



138
796
THS

1
2005
14130055

**LIBRARY
Michigan State
University**

This is to certify that the
thesis entitled

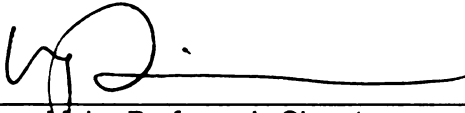
An Overset Adaptive Cartesian/Prism Grid Method for Moving
Boundary Problems

presented by

Ravishekar Kannan

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Mechanical Engineering



Major Professor's Signature

05-05-05

Date

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

DATE DUE	DATE DUE	DATE DUE

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

By

Ravishekar Kannan

A THESIS

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

MASTER OF SCIENCE

Department of Mechanical Engineering

2005

ABSTRACT

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

By

Ravishekar Kannan

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 re-meshing when dealing with moving boundary flow problems. Traditionally the overset grid approach named the chimera approach was mainly used for structured grids to simplify the grid generation process.

In this report, two particular unstructured grids are advocated for moving boundary flow simulation, i.e., the use of overset adaptive Cartesian/prism grids. An algorithm using the algebraic grid generation process was developed to construct semi-structured prism grids around solid walls. These body fitted 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. For a moving grid, it is necessary to readapt the Cartesian grid frequently. The overset adaptive Cartesian/prism grid method is tested for both steady and unsteady flow computations at a variety of Reynolds numbers. It is demonstrated that moving boundary flow computations can be carried out with minimum user interferences.

Copyright © by

RAVISHEKAR KANNAN

2005

To my Family

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Z. J. Wang, for introducing me to the research arena and for his constant support and his relentless motivation during this endeavor. It was due to him that I could surmount the various obstacles in my path. Without his perpetual encouragement, this project would have lagged by eons.

This work was funded by the Air Force Office of Scientific Research (Grant Number FA9550-04-1-0053). I am grateful to the Technical Monitor Dr. Fariba Fahroo for her support during the last two years.

I would like to thank my committee members Dr. Farhad Jaber and Dr. Andre Benard for their valuable time and for their constructive comments and suggestions. I would also like to thank the scholars in the CFD lab at MSU and my close friends for their invaluable help during the course of my research.

This research could not have been successful without the basic knowledge of heat transfer, fluids and computing. I am thankful to my undergraduate faculty in Indian Institute of Technology Madras (IITM), India and the graduate faculty in MSU for strengthening my foundation.

Last but definitely not the least, I thank my family for their help and continuous support throughout my career.

TABLE OF CONTENTS

LIST OF TABLES	VIII
LIST OF FIGURES	IX
NOMENCLATURE	XI
CHAPTER 1	
INTRODUCTION	1
Overview.....	1
Objectives of the present study.....	3
Organization of the thesis.....	4
CHAPTER 2	
ELEMENTS OF GRID GENERATION	6
The prism grid generation scheme.....	6
Obtaining the marching vectors.....	6
Marching step based on the Curvature.....	9
Mean filter smoothing algorithm.....	10
Checking for intersections.....	11
Checking for overlaps.....	13
The adaptive Cartesian grid generation scheme.....	15
Automated hole cutting and donor cell identification.....	15
CHAPTER 3	
NUMERICAL METHOD	20
Finite volume method for dynamic grids.....	20
Determining the viscous flux.....	21
The Geometric Conservation Law.....	23
Time integration algorithm.....	24
Boundary conditions.....	26
Limiting time step based on frequency of	
Grid adaptation.....	28
The Spalart-Allmaras model.....	29
CHAPTER 4	
RESULTS AND DISCUSSIONS	31
Viscous flow over a stationary sphere.....	31
Flow at low Reynolds number.....	32

Flow at high Reynolds number.....	35
Inviscid flow over a moving sphere.....	39
Viscous flow over a moving sphere.....	41
Prolate spheroid undergoing a pitch-up maneuver in a laminar fluid.....	43
Wing-Pylon-Store problem.....	46

CHAPTER 5

CONCLUSIONS.....	48
Plans for the future.....	49

BIBLIOGRAPHY.....	51
--------------------------	-----------

LIST OF TABLES

4.1	Data on Drag and Separation angle; Experimental Results from Achenbach ⁴² and Schlichting ⁴⁵ ; DES results from Constantinescu ⁴⁰	37
-----	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----

LIST OF FIGURES

2.1	Two Dimensional Concave and Convex models.....	9
2.2	An example of a prism with no intersections.....	12
2.3	An example of an overlap.....	13
2.4	Schematic of Hole-Cutting.....	16
2.5	Example of an overset Cartesian/Prism grid for the store configuration.....	18
2.6	Example of an overset Cartesian/Prism grid for the missile configuration.....	18
2.7	Example of an overset Cartesian/Prism grid for the F-16 aircraft configuration.....	19
2.8	Example of an overset Cartesian/Prism grid for the wing-pylon-store configuration.....	19
3.1	Regular Cartesian grid stencils for gradient computation at face $(i+1)/2$	21
3.2	Schematic of viscous flux computation at a face.....	22
4.1	Coarse and Fine Grids Used for Flow over a Sphere at $Re = 118$	32
4.2	Velocity vector plot showing the separation region at $Re = 118$	33
4.3	Entropy distribution depicting the vortex pair seen at $x/D = 2$ for $Re = 800$	34
4.4	Static Pressure Coefficient at Two Different Reynolds Numbers.....	34
4.5	Skin Friction Coefficients at Two Different Reynolds Numbers.....	34
4.6	Comparison of Static Pressure Coefficients at Reynolds Number = $1.1e6$	36
4.7	Comparison of Skin Friction Coefficient at Reynolds Number = $1.1e6$	37
4.8	The velocity vectors denoting the Ω vortex plots at $x/D = 0.65$ and 1 at a Reynolds number of $1.1e6$	38

4.9	Computational grids at two different times for the moving sphere problem.....	39
4.10	Pressure distributions at two different times for the moving sphere problem.....	40
4.11	Comparison of pressure distributions for a moving sphere in quiescent air (a) and flow around a stationary sphere (b).....	40
4.12	Static pressure Coefficient obtained for inviscid flow over a sphere using the moving boundary method.....	41
4.13	Drag force of a sphere in inviscid flow (Obtained using moving boundary method).....	41
4.14	Static pressure Coefficient obtained for laminar flow over a sphere ($Re = 118$).....	42
4.15	Skin Friction Coefficient of a sphere in laminar flow ($Re = 118$).....	42
4.16	C_p distribution at $x/L = 0.11$ at 10 and 20 Degrees angle of attack.....	44
4.17	C_p distribution at $x/L = 0.43$ at 10 and 20 Degrees angle of attack.....	44
4.18	C_f distribution at $x/L = 0.11$ at 10 and 20 Degrees angle of attack.....	45
4.19	C_f distribution at $x/L = 0.43$ at 10 and 20 Degrees angle of attack.....	46
4.20	Hole boundary generated in the adaptive Cartesian grid.....	47
4.21	Overset adaptive Cartesian/prism grid for the wing-pylon-store case.....	47
4.22	Computed pressure distribution for the wing-pylon-store case.....	47

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
M_i	=	Marching vector at a node i.
θ_{\max}	=	Maximum angle between the marching vector and the face normals of its node-manifold
m	=	Area weighted normal vector of a triangle
Q	=	Vector of conserved variables
r	=	Position vector
r_j^c	=	Position vector of the centroid of triangle j
Re	=	Reynolds number
v_g	=	Grid velocity
v_{gn}	=	Surface normal grid velocity component
V_i	=	Volume of control volume i
n_j	=	Surface normal of the triangle j

INTRODUCTION

A. Overview

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-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 methods¹⁰⁻¹⁷ 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 smoothing¹ have also been employed in this context. However, for large-scale problems and especially for the solution of viscous turbulent flows, implicit schemes¹⁹⁻²⁵ 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 re-meshing can be applied whenever the grid becomes too skewed. With local re-meshing, solution interpolations from the old to the new grid become necessary. The hybrid approach of combining grid deformation with grid local re-meshing 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 re-meshing 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 report, 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 re-meshing. 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. It is well known that it takes 2 right triangles to fill a square(i.e. in 2D) and 12 tetrahedra (though not regular) to fill a cube

2. Searching operations can be performed very efficiently with the Octree data structure. A brute force searching operation consumes time that is of the order of n^2 . Using a clever implementation of the Octree based data structure, the time consumed can be made of the order of $n \log n$.

3. Solution based and geometry-based grid adaptations are straightforward to carry out. It is well known that the solution-based adaptation using the magnitude of the gradients can be carried out easily for a Cartesian grid

B. Objectives of the Present Study

B.1 To develop a robust prism grid generator.

This prism grid generator must be capable of generating good body fitted grids for most real life geometries in an optimal fashion. These body-fitted prism grids are meant to resolve viscous boundary layers.

B.2 To generate a stationary background adaptive Cartesian grid.

The adaptive Cartesian grid is generated to cover the outer domain and to serve as the background grid for bridging the “gaps” between the prism grids.

B.3 An algorithm to generate holes in the adaptive Cartesian grid to facilitate the data communication.

The prism grids are used to generate holes in the adaptive Cartesian grid for data communication. If the bodies move, the prism grids move with the bodies, while the Cartesian grid remains stationary.

B.4 Automate the creation of new holes and identification of new donor cells.

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.

C. Organization Of The Thesis

The report is organized as follows. In chapter 2, the overset adaptive Cartesian/prism grid generation approach will be presented, together with illustration examples.

Chapter 3 gives a brief overview about the solver, boundary conditions and the closure models for turbulence equations. In chapter 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 chapter 5.

ELEMENTS OF GRID GENERATION

A. THE PRISM GRID GENERATION SCHEME

Since we do not address geometry modeling issues in this report, it is assumed that watertight 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 is outlined next.

A.1 Obtaining the Marching Vectors

This is the quintessential aspect of prism grid generation. Kallinderis³¹ 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° 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 surfaces and causing the flow solver to deliver unrealistic

results. 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. For the above scenario, the maximum of the angles between the marching vector at a node and the face normals of its node-manifold is obtained. This angle, θ_{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. According to this, the marching vector \mathbf{M}_i at the node i is given by

$$\mathbf{M}_i = \frac{\sum \theta_j \mathbf{n}_j}{\sum \theta_j} \quad (2.1)$$

Where θ_j is the angle subtended by the triangle j at the node i , \mathbf{n}_j is the surface normal of the triangle j and the summation is from 1 to the number of triangles containing the node i .

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 that 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 $\{S\}$ which make a small angle δ (about 1°) 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 θ_{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. This was done to decrease the possibility of intersections.

Accordingly,

$$\mathbf{M}_i = \alpha \mathbf{M}_i + \frac{\sum_j (1-\alpha) \frac{\mathbf{M}_j}{d_{ij}}}{\sum_j \frac{1}{d_{ij}}}, \quad (2.2)$$

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 . In other words, the orientation of the marching vectors which make a large angle θ_{max} i.e. the critical angles are affected to a minimal extent.

A.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 manifold.

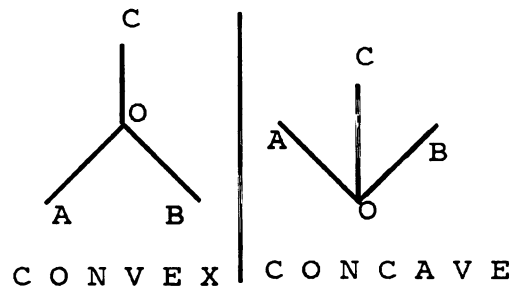


Figure 2.1 Two Dimensional Concave and Convex models

A new scheme was devised to estimate the surface curvature. For better understanding, let us start with a 2 dimensional model. Figure 2.1 depicts the marching vectors for a two dimensional case. OC is the un-smoothed marching vector in figure 2.1. For the convex case, the angle between OC and OA is greater than 90° .

Similarly the angle between OC and OB is greater than 90° . For the concave case, the angle between OC and OA is less than 90° . Similarly the angle between OC and OB is less than 90° . 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° , the surface is convex. If each of the above angles is lesser than 90° , 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° , the surface is treated as a convex surface else it is treated as a concave surface.

A.3 Mean Filter Smoothing Algorithm

The algorithm discussed till now is not totally perfect. As per the algorithm, the nodes in the concave region get closer with advancing layers. This results in the triangles becoming increasingly obtuse with advancing layers and hence causing intersections between the marching vectors. In order to circumvent the above, smoothing of the nodes in the new layer is to be done so as to even the spacing between the nodes. The most obvious choice for a smoothing operator was the Laplacian Smoothing operator. This smoothing operator did not work out very well especially at concave regions. After doing some literature survey, a new type of smoothing (Called Mean filtering) which works well in the concave regime was obtained. This filter³⁴ 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:

$$\mathbf{m}_i = \frac{\sum_j A_j \mathbf{n}_j}{\sum_j A_j}. \quad (2.3)$$

- b. Normalize the averaged normals;
- c. For each mesh vertex, perform the following vertex updating procedure:

$$\mathbf{r}_{new} = \mathbf{r}_{old} + \frac{\sum_j A_j (\mathbf{m}_j \cdot \mathbf{r}_j^c) \mathbf{m}_j}{\sum_j A_j}, \quad (2.4)$$

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.

A.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 areas and 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.

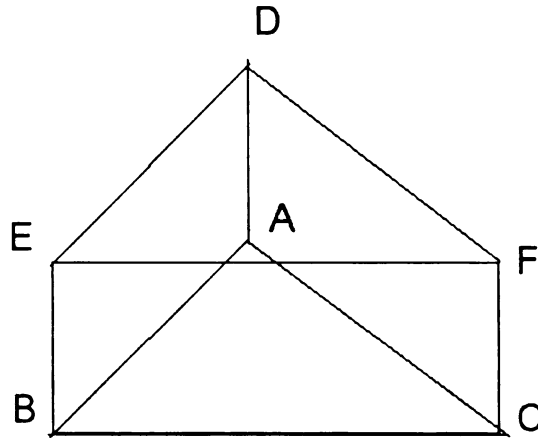


Figure 2.2 An example of a prism with no intersections

Figure 2.2 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 can occur when either of the below can occur

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 EDDB

In spite of performing the mean filter smoothing operation, intersections can persist. In this scenario, the marching vectors which cause intersections are tweaked locally so as to remove the intersections. Correcting intersections locally is exceedingly time consuming. However the possibility of intersections occurring is uncommon.

A.5 Checking for Overlaps

Overlaps can occur in small gaps and in multi-body configurations. A novel scheme was developed to check for overlaps. Consider any 2 triangles T1 and T2 in the outermost prism layer. If T1 intersects T2 or if T2 intersects T1, it can be concluded that an overlap has occurred. This is shown in figure 2.3.

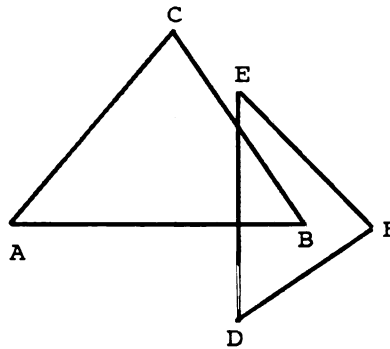


Figure 2.3 An example of an overlap

A cursory approach for detecting overlaps is to check every triangle in the outermost layer with all the possible triangles in that layer. The detection of these overlaps can be exceedingly time consuming if done by the above detection mechanism. It could be intuitively figured out that this brute force detection strategy consumes time which is

of the order of n^2 where n is the number of triangles.

An Alternating Digital Tree (ADT) based search was employed. The coordinates of each node in the outermost prism layer is fed into the ADT subroutine. ADT stores the nodes based on their coordinates. Thus for each triangle node i , the nodes closest to it can be determined easily by constructing a bounding box and searching the nodes which lie inside that bounding box.

Extending the above idea, a bounding box is constructed for each triangle and all the other triangles intersecting this bounding box are determined. Thus the check for overlap for this triangle is done with the dozen or lesser triangles intersecting the bounding box. In contrast the brute force method requires checking with all the other triangles. ADT checking mechanism reduces the run time to be of the order of $n \log(n)$.

Currently the correction is done on a local scale. However it is possible that the correction done locally be totally inadequate. In these circumstances, the correction measures needs to be applied from the first layer. This global corrective mechanism is exceedingly time consuming and was avoided as much as possible.

B. THE ADAPTIVE CARTESIAN GRID GENERATION SCHEME

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 local discretization error.

B.1 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. A schematic of the hole cutting operation is shown in figure 2.4.

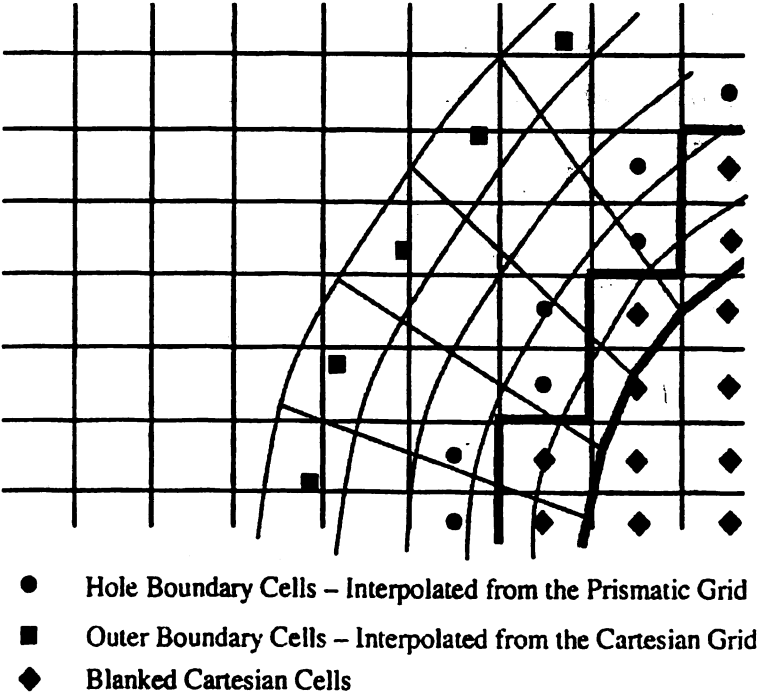


Figure 2.4 Schematic of Hole-Cutting

The efficiency of the hole cutting algorithm is critical since many steps of the hole-cutting operation is performed in the moving boundary flow simulation as the prism grids move in the flow field. To achieve the maximum efficiency, search trees were used extensively. One is the Octtree for the adaptive Cartesian grid and the other is the Alternating Digital Tree (ADT) for bounding the boxes of prism cells. The use

of Octree to speed up the search operations is another significant advantage of using the adaptive Cartesian grid for moving boundary problems. 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.

After each time step/iteration, the field variables at the outer boundary cells are interpolated from the Cartesian grid, while solutions at the hole boundary are interpolated from the prismatic grids.

The Cartesian-Prism Overset grid generation algorithm was tested for several real life geometries like a store configuration, missile configuration and a F16 aircraft. These are shown in the figures 2.5, 2.6 and 2.7. The Cartesian-Prism Overset grid was also generated on a wing-pylon-store multi body configuration as is shown in figure 2.8.

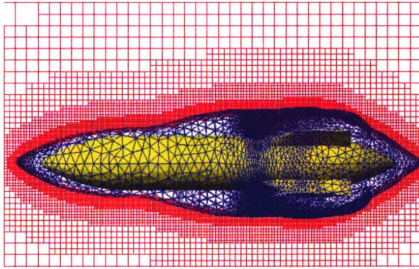


Figure 2.5 Example of an overset Cartesian/Prism grid for the store configuration

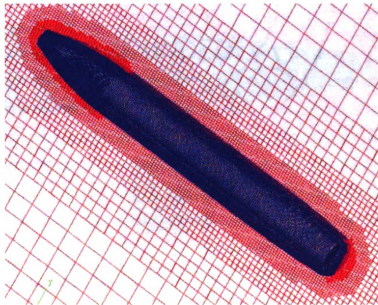


Figure 2.6 Example of an overset Cartesian/Prism grid for the missile configuration

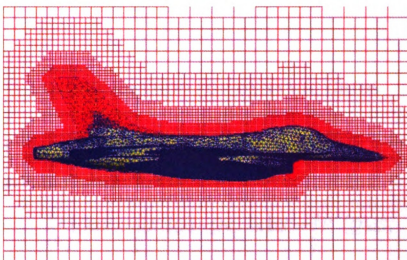


Figure 2.7 Example of an overset Cartesian/Prism grid for the F-16 aircraft configuration

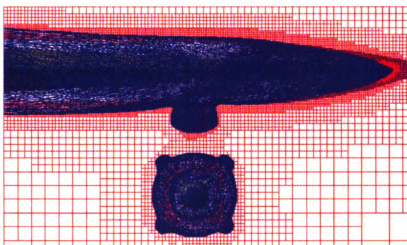


Figure 2.8 Example of an overset Cartesian/Prism grid for the wing-pylon-store configuration

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

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

where S is the surface surrounding the control volume V , \mathbf{n} is the out-going unit normal of S , \mathbf{v}_g is the velocity of S , and \mathbf{Q} is the vector of conserved variables, F^i is the inviscid and 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

$$\frac{\partial}{\partial t} (Q V_i) + \sum_f (F^i(Q) - Q \mathbf{v}_{gn})_f dS_f = \sum_f F_f^v(Q) dS_f, \quad (3.2)$$

where the summation index f represents all the faces surrounding control volume V_i , and $\mathbf{v}_{gn} = \mathbf{v}_g \cdot \mathbf{n}$. The inviscid flux is calculated using Roe's approximate Riemann solver³⁶ with reconstructed state variables at both sides of a face. A least square linear reconstruction scheme of the primitive variables is used.

A.1 Determining the viscous flux

The viscous flux at a face can be expressed as a function of the flow variables and of their gradients i.e.

$$\mathbf{F}_{v,f} = f_v(q_f, \nabla q_f) \quad (3.3)$$

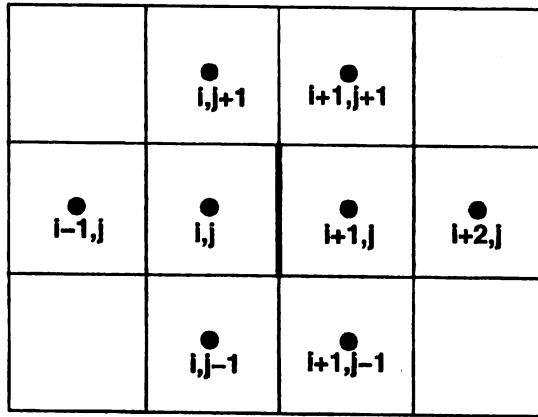


Figure 3.1 Regular Cartesian grid stencils for gradient computation at face $(i+1)/2$

Where Q_f is the mean of $Q_{f,L}$ and $Q_{f,R}$. If a simple average of ∇q_L and ∇q_R is used as the gradient at the face, the variables at the two cells sharing the face contribute less to the gradient than data further away from the face. In the case of a regular Cartesian grid shown in Fig 3.1, the x gradient at the face $i + 1/2$ resulting from an average of ∇q_L and ∇q_R is

$$\frac{\partial q_{i+1/2,j}}{\partial x} = \frac{1}{4} \frac{(q_{i+1,j} - q_{i,j})}{\Delta x} + \frac{3}{4} \frac{(q_{i+2,j} - q_{i-1,j})}{3\Delta x} \quad (3.4)$$

The terms in the brackets are the approximations for dq/dx . Thus only one quarter of the final derivative is contributed by the data closest to the face. The other three quarters are contributed by far-flung data which can obviously cause numerical stiffness. To overcome this drawback, the following viscous reconstruction^{14, 15} was used.

Let \mathbf{m} be the unit normal in the face tangential direction and \mathbf{l} be the unit vector connecting the left cell and the right cell of a face as shown in Fig 3.2. The derivative of a variable in \mathbf{m} direction is obtained from the cell wise inviscid reconstruction, i.e.

$$\frac{dq}{dm} = \frac{(\nabla q_L \cdot \mathbf{m} + \nabla q_R \cdot \mathbf{m})}{2} \quad (3.5)$$

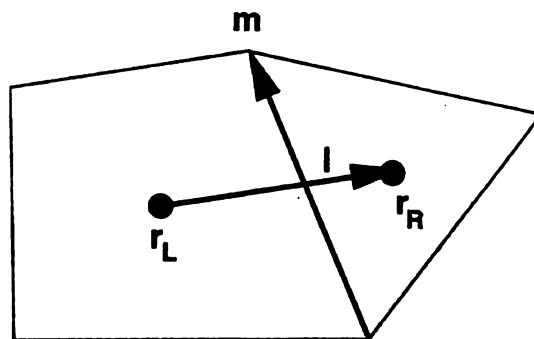


Figure 3.2 Schematic of viscous flux computation at a face

Where ∇q_L and ∇q_R are the gradients in the left and right cells of the face from inviscid reconstruction. Thus dq/dl is now

$$\frac{dq}{dl} = \frac{q_R - q_L}{|r_R - r_L|} \quad (3.6)$$

Thus $\nabla q = (q_x, q_y)$ is obtained by solving the equations

$$\begin{aligned} q_x \cdot l_x + q_y \cdot l_y &= \frac{dq}{dl} \\ q_x \cdot m_x + q_y \cdot m_y &= \frac{dq}{dm} \end{aligned} \quad (3.7)$$

The gradient in a Cartesian grid calculated by the above scheme gives:

$$\begin{aligned} \frac{\partial q_{i+1/2,j}}{\partial x} &= \frac{(q_{i+1,j} - q_{i,j})}{\Delta x} \\ \frac{\partial q_{i+1/2,j}}{\partial y} &= \frac{(q_{i+1,j+1} + q_{i,j+1} - q_{i+1,j-1} - q_{i,j-1})}{4\Delta y} \end{aligned} \quad (3.8)$$

which is both accurate and correct.

A.2 The Geometric Conservation Law

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

$$\frac{\partial V_i}{\partial t} = \sum_f v_{gn} dS_f \quad (3.9)$$

This is the so-called Geometric Conservative Law³⁷ in its semi-discretized form.

Assume that the grid velocity is computed at time level $n+1/2$. Then we can use the following time discretization to achieve second-order accuracy

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} = \sum_f v_{gn} dS_f \quad (3.10)$$

Instead of having the grid velocity v_{gn} satisfy Equation (3.10), 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.

$$V_i^{n+1} - V_i^n = \sum_f \Delta V_f \quad (3.11)$$

where ΔV_f represents the volume swept by face f . Comparing Equations (3.10) and (3.11), we arrive at the following equation:

$$\Delta V_f = \Delta t v_{gn} dS_f \quad \text{or} \quad v_{gn} = \frac{\Delta V_f}{\Delta t dS_f} \quad (3.12)$$

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:

$$\frac{\partial(QV)_i}{\partial t} + R_i(Q) = 0, \quad (3.13)$$

where R_i is the residual given by

$$R_i(Q) = \sum_f \left(F^i(Q) - Q v_{gn} - F^v(Q) \right)_f dS_f. \quad (3.14)$$

The easiest time integration algorithm for equation (3.14) 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

$$\frac{Q_i^{n+1} V_i^{n+1} - Q_i^n V_i^n}{\Delta t} + (1 - \theta) R_i(Q^{n+1}) + \theta R_i(Q^n) = 0. \quad (3.15)$$

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 (3.15) 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

$$\frac{\partial(QV)_i}{\partial \tau} + R_i^*(Q) = 0, \quad (3.16)$$

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

$$R_i^*(Q) = \frac{Q_i V_i^{n+1} - Q_i^n V_i^n}{\Delta t} + (1 - \theta) R_i(Q) + \theta R_i(Q^n) \quad (3.17)$$

Obviously, Equation (3.17) 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) approach¹⁷ 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:

$$u_{ghost} = u - 2n_x v_n \quad v_{ghost} = v - 2n_y v_n, \quad (3.18)$$

Where v_n is the normal velocity given by

$$v_n = un_x + vn_y. \quad (3.19)$$

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

$$v_n = un_x + vn_y - v_{gn}. \quad (3.20)$$

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

$$u_{ghost} = -u + 2n_x v_{gn} \quad v_{ghost} = -v + 2n_y v_{gn}. \quad (3.21)$$

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.

D. LIMITING TIMESTEP BASED ON GRID ADAPTATION FREQUENCY

It is well known that the time step is always limited by the CFL criterion. In this section, we get a first order estimate for the maximum time step possible based on the frequency with which the Cartesian Grid is adapted.

It was explained earlier that the Cartesian grid is adapted once in every n time steps where n varies from 8 to 20. During these n time steps, the position of the Cartesian Donor remains constant. This implies that the quality of interpolation degrades from time step 1 to time step n . The interpolation carried out is a linear interpolation. If the prism traverses to such an extent that the prism recipients are outside the Cartesian donor cells, a bad quality interpolation is obtained. In other words, a ceiling for the time step is

$$\Delta t \leq \frac{\delta}{nV} \quad (3.22)$$

Where δ is the average thickness of the inner boundary Cartesian cells. This style of estimation is only first order accurate. In fact for some problems, a time step which is about a third of the ceiling needs to be used to obtain an accurate solution. In addition, the above scheme needs to be modified for bodies which undergo a rotation along with the translation like a pitching prolate.

E. THE SPALART-ALLMARAS MODEL

To simulate flow turbulence, a RANS Spalart-Allmaras (S-A) model approach was employed. The S-A one-equation model³⁵ solves a single partial differential equation for a variable $\tilde{\nu}$ which is related to the turbulent viscosity. The differential equation is derived by using empiricism and arguments of dimensional analysis, Galilean invariance and selected dependence on the molecular viscosity. The model includes a wall destruction term that reduces the turbulent viscosity in the log layer and laminar sublayer. The equation can be written in the following form

$$\frac{D\tilde{\nu}}{Dt} = c_{b1}\tilde{S}\tilde{\nu} - c_{w1}f_w\left[\frac{\tilde{\nu}}{d}\right]^2 + \frac{1}{\sigma}\left[\nabla \cdot ((\nu + \tilde{\nu})\nabla \tilde{\nu}) + c_{b2}(\nabla \tilde{\nu})^2\right] \quad (3.23)$$

The turbulent viscosity is determined via,

$$\nu_t = \tilde{\nu}f_{v1}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi \equiv \frac{\tilde{\nu}}{\nu} \quad (3.24)$$

Where ν is the molecular viscosity. Using S to denote the magnitude of the vorticity, the modified vorticity is defined as

$$\tilde{S} \equiv S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \quad (3.25)$$

Where d is the distance to the closest wall. The wall destruction function is defined as

$$f_w = g \left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right]^{1/6}, \quad g = r + c_{w2}(r^6 - r), \quad r \equiv \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \quad (3.26)$$

The closure coefficients are given by:

$$c_{b1} = 0.1355$$

$$\sigma = 2/3$$

$$c_{b2} = 0.622$$

$$k = 0.41$$

$$c_{w1} = \frac{c_{b1}}{k * k} + \frac{(1 + c_{b2})}{\sigma}$$

$$c_{w2} = 0.3$$

$$c_{w3} = 2$$

$$c_{v1} = 7.1$$

Once again, the ADT comes in handy. The distance to the closest wall is calculated

using the ADT data structure. A brute force method would be highly inefficient.

RESULTS AND DISCUSSIONS

In this chapter, the results obtained by the overset adaptive Cartesian/prism grid method are for both stationary and moving boundary flow problems are discussed and compared with existing numerical and experimental data. The following 5 cases are presented. Images in this thesis are presented in color.

A. VISCOUS FLOW OVER A STATIONARY SPHERE

In this section, we present the computational results of flow over spheres 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 1.1×10^6 . 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 than 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.

A.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 4.1.

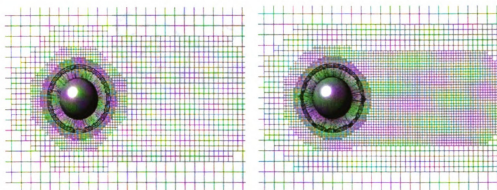
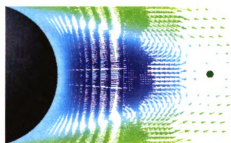


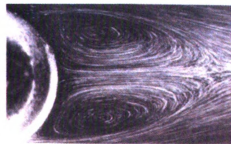
Figure 4.1 Coarse and Fine Grids Used for Flow over a Sphere at $Re = 118$

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 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 4.4 and 4.5 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 picture in Figure 4.2. 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 4.3, which displays the entropy distribution on plane $x = 2D$. The vortex pair was also observed experimentally observed.⁴³



(a) Computation



(b) Experiment

Figure 4.2 Velocity vector plot showing the separation region at $Re = 118$

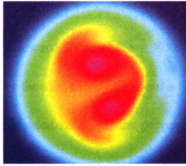


Figure 4.3 Entropy distribution depicting the vortex pair seen at $x/D = 2$ for $Re = 800$

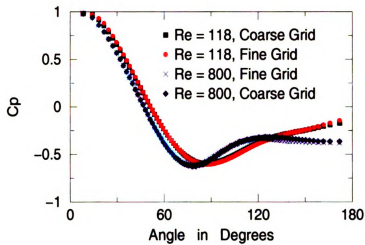


Figure 4.4 Static Pressure Coefficient at Two Different Reynolds Numbers

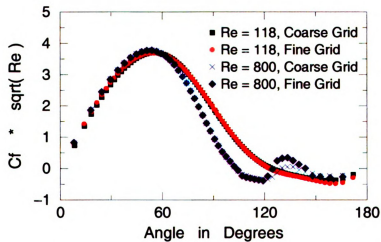


Figure 4.5 Skin Friction Coefficients at Two Different Reynolds Numbers

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 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 4.5 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 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 $Re = 118$.

A.2 Flow at High Reynolds Number

Next turbulent flow over a sphere at $Re = 1.1 \times 10^6$ was computed with the S-A

turbulence model. The computed c_p profiles on the coarse (1.5×10^6 cells) and fine grids (2.1×10^6 cells) are compared with experimental data by Achenbach⁴² and simulation results by Constantinescu⁴⁰ in Figure 4.6. Note that the agreement in c_p is quite good between the present computation and other experimental and computational data.

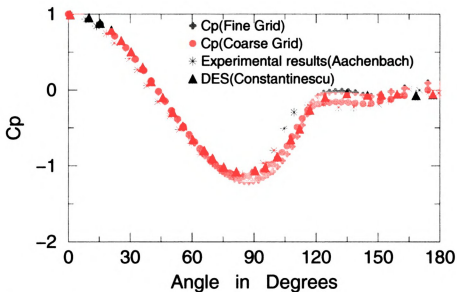


Figure 4.6 Comparison of Static Pressure Coefficients at Reynolds Number = 1.1×10^6

The simulation over the sphere at a Reynolds number of 1.1×10^6 yielded a drag coefficient C_d of 0.09 and 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 4.6. 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 4.1. 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 C_d obtained from the current simulation was in good agreement with the results of Constantinescu. The c_f distribution did not match other data well.

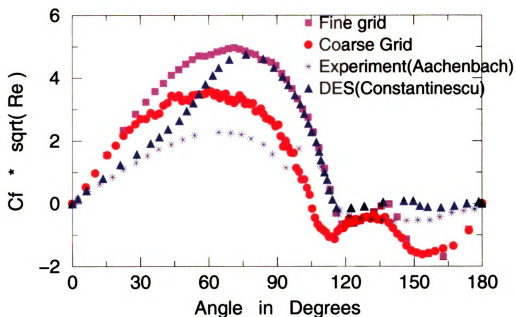


Figure 4.7 Comparison of Skin Friction Coefficient at Reynolds Number = 1.1e6

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 4.7.

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

Table 4.1 Data on Drag and Separation angle; Experimental Results from Achenbach⁴² and Schlichting⁴⁵; DES results from Constantinescu⁴⁰

Taneda³⁹ reported the presence of a Ω shaped vortex ending in a pair of spiral points. He performed visualization experiments 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 4.8.

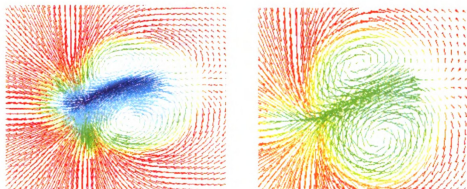


Figure 4.8 The velocity vectors denoting the Ω vortex plots at $x/D = 0.65$ and 1 at a Reynolds number of $1.1e6$.

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 seen from the Figure 4.8) 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 propagate out of the solution domain.

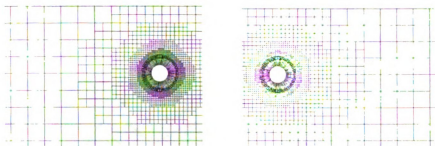


Figure 4.9 Computational grids at two different times for the moving sphere problem

The computational grids at two different times are shown in Figure 4.9. 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 4.10. 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 on, 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 4.11. It is observed that the pressure fields are very similar.

The steady state C_p distribution is shown in the figure 4.12. This C_p distribution agrees with the distribution predicted by potential flow. The Drag force as a function of time is plotted in the figure 4.13. As expected, the drag force goes to zero after a long time.

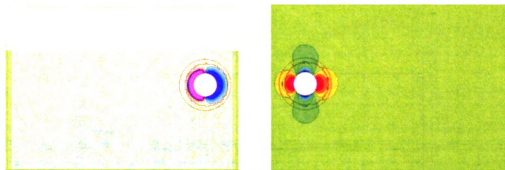


Figure 4.10 Pressure distributions at two different times for the moving sphere problem

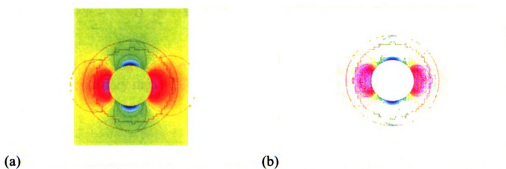


Figure 4.11 Comparison of pressure distributions for a moving sphere in quiescent air (a) and flow around a stationary sphere (b)

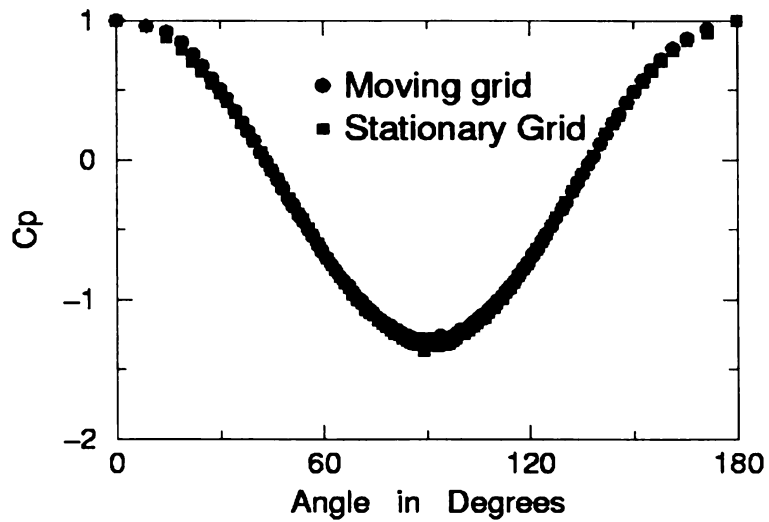


Figure 4.12 Static pressure Coefficient obtained for inviscid flow over a sphere.

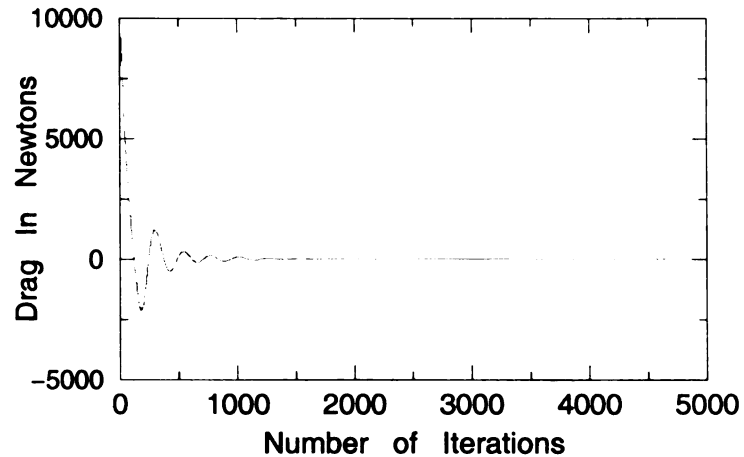


Figure 4.13 Drag force of a sphere in inviscid flow (Obtained using moving boundary method)

C. VISCOUS FLOW OVER A MOVING SPHERE

The moving grid solver was tested for a sphere moving in a viscous but laminar fluid.

The free stream pressure, temperature and density were prescribed the values used in

Case A. The Reynolds number based on the sphere speed was 118. The steady C_p and

Cf distributions were obtained and were compared with the results obtained from Case A. The match was very good as is seen from the figures 4.14 and 4.15. The drag obtained was in good accord with that of the stationary sphere case (variation was around 3%).

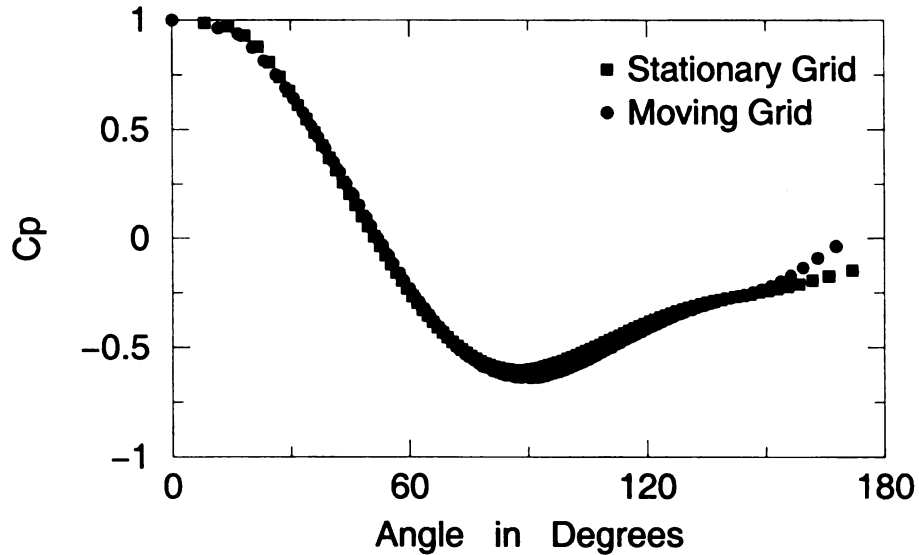


Figure 4.14 Cp obtained for laminar flow over a sphere (Re = 118)

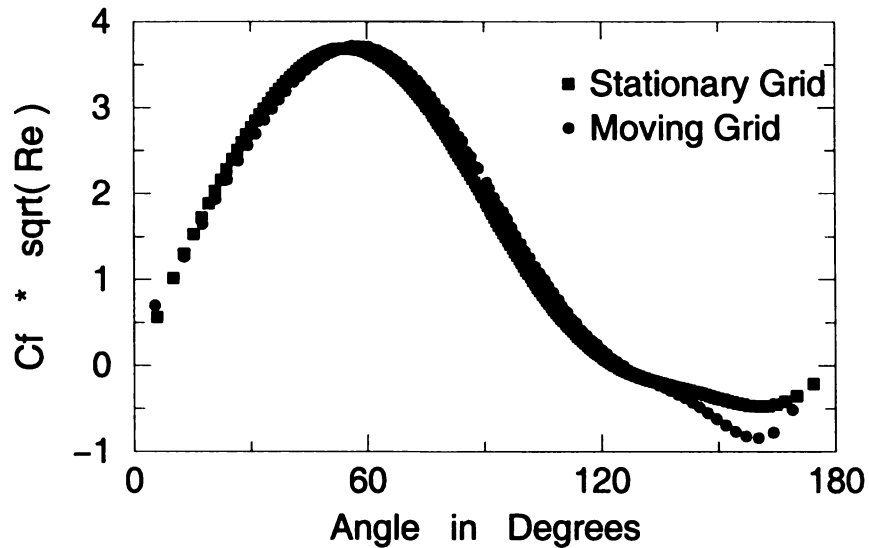


Figure 4.15 Cf of a sphere in laminar flow (Re = 118)

D. PROLATE SPHEROID UNDERGOING PITCH-UP MANEUVER IN A LAMINAR FLUID

This case was selected to demonstrate the ability of the moving grid solver to handle a rotational degree of freedom (DOF). The prolate was a 6:1:1 ellipsoid. The non-dimensional angular velocity was 0.047. In addition, the centroid of the spheroid performs a translation with 45.7 m/s. The Reynolds number based on the length of the prolate was 100. The simulations were performed till the prolate was rotated by 30°

The motivation behind this case was to simulate a submarine entering a turning maneuver. The experiments and the simulations (using the traditional methods) performed for this are for turbulent flow (i.e. $Re > 4.2 * 10^6$). The current simulations were performed using a single processor. Thus it is virtually impossible to get a solution in a 'finite' time using a RANS/LES based turbulence model. Hence simulating the motion in a laminar flow was a good alternative.

The c_p and c_f distributions were obtained and are shown in a reference frame attached to the spheroid and aligned with the body axes. Figures 4.16 and 4.17 show the c_p distributions at x/L of 0.11 and 0.43 plotted as a function of azimuthal angle ϕ ($\phi = 0$ corresponds to the windward symmetry plane). The match between the fine (380000 cells) and the coarse (260000 cells) grids is not up to the mark at 10° angle of attack. The transients are still in action at an angle of attack of 10°. Hence an extremely fine grid is necessary to resolve the high pressure gradients arising from these transients.

As expected, the c_p at $\phi=0$ increases with the angle of attack.

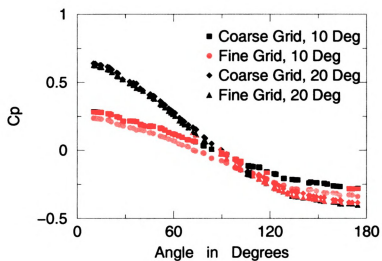


Figure 4.16 C_p distribution at $x/L = 0.11$ at 10 and 20 Degrees angle of attack

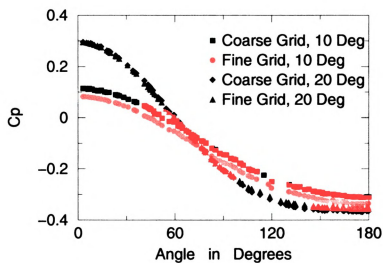


Figure 4.17 C_p distribution at $x/L = 0.43$ at 10 and 20 Degrees angle of attack

The c_f distributions are plotted in the figure 4.18 and 4.19. As the c_f does not change sign, it can be concluded that separation has not occurred at the above-mentioned locations. In addition, the c_f at $x/L = 0.11$ is much higher than the c_f at $x/L = 0.43$. Hence the velocity gradients at $x/L = 0.11$ are higher than the velocity gradients at $x/L = 0.43$. However, the grid density is nearly the same at $x/L = 0.11$ and $x/L = 0.43$. Hence, the quality of the solution at $x/L=0.43$ is better than the quality at $x/L = 0.11$. This can be seen from the c_f plots. The match between the fine and the coarse grids is not up to the mark at $x/L = 0.11$ (due to the high gradients). In contrast, the c_f obtained from the fine and the coarse grids are in good accord at $x/L = 0.43$.

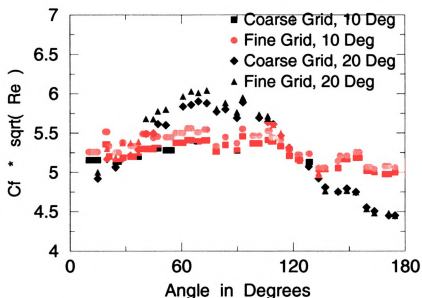


Figure 4.18 C_f distribution at $x/L = 0.11$ at 10 and 20 Degrees angle of attack

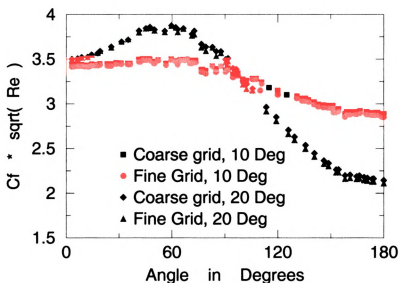


Figure 4.19 C_f distribution at $x/L = 0.43$ at 10 and 20 Degrees angle of attack

E. 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 4.21. The Chimera holes generated in the Cartesian grid by the prism grids are shown in Figure 4.20. The pressure distribution is shown in Figure 4.22. Detailed comparison with moving body experimental data will be carried out in the future.



Figure 4.20 Hole boundary generated in the adaptive Cartesian grid

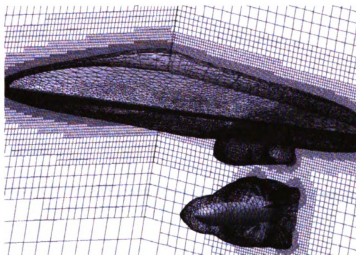


Figure 4.21 Overset adaptive Cartesian/prism grid for the wing-pylon-store case

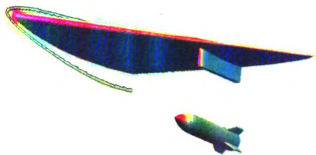


Figure 4.22 Computed pressure distribution for the wing-pylon-store case

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,

1. A stationary vortex ring was formed behind the sphere at Reynolds numbers lesser than 400. For Reynolds numbers between 400 and 1000, a vortex pair loop was observed in the wake region.

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. However the peak value of c_f is still negative as the effect of adverse pressure gradient dominates over the turbulent mixing.

The grid generator and flow solver have been coupled successfully to tackle a moving boundary flow problem with reasonable computational results. The results obtained by using the moving body flow solver were identical to the well-known results for a translating sphere in a laminar (and in inviscid) fluid. The moving body flow solver was also equipped to tackle a rotating degree of freedom. A rotating prolate in laminar flow was attempted. The c_p and c_f profiles looked realistic.

A. PLANS FOR THE FUTURE

A.1 Parallelize the code.

The current solver was implemented on a single processor. It is virtually impossible to run a moving body simulation involving more than a couple of million cells using one processor. In addition, the turbulent moving body problems require additional

computational time for solving the equations of closure and for resolving the laminar sub-layer. Hence the need for parallization of the code.

A.2 Further validation and demonstration with a high Reynolds number store-separation problem

Store separation problem is probably the ultimate test for a moving boundary solver. This problem involves unsteady flow that experiences huge separation. In addition, this problem requires lots of computational time and resources.

A.3 Enable solution based grid adaptation

Solution based grid adaptation is straightforward when the grids are Cartesian grids. Solution based adaptation is unnecessary for prism grids as their grid density is much higher than the Cartesian grids.

A.4 Enable the moving body solver to handle 6 DOF (Degrees of Freedom) i.e. 3 translations + 3 rotations (currently only 4 DOF are possible)

This is necessary for simulating an actual store separation problem. After the store is released, the store is given a nose-down maneuver. The aircraft needs to be given a nose-up maneuver and a yaw maneuver. Thus the need for all the 6 degrees of freedom.

BIBLIOGRAPHY

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. Venkatakrisnan 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 Journal*, 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 model for aerodynamic flows," *La Recherche Aeronautique* 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 its 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.
 44. Kotapati, A.R, Squires K. and Forsythe, “J.R. Prediction of a Prolate Spheroid Undergoing a Pitchup Maneuver”, *AIAA Paper No 2003-0269*
 45. Chesnakas, C.J. and Simpson, R.L., 1997, “Detailed investigation of the three-dimensional separation about a 6:1 prolate spheroid”, *AIAA J.*, **35**(6), pp. 990-999
 46. Wetzel, T.G and Simpson, R.L., 1998, “Unsteady Crossflow Separation Location Measurements on a Maneuvering 6:1 Prolate Spheroid”, *AIAA J.*, pp.2063-2071

MICHIGAN STATE UNIVERSITY LIBRARIES



3 1293 02736 2106