store and a fighter aircraft. They are shown in Figure 3.

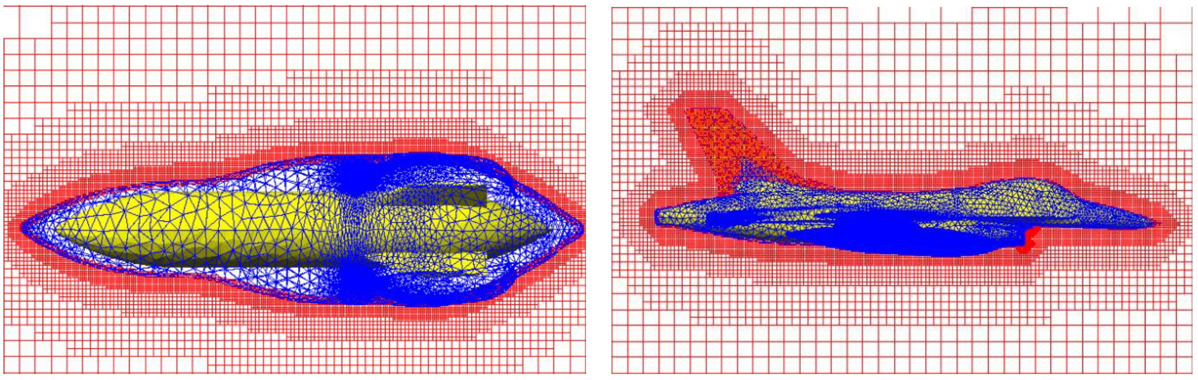
Figure 3. Example overset adaptive Cartesian/prism grids

NUMERICAL METHOD

A. Finite Volume Method for Dynamic Grids

The time-dependent Reynolds-averaged Navier-Stokes equations for dynamic grids can be expressed in the integral form as

$$\frac{\partial}{\partial t}\int_V Q dV + \oint_S \left(F^i(Q) - Q\mathbf{v}_g \cdot \mathbf{n}\right) dS = \oint_S F^v(Q) dS, \tag{3.1}$$

where S is the surface surrounding the control volume V, $\boldsymbol{n}$ is the out-going unit normal of S, $\boldsymbol{v_g}$ is the velocity of S, and $\boldsymbol{Q}$ is the vector of conserved variables, $\boldsymbol{F^i}$ is the inviscid and $\boldsymbol{F^v}$ the viscous flux vectors. The eddy viscosity for turbulent flow is calculated with the S-A turbulence model.[35] The governing equations for inviscid flow and for fixed control volumes are only sub-sets of Equation (3.1). If we integrate Equation (3.1) in a polygonal control volume V_i, we obtain

$$\frac{\partial}{\partial t}(QV_i)+\sum_f\left(F^i(Q)-Qv_{gn}\right)_f dS_f=\sum_f F_f^v(Q)dS_f, \tag{3.2}$$

where the summation index f represents all the faces surrounding control volume V_i, and $v_{gn}=\mathbf{v}_g\cdot\mathbf{n}$. The inviscid flux is calculated using Roe's approximate Riemann solver[36] with reconstructed state variables at both sides of a face. A least square linear reconstruction scheme of the primitive variables is used. For the viscous terms, an efficient second-order centered scheme is employed. Details of the flow solver are contained in. [14, 15]

The conservation of a constant flow is a necessary condition for any viable numerical scheme. Otherwise mass, momentum or energy would be produced unphysically by the numerical simulation. If we examine Equation (3.2), in order to preserve a uniform free stream, we must have:

$$\frac{\partial V_i}{\partial t}=\sum_f v_{gn}dS_f \tag{3.3}$$

This is the so-called Geometric Conservative Law[37] in its semi-discretized form. Assume that the grid velocity is computed at time level n+1/2. Then we can us the following time discretization to achieve second-order accuracy

$$\frac{V_i^{n+1}-V_i^n}{\Delta t}=\sum_f v_{gn}dS_f \tag{3.4}$$

Instead of having the grid velocity v_{gn} satisfy Equation (3.4), we utilize the equation to calculate v_{gn}. In this case, we are sure that GCL is guaranteed. To this end, we employ a simple fact: the volume that a cell sweeps over is equal to the total of the volumes swept by its faces, i.e.,

$$V_i^{n+1}-V_i^n=\sum_f \Delta V_f \tag{3.5}$$

where ΔV_f represents the volume swept by face f. Comparing Equations (3.4) and (3.5), we arrive at the following equation:

$$\Delta V_f=\Delta t v_{gn}dS_f \quad or \quad v_{gn}=\frac{\Delta V_f}{\Delta t\, dS_f} \tag{3.6}$$

B. Time Integration Algorithm

Once the fluxes are evaluated for each cell face using the preceding finite volume scheme, the semi-discrete form of the governing equations is then integrated in time. For convenience, we rewrite Equation (3.2) as the following nonlinear system:

$$\frac{\partial(QV)_i}{\partial t}+R_i(Q)=0, \tag{3.7}$$

where R_i is the residual given by

$$R_i(Q) = \sum_f \left(F^i(Q) - Qv_{gn} - F^v(Q)\right)_f dS_f . \tag{3.8}$$

The easiest time integration algorithm for equation (3.7) is the explicit multi-stage Runge-Kutta method. However for viscous flow computational, the heavily clustered mesh imposes a too severe time step limit. We therefore employ the following family of implicit schemes

$$\frac{Q_i^{n+1}V_i^{n+1} - Q_i^n V_i^n}{\Delta t} + (1-\theta)R_i\,(Q^{n+1}) + \theta R_i\,(Q^n) = 0. \tag{3.9}$$

If $\theta = 0$, the scheme is the backward Euler method. If $\theta = 1/2$, the resulting scheme known as the Crank-Nicolson method is second-order accurate in time. Equation (3.9) represents a nonlinear system of coupled equations, which has to be solved at each time step. It can be solved by introducing a pseudo-time variable τ,[38]

$$\frac{\partial (QV)_i}{\partial \tau} + R_i^*(Q) = 0, \tag{3.10}$$

and 'time-marching' the solution using local pseudo-time $\Delta\tau$, until Q converges to Q^{n+1}. In (3.10), Q is the approximation of Q^{n+1} and the unsteady residual $R_i^*(Q)$ is defined as

$$R_i^*(Q) = \frac{Q_i V_i^{n+1} - Q_i^n V_i^n}{\Delta t} + (1-\theta)R_i\,(Q) + \theta R_i\,(Q^n) \tag{3.11}$$

Obviously, Equation (3.11) can be solved by using a variety of numerical schemes including the explicit multi-stage Runge-Kutta method. Note that this dual time method with an explicit inner iteration scheme should be many times faster than the explicit time marching method because local time-stepping in the pseudo time can be used to accelerate the convergence rate. Of course the best efficiency is expected to be achieved by an implicit inner iteration schemes. An efficient Block Lower-Upper symmetric Gauss-Seidel (BLU-SGS) approach[17] is employed to solve the inner iteration.

C. Boundary Conditions

In order to treat the boundary cells as transparently as possible, a ghost cell is generated for each boundary cell. Then the solution variables at the ghost cell are computed from the boundary cell according to the physical boundary condition. For a steady inviscid flow, the velocity components at the ghost cell for a solid wall boundary are computed as:

$$u_{ghost} = u - 2n_x v_n \qquad v_{ghost} = v - 2n_y v_n , \tag{3.12}$$

where v_n is the normal velocity given by

$$v_n = un_x + vn_y . \tag{3.13}$$

Meanwhile, the density and pressure of the ghost cell are set to be the same as those of the boundary cell. For unsteady moving boundary problems, the condition must be adjusted since the boundary face is moving. Then the normal velocity should be modified as

$$v_n = un_x + vn_y - v_{gn} . \tag{3.14}$$

Similarly for an unsteady viscous surface boundary, the velocity components at the ghost cell are computed using the following equation,

$$u_{ghost} = -u + 2n_x v_{gn} \qquad v_{ghost} = -v + 2n_y v_{gn}. \tag{3.15}$$

In the far field, a characteristic analysis based on Riemann invariants is used to determine the values of the flow variables on the outer ghost cells. This analysis correctly accounts for wave propagations in the far field, which is important for rapid convergence to steady state and serves as a 'non-reflecting' boundary condition for unsteady applications.

For a hole boundary face or an interpolation boundary face, the fluxes are required at the face center to update the conservative variables. In order to compute the flux at the face center, the solution at the face center is required. Once a donor cell for the face center is found from another grid, the solution is assumed linear over the donor cell, and then the solutions at the face center are computed using a first-order Taylor expansion.

Test Results

In this Section, the overset adaptive Cartesian/prism grid method is tested for both stationary and moving boundary flow problems. The following three cases are presented.

A. Viscous Flow over a Stationary Sphere

First, we present the computational results of flow over a sphere at various Reynolds Numbers to validate the overset adaptive Cartesian/prism grid solver. The flow over the sphere was simulated at Reynolds numbers of 118, 800 and 1.1x10^6. In all the simulations, the incoming flow has a Mach number of 0.37. The cells are clustered near the solid boundary and in the wake to capture the viscous effects. Local time stepping was employed with CFL numbers in the range of 40-100. No slip and no penetration boundary conditions were imposed at the wall. Characteristic boundary conditions were imposed at the outer boundary of the computational domain. Both c_p and c_f distributions were obtained and compared with experimental data or other simulations. In order to obtain accurate c_p and c_f distributions, the non-dimensional wall distance y+ at the wall needs to be less than 1. So an iterative approach was followed. The solutions were obtained for a particular grid clustering near the wall. The y+ values were then calculated. If the maximum of the above was greater than 1, the grid was refined. This iterative process was carried on till the maximum y+ was less then 1. To simulate flow turbulence, a RANS Spalart-Allmaras (S-A) model was employed.

A parameter of much interest to design engineers is the total drag coefficient. The total drag is composed of pressure drag and viscous shear drag, which can be easily computed using surface integrals. From the c_f distribution, one can easily compute the separation angle since separation occurs at the angle where c_f changes its sign.

1. Flow at Low Reynolds Number

The overset Cartesian/prism solver was first tested for two low Reynolds number flow cases, i.e., Re = 118 and 800. The coarse and the fine meshes used for Re = 118 are displayed in Figure 4. At a Reynolds number of 118, the flow was steady and there was virtually no shedding of vortices. There was a stationary vortex ring at the rear of the sphere, which was also experimentally observed.[39] At Re = 800, the flow field was unsteady and there was periodic

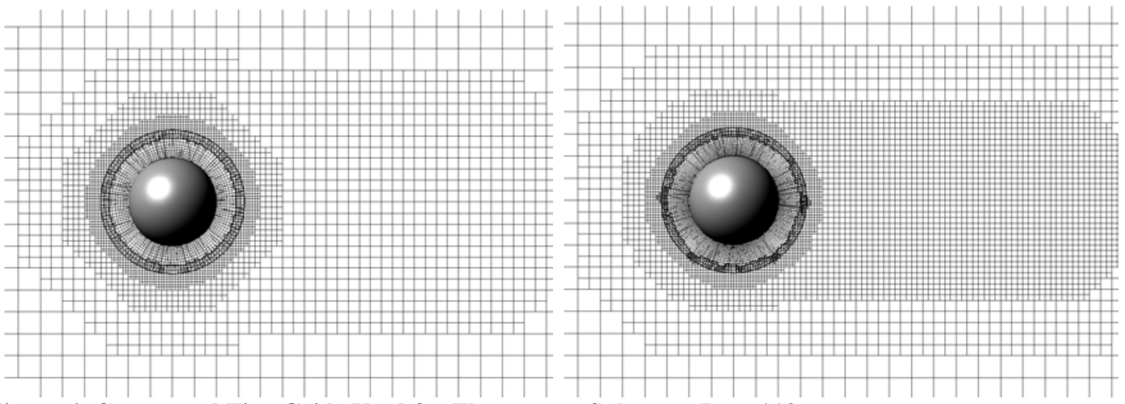
Figure 4. Coarse and Fine Grids Used for Flow over a Sphere at Re = 118

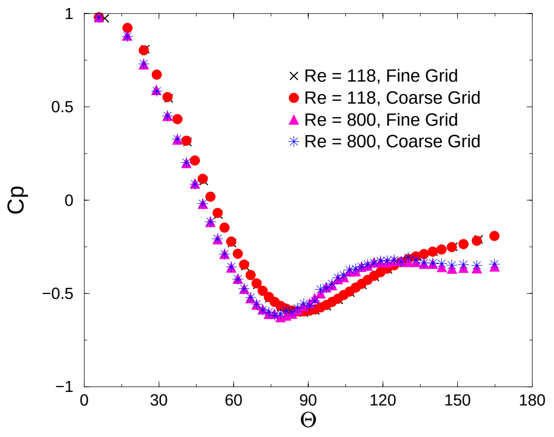

Figure 5. Static Pressure Coefficient at Two Different Reynolds Numbers

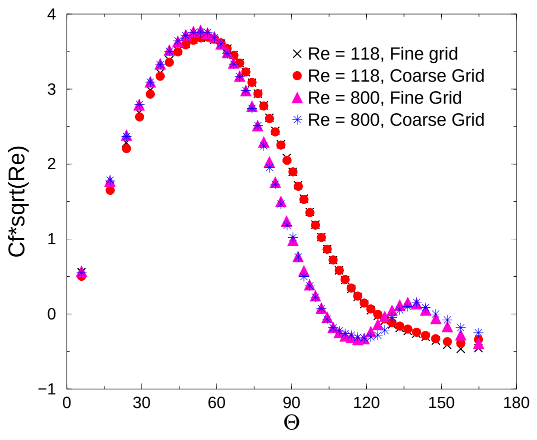

Figure 6. Skin Friction Coefficients at Two Different Reynolds Numbers

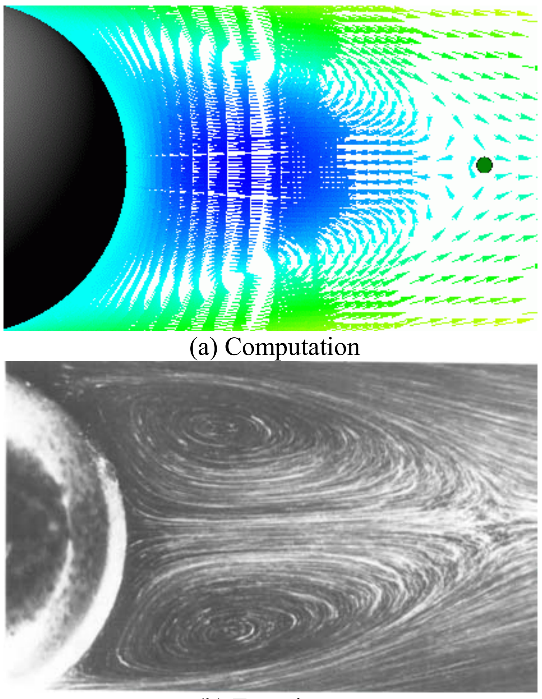

Figure 7. Velocity vector plot showing the separation region at Re = 118

distributions were obtained by time averaging the unsteady flow (but statistically steady) field. The drag coefficients for Re = 118 and 800 are 1.03 and 0.52, and the separation angles are 112.3 and 101.5 degrees respectively.

The skin friction coefficient profile for Re = 800 shown in Figure 6 is interesting. Due to the separation of the boundary layer, c_f changes its sign. However at around 120 degrees, c_f starts to increase. This means that the boundary layer tries to reattach to the sphere. Even though the incoming flow is laminar, the flow becomes unsteady vortex shedding. Therefore the simulation was run in a time accurate mode. Time and spatial (circumferential) averaged c_p and c_f profiles for both cases were obtained and displayed in Figures 5 and 6 for both coarse and fine meshes. Note that there is excellent agreement between the solutions on the coarse and fine meshes, indicating that the numerical solution is nearly grid independent. The velocity vector plot on the fine grid on z = 0 for Re = 118 is compared with an experimental flow pattern in Figure 7. There is very good agreement on the size of the separation region between the experiment and computation. At Re = 800, a vortex pair loop was observed in the wake, as shown in Figure 8, which displays the entropy distribution on plane x = 2D. The vortex pair was also observed experimentally.[43]

The sphere experienced a sideward force at Re = 800 due to the formation of the vortex pair. The magnitude of the side force was about a fifth of the drag force. A time averaging of the side force yielded zero. All the properties like the drag, separation angle and the c_p and c_f

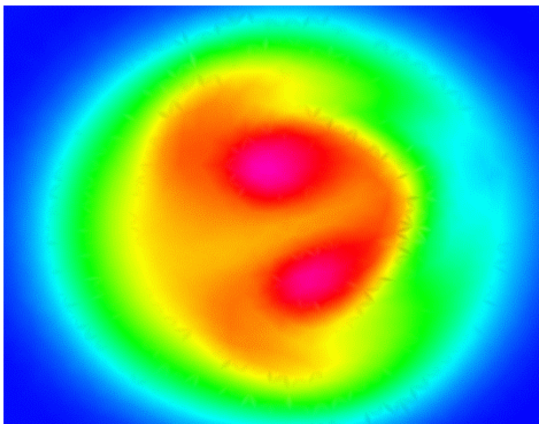

Figure 8. Entropy distribution depicting the vortex pair seen at x/D = 2 for Re = 800

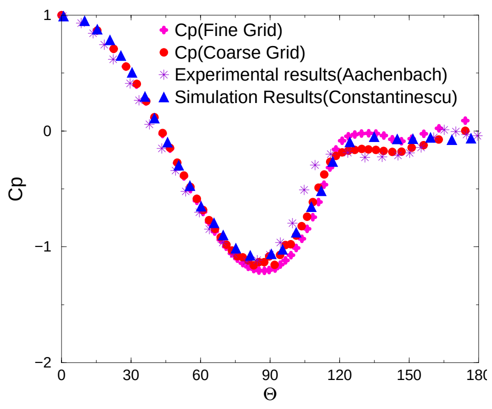

Figure 9. Comparison of Static Pressure Coefficient at Reynolds Number = 1.1e6

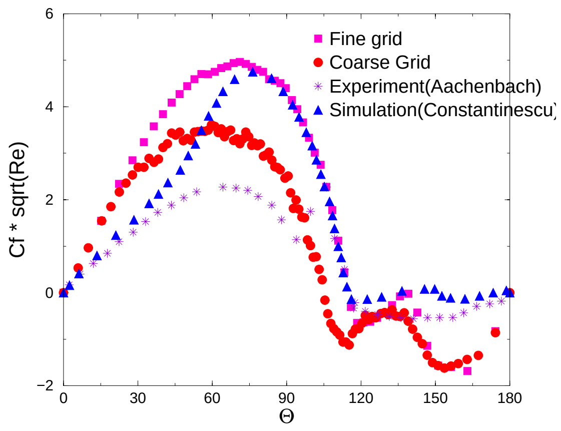

Figure 10. Comparison of Skin Friction Coefficient at Reynolds Number = 1.1e6

after separation. The viscous effects are negligible and the flow is now turbulent. There is an enhanced mixing of momentum. This means the flow has more momentum to move downstream. The c_f increases and peaks at around 140 degrees. However the adverse pressure gradient starts to dominate over the turbulent mixing. The value of c_f starts to decrease and becomes negative again. In contrast, at a Reynolds number of 118, the viscous effects are dominant even after separation. Thus there is no reattachment of the boundary layer to the sphere at Re = 118.

2. Flow at High Reynolds Number

Next turbulent flow over a sphere at Re = 1.1x10^6 was computed with the S-A turbulence model. Again to assess grid sensitivity, coarse and fine meshes were used in the simulation. The coarse mesh has 1.5x10^6 cells while the fine mesh has 2.1x10^6 cells. Experimental data and other simulations are available at this Reynolds for comparison. The computed c_p profiles on both the coarse and fine grids are compared with experimental data by Achenbach[42] and simulation results by Constantinescu[40] in Figure 9. Note that the agreement in c_p is quite good between the present computation and other experimental and computational data.

The computed drag coefficient Cd is 0.09 with a separation angle of 115°. As expected, turbulent mixing increases the separation angle to 115°. As separation is stalled, the c_p curve behaves like its inviscid counterpart till around 90°. This can be seen from the Figure 9. A sharp reduction in the drag occurs at this Reynolds number. In this case, the viscous contribution to the drag force was negligible (around 10%). The separation angle and c_p distribution were in good agreement with the experimental results of Achenbach and the data provided by Schlichting, as presented in Table 1. The drag coefficient obtained from the current simulations was slightly lower than the experimental results. The results of this super-critical case were compared with the DES results of Constantinescu. The Cd obtained from the current simulation was in good agreement with the results of Constantinescu. The c_f distribution did not match other data well. The peak value of c_f distribution and the separation angle obtained by Constantinescu was in accord with the current simulation. However there were some differences at angles close to zero and 180°. This can be seen from Figure 10.

Case Study	**Re**	**Cd**	**Θ (sep angle)**
Current Simulation	1.1e6	0.09	114.7
Experimental	1.1e6	0.12	118
Constantinescu's results	1.1e6	0.084	114

Table 1. Data on Drag and Separation angle; Experimental Results from Achenbach[42] and Schlichting[45]; DES results from Constantinescu[40]

Taneda reported the presence of a Ω shaped vortex ending in a pair of spiral points. He performed flow visualization and observed that the vortex sheet separating from the sphere rolls up into a Ω shaped structure to form a pair of strong stream-wise vortices. The Ω vortex plot obtained from the computations is shown in the Figure 11. Taneda also reported that the wake was not symmetrical but tilted. The tilting of the wakes causes sideward forces on the sphere. These sideward forces are non zero even in the mean and were observed in our study. The wake (as

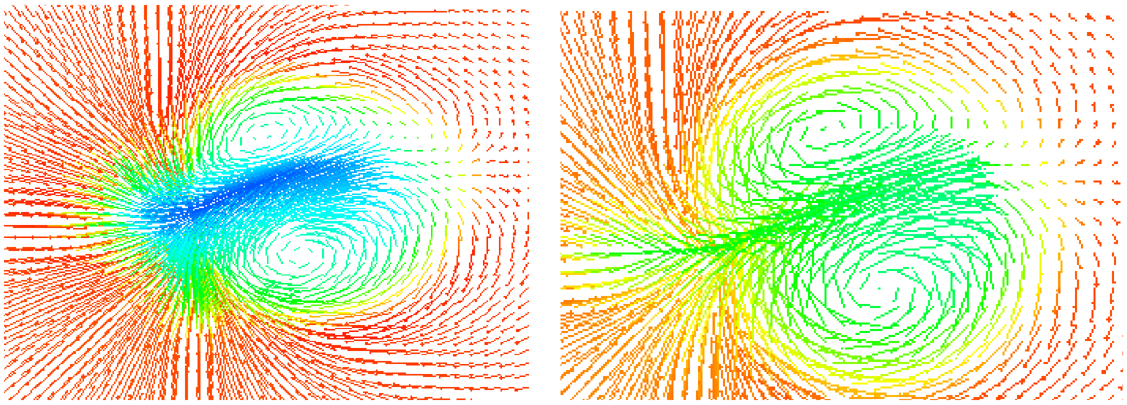
Figure 11. The velocity vectors denoting the Ω vortex plots at x/D = 0.65 and 1 at a Reynolds number of 1.1e6.

seen from the Figure 11) has the same orientations at x/D = 0.625 and x/D = 1.5. This results in non-zero lateral forces. The direction of this sideward force was random. Moreover the magnitude of the sideward force was the same order of magnitude as the drag force.

B. Inviscid Flow over a Moving Sphere

This case was selected to validate the moving grid flow solver. A sphere moves from right to left in quiescent air with a Mach number of 0.2. It is assumed that the flow is inviscid. If the reference frame is fixed on the moving sphere, the flow field should reach a steady state after the initial transients propagates out of the solution domain.

The computational grids at two different times are shown in Figure 12. The outer boundary of the computational grid is located 32 times the diameter away from the initial position of the sphere. The moving grid flow solver was first verified that the GCL was satisfied. Then it was used to solve the moving sphere problem. The pressure distributions at two different times are displayed in Figure 13. Note that initially a very high/low pressure region was created on the left/right side of the sphere due to the sudden motion. As time goes, the flow field becomes nearly "steady" for an observer stationed on the sphere. In fact, the pressure field created by the moving sphere after a long time is compared with that created by a free stream of Mach 0.2 over a stationary sphere in Figure 14. It is observed that the pressure fields are very similar.

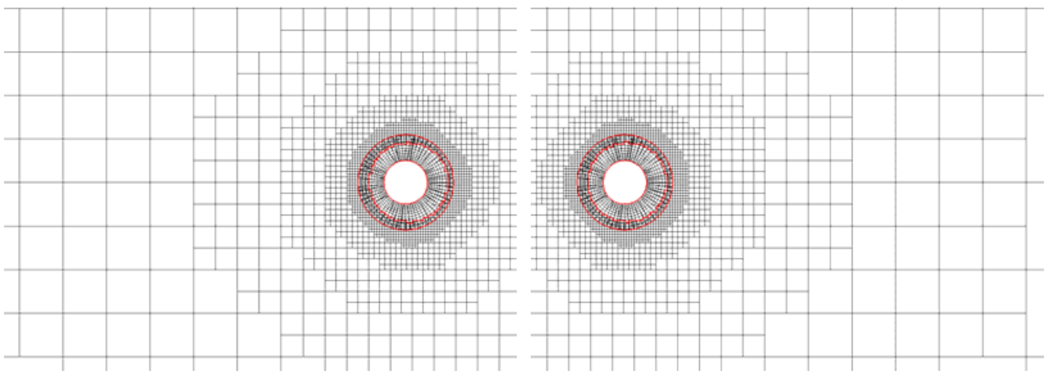
Figure 12. Computational grids at two different times for the moving sphere problem

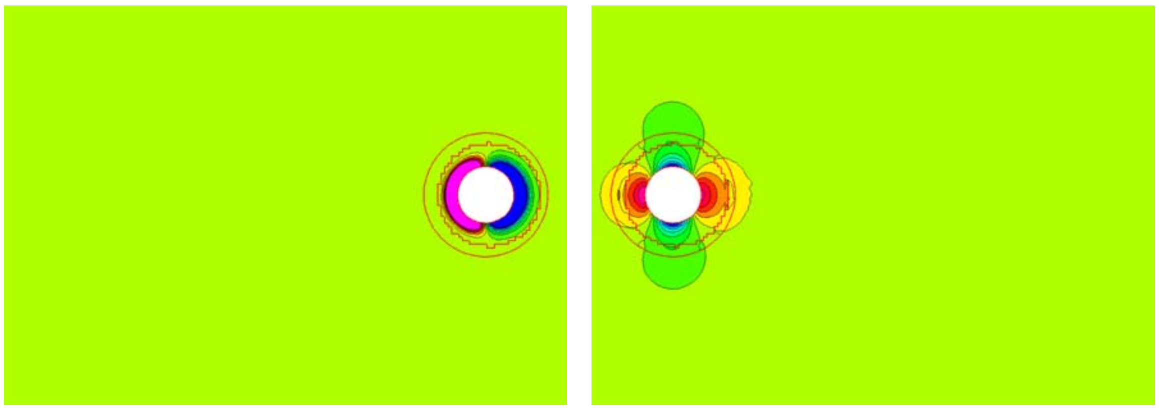

Figure 13. Pressure distributions at two different times for the moving sphere problem

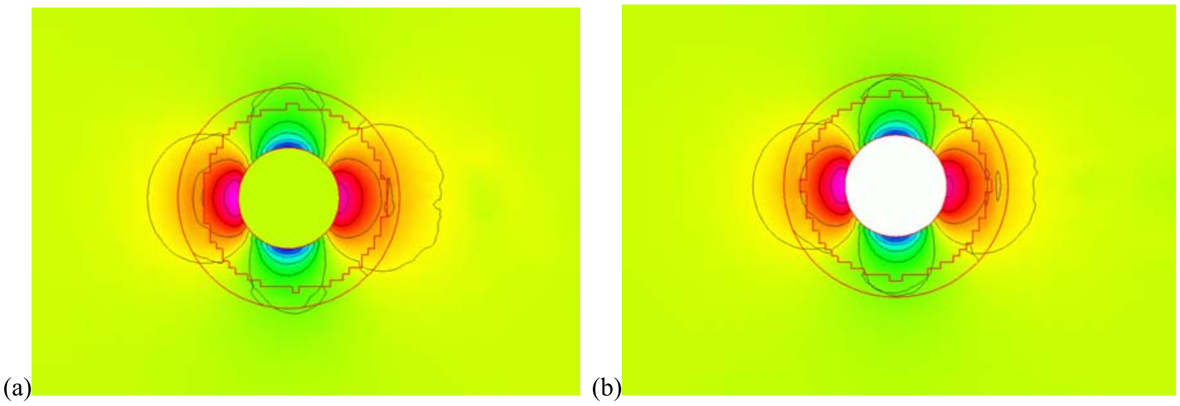

Figure 14. Comparison of pressure distributions for a moving sphere in quiescent air (a) and flow around a stationary sphere (b)

C. Wing-Pylon-Store Problem

As a final demonstration case, steady inviscid subsonic flow at Mach = 0.2 over a relatively complex geometry – wing-pylon-store was computed. This steady flow is simulated as a first step towards computing the store separation problem. The computational grid is shown in Figure 15. The Chimera holes generated in the Cartesian grid by the prism grids are shown in Figure 16. The pressure distribution is shown in Figure 17. Detailed comparison with moving body experimental data will be carried out in the future.

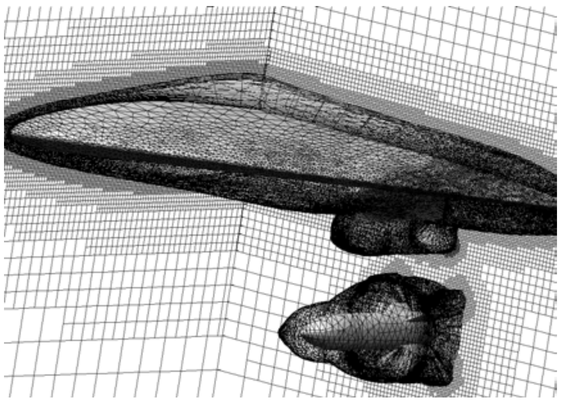

Figure 15. Overset adaptive Cartesian/prism grid for the wing-pylon-store case

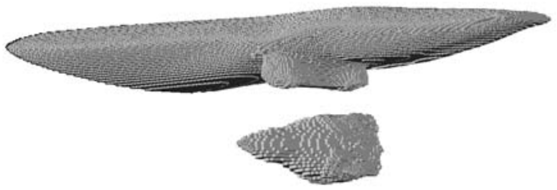

Figure 16. Hole boundary generated in the adaptive Cartesian grid

Conclusions

In the present study, an overset adaptive Cartesian/prism grid method has been developed to simulate moving boundary flow problems. The method combines the advantage of adaptive Cartesian/prism grid in geometry flexibility with that of Chimera approach in tackling moving boundary flow without grid remeshing. Advantages of the method include: 1. Cartesian cells are more efficient in filling space given a certain length scale than triangular/tetrahedral cells; 2. Searching operations can be performed very efficiently with the Octree data structure; 3. Solution based and geometry-based grid adaptations are straightforward to carry out.

The grid generator and overset flow solver are then tested for several steady and unsteady flow problems with stationary and moving bodies. The GCL has been satisfied with arbitrary grid motions. To test the accuracy of the overset interface algorithm, steady flows around a sphere at various Reynolds number were computed and compared with experimental data and other computations. There is very good agreement between the present computation and other data. More specifically,

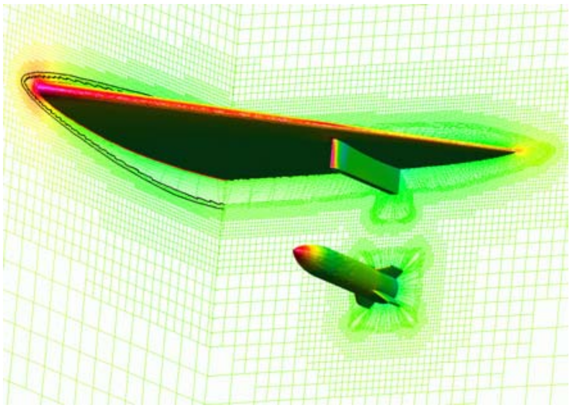
Figure 17. Computed pressure distribution for the wing-pylon-store case

1. A stationary vortex ring was formed behind the sphere at the Reynolds number of 118. For the case of Reynolds number of 800, a vortex pair loop was observed in the wake region, matching experimental observations.
2. At a Reynolds number of 800, c_f changes sign three times. This is due to initial separation of the boundary layer, followed by the reattachment of the boundary layer and the separation which occurs for the second time.
3. At a Reynolds number of 1.1e6, a Ω shaped vortex ending in a pair of spiral points was observed. These vortices are aligned in a direction which produces lateral forces which are non-zero in the mean. Once again the c_f increases after separation due to turbulent mixing of momentum.

The grid generator and flow solver have been coupled successfully to tackle a moving boundary flow problem with reasonable computational results. Further validation and demonstration with a high Reynolds store-separation problem is planned for the future.

Acknowledgements

The authors gratefully acknowledge the support from the Air Force Office of Scientific Research (Grant Number FA9550-04-1-0053), with Dr. Fariba Fahroo as the Technical Monitor.

References

1. Jameson A, Baker TJ and Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86-0103, 1986.
2. Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. J. Comput. Phys. 1987; 72:449-466.
3. Lohner R. and Parikh P. Generation of three-dimensional unstructured grids by the advancing front method. International J. Numerical Methods Fluids 1988; 8:1135-1149.
4. Anderson WK. A grid generation and flow solution method for the Euler equations on unstructured grids. J. of Comput. Phys. 1994; 110:23-38.
5. Venkatakrishnan V. A perspective on unstructured grid flow solvers. AIAA Paper 95-0667, Jan. 1995.
6. Pirzadeh S. Three-dimensional unstructured viscous grids by the advancing-layers method, *AIAA Journal*, 1996, Vol.34, No.1: 43-49.
7. Weatherill NP. Unstructured grids: procedures and applications. Handbook of grid generation, Edited by Thompson JF, Soni BK and Weatherill NP, CRC Press, 1998: Chapter 26.
8. Kallinderis Y, Khawaja A, and McMorris H. Hybrid prismatic/tetrahedral grid generation for complex geometries. AIAA Journal 1996; 34:291-298.
9. Coirier WJ and Jorgenson PCE. A mixed volume grid approach for the Euler and Navier-Stokes equations. AIAA Paper 96-0762, Jan. 1996.
10. Bayyuk SA, Powell KG and van Leer B. A simulation technique for 2D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrarily geometry," AIAA Paper 93–3391–CP, 1993.
11. Coirier WJ and Powell KG. Solution–adaptive Cartesian cell approach for viscous and inviscid flows. *AIAA J.* 1996; 34:938–945.
12. Aftosmis MJ, Berger MJ and Melton JE. Robust and efficient Cartesian mesh generation for component–based geometry. AIAA Paper No. 97–0196, 1997.

13. Karman SL. SPLITFLOW: a 3D unstructured Cartesian/prismatic grid CFD code for complete geometries. AIAA–95–0343, 1995.
14. Wang ZJ. A fast nested multi-grid viscous flow solver for adaptive Cartesian/quad grids, *International Journal for Numerical Methods in Fluids*, vol. 33, No. 5, pp. 657-680, 2000.
15. Wang Z.J., and Chen R.F., Anisotropic Solution-Adaptive Viscous Cartesian Grid Method for Turbulent Flow Simulation, *AIAA Journal*, Vol. 40, pp. 1969-1978, 2002.
16. Murman S, Aftosmis M and Berger M. Implicit approaches for moving boundaries in a 3-d Cartesian method, AIAA Paper 2003-1119.
17. Zhang, L.P. and Wang, Z.J. "A Block LU-SGS Implicit Dual Time-Stepping Algorithm for Hybrid Dynamic Meshes," *Computer & Fluids* Vol. 33, pp. 891-916, 2004.
18. Van Leer B. Towards the ultimate conservative difference scheme, *Journal of Computational Physics*, 1979, vol.32: 101.
19. Vassberg JC. A fast, implicit unstructured-mesh Euler method. AIAA Paper 92-2693, 1992.
20. Venkatakrishnan V and Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. AIAA Paper 95-1705, 1995.
21. Crumpton PI and Giles MB. Implicit time accurate solutions on unstructured dynamic grids. AIAA Paper 95-1671, 1995.
22. Luo H, Baum JD and Lohner R. A fast, matrix-free implicit method for compressible flows on unstructured grid. *Journal of Computational Physics*, 1998, vol.146: 664-690.
23. Chen RF and Wang ZJ, Fast, Block Lower-Upper Symmetric Gauss Seidel Scheme for Arbitrary Grids, *AIAA Journal*, 2000, vol. 38, no. 12: 2238-2245.
24. Van Leer B and Mulder WA, Relaxation methods for hyperbolic conservation laws, in Proceedings of the INRIA Workshop on Numerical Methods for the Euler Equations of Fluid Dynamics, Rocquencourt, France, December 1983.
25. Jameson A and Caughey DA. How many steps are required to solve the Euler equations of steady compressible flow: in search of a fast solution algorithm, 15th AIAA Computational Fluid Dynamics Conference, June 2001, Anaheim, CA.
26. Batina JT. Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis. AIAA Journal, 1991, vol.29, no.3: 327-333.
27. Luo H, Baum JD and Lohner R. An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grid. Computers & Fluids, 2001, vol.30: 137-159.
28. Benek, J.A., Steger, J.L., and Dougherty, F.C., "A Flexible Grid Embedding Technique with Application to the Euler Equations," AIAA Paper 83-1944, 1983.
29. Meakin, R.L., "On the Spatial and Temporal Accuracy of Overset Grid Methods for MovingBody Problems," AIAA Paper 94-1925, 1994.
30. Togashi, F., Nakahashi, K., Ito, Y., Iwamiya, T. and Shimbo, Y., "Flow Simulation of NAL Experimental Supersonic Airplane/Booster Separation Using Overset Unstructured Grids," *Computers & Fluids*, Vol. 30, Issue 6, July 2001, pp. 673-688.
31. Kallinderis, Y., and Ward, S., "Prismatic Grid Generation for 3-D Complex Geometries," *AIAA Journa*l, Vol 31, No. 10, 1993, pp. 1850-1856.
32. Pirzadeh, S., "Viscous Unstructured Three-Dimensional Grids by the Advancing-Layers Method," *AIAA Paper 94-0417,* Jan 1994
33. Parthasarathy, V., Kallinderis, Y., and Nakajima, K., "A Hybrid Adaptation Method and Directional Viscous Multigrid with Prismatic/Tetrahedral Meshes," *AIAA Paper 95-0670*, Jan 1995
34. Yagou, H., Ohtek, Y. and Belyaev, A., "Mesh Smoothing via Mean and Median Filtering Applied to Face Normals," Geometric Modelling and Processing: Theory and Applications (GMP'02), Jul 2002.
35. Spalart, P.R., Allmaras, S.R., 1994. "A one-equation turbulence modelfor aerodynamic flows," La Recherche Aerospatiale 1, 5–21.
36. Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes, *Journal of Computational Physics,* 1981, Vol. **43,** pp. 357-372.
37. Thomas PD. And Lombard CK. Geometric conservation law and it's application to flow computations on moving grids. *AIAA Journal,* 1979: 1030-1037.
38. Jameson, A. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings, AIAA Paper No 91-1596.
39. Taneda, S., 1978, "Visual Observations of the Flow Past a Sphere at Reynolds Numbers Between 10^4 and 10^6," *J. Fluid Mech*., 85, pp. 187-192.
40. Constantinescu, G.S., Pacheco, R. and Squires, K.D., 2002, "Detached-Eddy Simulation of Flow over a Sphere," *AIAA Paper 2002-0425*
41. Achenbach, E., 1972, "Experiments on the Flow Past Spheres at High Reynolds Numbers," *J. Fluid Mech*., **54**(3), pp. 565-575.
42. Achenbach, E., 1974, "Vortex shedding from the spheres", *J. Fluid Mech*., **62**, pp. 209-221.
43. Schlichting, H., 1979, *Boundary Layer Theory*, seventh edition, McGraw-Hill, New York.