1-1-2002

Application of Generalized Grids to Turbomachinery CFD Simulations

Rajkeshar Singh

APPLICATION OF GENERALIZED GRIDS TO TURBOMACHINERY CFD SIMULATIONS

By

Rajkeshar Singh

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computational Engineering
in the College of Engineering

Mississippi State, Mississippi

December 2002

APPLICATION OF GENERALIZED GRIDS TO TURBOMACHINERY CFD SIMULATIONS

By

Rajkeshar Singh

Approved:

Bharat K. Soni Professor of Aerospace Engineering (Major Professor)	Roy Koomullil Assistant Research Professor Engineering Research Center (Thesis Advisor)
J. Mark Janus Associate Professor of Aerospace Engineering (Committee Member)	Edward Luke Assistant Professor of Computer Scinence (Committee Member)
Boyd Gatlin Associate Professor of Aerospace Engineering (Graduate Coordinator)	A. Wayne Bennett Dean of the College of Engineering

Name: Rajkeshar Singh

Date of Degree: December 13, 2002

Institution: Mississippi State University

Major Field: Computational Engineering

Major Professor: Dr. Bharat K. Soni

Title of Study: APPLICATION OF GENERALIZED GRIDS TO TURBOMACHINERY CFD SIMULATIONS

Pages in Study: 66

Candidate for Degree of Master of Science

A generalized grid based technique was developed for handling the relative motion of grids in CFD simulations involving rotating machineries. In the present method, the relative motion between the grid blocks is handled by splitting the cell-faces at the interface and updating the grid data structure appropriately. The resulting grid will have cells and cell-faces with an arbitrary number of nodes and which are stored in a cell-face based data structure. The current methodology is developed for cells with any number of nodes. However, the present work supports only tetrahedral elements at the interface of the rotating grid-blocks at the beginning of the simulation. Also the present approach can handle multiple objects in the domain of interest which are rotating in arbitrary directions. The current approach was tested by rotating a generalized grid for a single un-ducted SR7 propeller with eight blades designed with 41 degrees of sweep at the tip. This was also tested for two counter rotating SR7 propellers. After every rotation the new grid was tested for negative volumes, folded cell-faces, proper connectivity of nodes forming the cell-faces, and for gaps. Preliminary work has been conducted to couple the grid generation strategy to a generalized grid based flow solver.

ACKNOWLEDGMENTS

My sincere thanks to Dr. Roy Koomullil for making himself almost always available for constructive comments and suggestions, as and when needed. I also thank him for being so patient during my entire stay here and giving me an opportunity to work on a variety of other small projects which gave me a broader perspective of the things and motivated me to continue my studies in these areas.

I want to thank Robert Nichols of AEDC, Tullahoma, for suggesting this problem which eventually became my thesis. I am greatly thankful to Dr. Bharat Soni who as the director of the CCS group in the Mississippi State University, gave us an opportunity to be a part of this group and provided his constant support and motivation.

My sincere thanks go to Dr. J. Mark Janus of Aerospace engineering for his suggestions over some issues related to turbomachinery CFD simulations. I am very thankful to Dr. Edward Luke of Computer science Dept. who was very considerate and helpful to provide me with some very useful informations related to this work.

At this moment, I would like to thank ERC as a whole for providing an excellent work-environment and an opportunity to work in this place. It will remain a matter of pride to have been associate with ERC.

I would not miss this opportunity to thank my apartment mates Satish, George (better known as Jumbo) and my room mate Mukti, along with some honorary members of apartment 165, like Jaya, Kums and Bhaskar, whose presence kept the place alive for two years and it was nothing less than memorable to have spent the past two years with these wonderful people around.

Nothing can ever be concluded without a mention of my parents, to whom I owe every single step taken in the forward direction. I would also, not miss to thank Vidya, for always being there and who was more concerned about me finishing my work on time than I myself could do!. A lot of names may have been missed in this short note but their presence collectively makes all the difference in life and I hope to be as lucky in the future too!

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Research interests in the field of rotating machines are whetted by it's variety of applications in organizations and industries such as aerospace, automotive, space, and marine research. Due to forbidding cost of experimental investigation and rapid development of computational power and relative economic viability of a computational simulation, CFD is emerging as a major tool for analysis and better understanding of the complex physics involved in the flow regime of such applications. The wide and critical application interests have been a driving force for research in this area. For the past several years, computational experiments in these areas have been a major thrust of research at the Engineering Research Center at Mississippi State University, with significant advancement[1][2].

A CFD simulation has essentially two critical components, namely, the domain decomposition and solution of the governing equations on the discretized domain. The process of domain decomposition is called grid generation while the solution algorithm is a procedure for discretizing and solving the governing mathematical model of the physics involved, in the form of a set of differential or integral equations. The aim of the present work confines itself to the first component of the simulation process, specifically, handling of the grid related issues involved in the CFD simulation of rotating machines.

The different types of grids have their advantages and disadvantages in terms of both solution accuracy and the complexity of the grid generation process. The process of grid generation can be broadly classified in to two categories based on the characteristics of the data structure and the topology of the elements to fill the domain. These

two basic categories are known as structured and unstructured grids. A structured grid is defined as a set of hexahedral elements/cells with an implicit connectivity of the points in the grid. On the other hand, an unstructured grid is defined as a set of tetrahedral elements wth an explicitly defined connectivity. The process of structured grid generation involves breaking the domain into smaller blocks in which the structured grid is generated[3]. An unstructured grid generation has two basic steps, point creation and point connection[4][5]. The choice of location of point creation and point connection gives the flexibility that makes this type of grid a favored choice for complicated computational domains, while a structured grid may also give a good quality of domain discretization but the process is very time consuming as the grid may need to be manually broken into several blocks depending on the nature of the geometry. The flexibility and the convenience makes an unstructured grid a favorable choice, but at a cost, as the solution accuracy may be relatively poor due to presence of skewed elements in sensitive regions like boundary layers. Although a structured grid may be a more desirable attribute in terms of the solution accuracy, an unstructured grid may be an easier way out for complicated geometries. Due to the accuracy and efficiency concerns with an unstructured grid as it may take a large number of elements to fill a region with this type of grid[6], a compromise is made in the form of hybrid grid which uses the desirable features of both of the above mentioned generic grid types. In a hybrid grid, the viscous region can be filled with prismatic or hexahedral cells while the rest of the zone can be filled with tetrahedral cells[6][7][8]. It has been observed that a hybrid grid in viscous regions, creates much lesser number of elements than completely unstructured grid with similar resolution. This idea has been generalized further to give rise to what is called a generalized grid[9][10]. Since generalized grids have no restriction on the number of edges or faces on a cell, it makes this type of grid extremely flexible for topological adaptativity.

The structured, unstructured and hybrid grids have been extensively employed in simulation of rotating machines[11][12][13][1][14]. The present work focuses itself on developing a method for handling the grid when there is a relative rotational motion between the grid blocks in such applications. A sliding grid[15] approach is one of the common tools for controlling the grids while simulating geometries in relative motion, typically encountered in turbomachinary simulations. In such situations, due to non-conformity of the faces of the grid-blocks, the solution is interpolated[16] to preserve the conservation laws. Another approach employs deformation in a local region around the interface. In this method, the grid is locally regenerated when the deformed grid is no longer within the quality requirements. Another deformation based method applied to this category of problems is called Local Grid Distortion (LGD) which is used with structured grids[1][17]. In LGD method, the grid is not regenerated when the quality needs improvement, but rather suitably reconnected.

The present work is similar to the first category. In this work, a grid with tetrahedral elements on the grid-block interface, was used. When a grid block rotates with respect to another grid block, the interface surfaces slide over each other and then the interface faces of one block are suitably broken based on the region of overlap with the interface faces of the other grid-block. Due to breaking of the faces by it's overlapping faces from the other block, generalized faces and cells appear on the interface. The capabilities of a generalized CFD solver can be exploited for the CFD simulation of rotating machines with the present method of handling the grids.

This report has been organized into six chapters. The second chapter gives an overview of the work along with a discussion of some preliminary settings and algorithms needed in the present application development. This chapter also presents the details of the interface surface projection and area coordinate searching on such planes. The third chapter deals with the grid update strategy after the grid-blocks undergo a relative motion. In this chapter a two dimensional analogy of the actual three dimensional case

has been presented along with a brief discussion of the process in the three dimensional case. The fourth chapter presents a basic introduction of the generalized grid solver concepts. The chapter five contains some results to demonstrate the performance of this method and the stage of the development of this work. The summary and conclusion of the present work are given in the sixth chapter.

CHAPTER II

OVERALL SIMULATION PROCESS AND INITIAL SETUP

This chapter describes the overall simulation strategy that is adopted for CFD simulations in turbomachinery applications. The current methodology is general enough to handle multiple rotating bodies with arbitrary rotation-directions and orientation of axes of rotations. In the present work, a cylindrical grid is generated around each rotating object and all these different grids are enclosed in a stationary grid. After each rotation, the grid, on the interfaces of the rotating and the stationary grids, is updated using the procedure described in the following sections.

2.1 Overview of Steps for CFD Simulation of Rotating Machineries

The figure 2.1 shows the basic components of the procedure for CFD simulation of a rotating machine using the present methodology to handle the changes in the grid due to rotation of one or more components.

First, all the grid blocks are assembled into a single block grid. One of these grid blocks is a stationary grid while the other blocks represent grids around rotating bodies. The assembled grid is stored under the name *OriginalGrid*. This grid is the first input grid to the solver. The solver will dictate a suitable angle of rotation for each rotating grid block. After this, the assembled grid named *OriginalGrid*, is copied to a new name *NewGrid* and then this new grid is updated according to the present method to incorporate the rotation of its components. After this update, the resulting single block grid is passed on to the solver for obtaining the solution. When the next set of rotation-

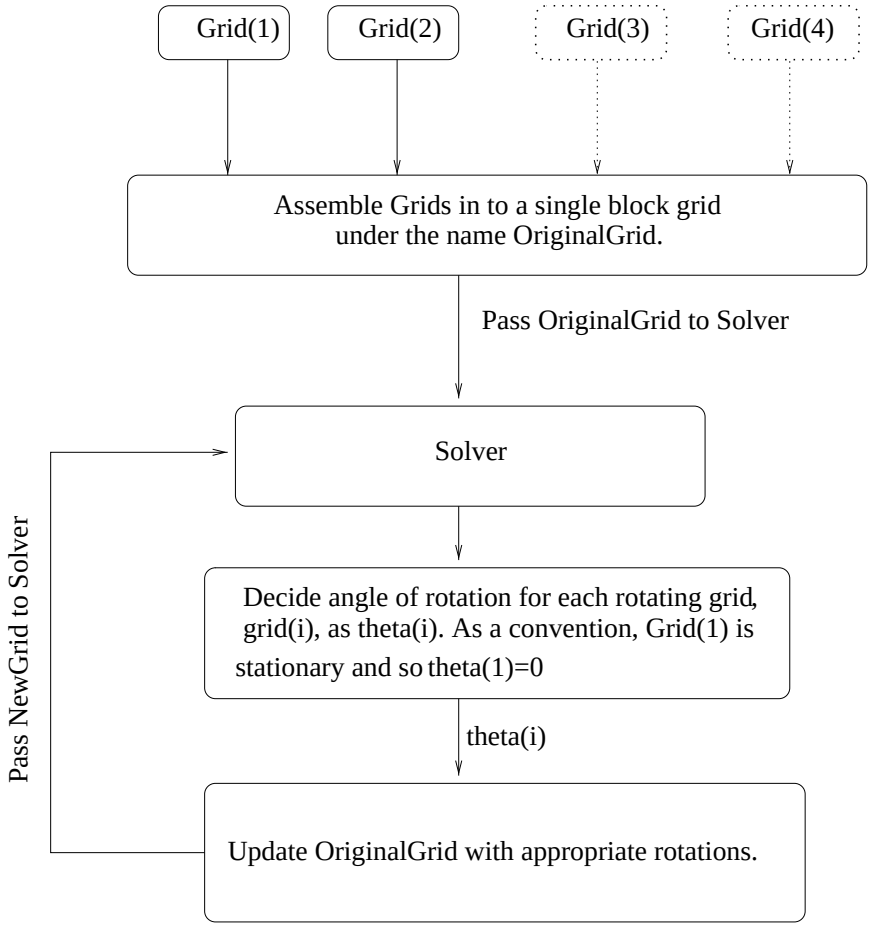

Figure 2.1: Outline of steps of CFD simulation of a rotating machine.

requests are placed by the solver, the same steps from copying the assembled grid to a new name and then updating it, are followed. The first step, combining grid blocks into a single grid block, is called the initial grid building process. The new updated grid, after rotation of any component grid block, is obtained by making appropriate changes in the assembled grid data.

2.2 Initial Setup

Initial grid building is an important part of the overall work aimed at achieving the capability for handling grid related issues involved in the CFD simulation of rotating machines using generalized grids. This part consists of preparing the initial grid by assembling grids around the rotating components with the stationary grid which houses these components. To illustrate this process, consider a fan with it's axis of rotation as shown in figure 2.2(a). First a cylindrical surface is created around this fan with the axis

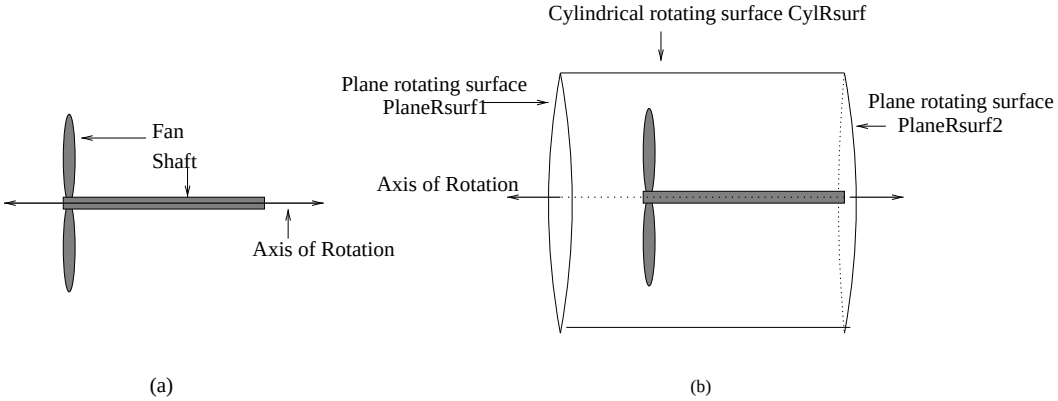

Figure 2.2: (a) A schematic diagram of a fan with it's axis of rotation and, (b) a cylindrical surface, extending up to the rear end of the shaft, created around the fan

of the cylinder coinciding with the axis of rotation of the fan. The cylindrical surface extends up to the rear end of the shaft so that the end of the shaft lies on one of the side surfaces, as suggested by the figure 2.2(b). A volume grid is generated in the region enclosed by the cylindrical surface and the two planar side surfaces. For the present

implementation, the only restriction for the volume grid is that all the faces having a node on any of these three surfaces (cylindrical and the two planar side surfaces) should be triangular in shape. However it is acceptable to have any kind of topological shapes for the faces and cells which do not have any node on any of these three surfaces. This grid around the rotating component will be referred to as the rotating grid. Now, a stationary grid needs to be created which will house this rotating grid. As shown in figure 2.3, this stationary grid contains a cylindrical 'hole' which is formed by the same surfaces as those surrounding the fan. This makes it possible to match each face and node of the outer surfaces of the rotating grid with the corresponding surfaces of the stationary grid. In the figure2.2, the cylindrical and the two planar side surfaces of the rotating grid have been named as $CylRsurf$, $PlaneRsurf1$ and $PlaneRsurf2$, respectively. The corresponding surfaces of the stationary grid, shown in figure 2.3, have been named as $CylSsurf$, $PlaneSsurf1$ and $PlaneSsurf2$.

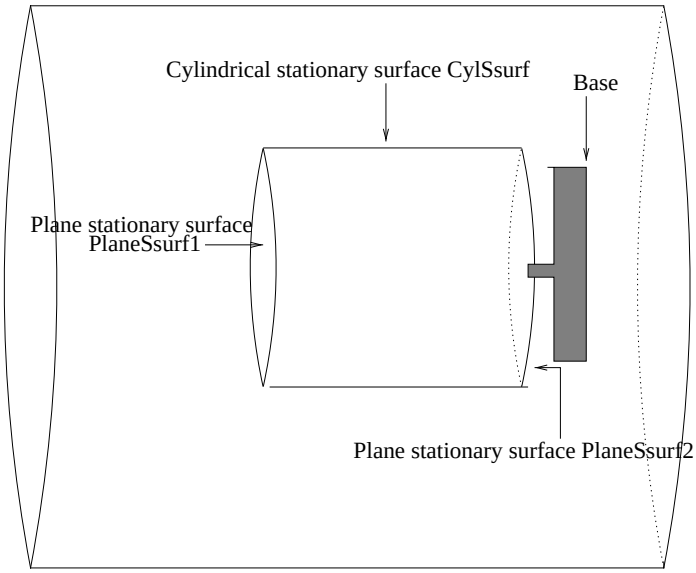

Figure 2.3: A schematic diagram of an outer or stationary grid-domain to house the grid around the fan

As shown in figure 2.3, the stationary grid may consist of a base on which the rotating shaft of the fan can be rested. For the present development, a small extrusion from the

base, as shown in figure 2.3, is created to avoid the planar side surface of the 'hole' from touching the surface of the base. When the rotating grid is placed in this 'hole' of the stationary grid, both the grids are joined to created a single block grid, as shown in figure 2.4(c). This process of creating a single block grid by combining the two grids is called grid assembling process and it has been described in the following section.

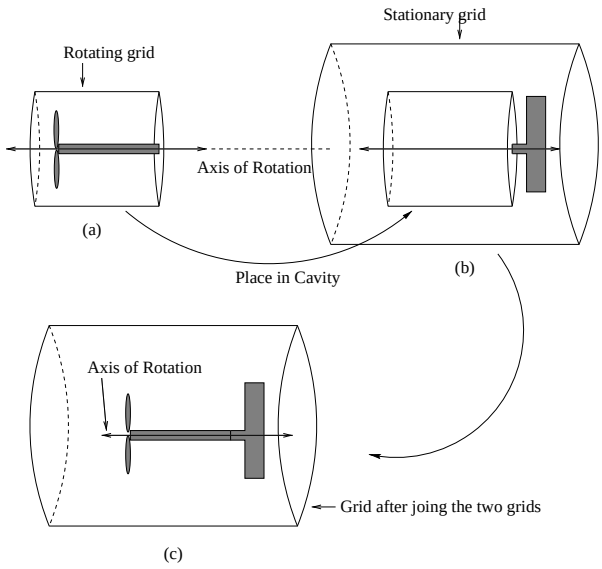

Figure 2.4: A schematic grid assembly depicting creation of a single block grid by combining stationary grid with the grid around the fan.

2.3 Assembling the Grids

When the rotating grid is placed inside the stationary grid, surfaces $CylRsurf$, $PlaneRsurf1$ and $PlaneRsurf2$ of rotating grid correspond to the surfaces $CylSsurf$, $PlaneSsurf1$ and $PlaneSsurf2$, respectively. To affect the grid assembling process, a match needs to be established between faces and nodes of each surface of one grid and those of the corresponding surfaces in the other grid. Next we assign a local coordinate system to the rotating grid. The $Z-Axis$ of the local coordinate system, for the rotating grid, is selected along the axis of rotation. The origin is set to the intersection point of the $Z-Axis$ with one of the selected side surfaces which is the surface $PlaneRSurf$

in the present example. The $X-Axis$ is a line passing through the origin and some selected point on the same side-surface, as shown in figure 2.5(a), while the $Y-Axis$ is appropriately assigned to create a right handed orthogonal coordinate system. The same coordinate system is assigned to the cylindrical 'hole' in the stationary grid also, as shown in figure 2.5(b).

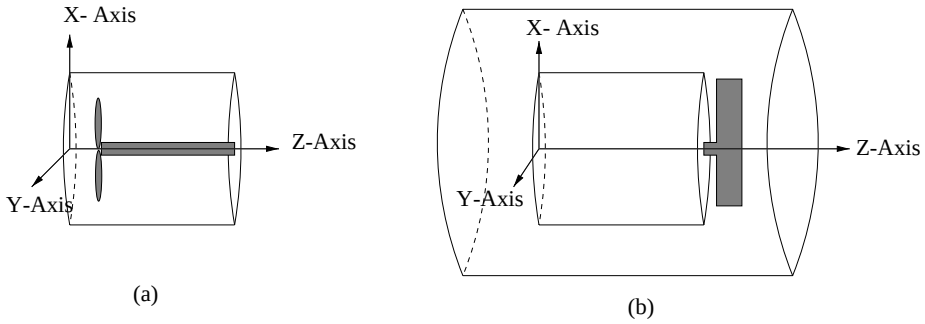

Figure 2.5: (a) Local orthogonal coordinate system for the rotating grid and, (b) the same coordinate system assigned to the stationary grid.

2.3.1 Surface Projection

A local face and node data is created for each surface and this local face and node numbering will be used for all the searching and further operations. A map between the local face and node numbering on the surfaces and their numbering in the corresponding grid is maintained.

For matching the grid-faces and the nodes on the surfaces of rotating grid with the corresponding surfaces of the stationary grid, each of these surfaces are projected on a two dimensional $\psi-\eta$ plane. The two side surfaces of the rotating grid and the corresponding side surfaces of stationary grid are planar in nature, so on these surfaces a node $'n'$ with $localX(n)$ and $localY(n)$ as the x and y coordinates in the local coordinate system, is assigned the $\psi-\eta$ coordinates in the projected plane as,

$$\psi(n) = localX(n) \tag{2.1}$$

$$\eta(n) = localY(n) \tag{2.2}$$

For projecting the cylindrical surfaces, the process is similar to cutting these surfaces along a line parallel to the axis and then opening them. For every node $'n'$ on these two surfaces $CulRsurf$ and $CylSsurf$ of figure 2.2 and figure 2.3, an angle θ, which the node makes with the $X-Z$ plane in the local coordinate system of the stationary grid, is computed and then these surfaces are cut along a line on which $\theta = 0$. When these surface are cut and opened, the $\psi-\eta$ coordinates for a node $'n'$ are assigned as,

$$\psi(n) = localZ(n) \tag{2.3}$$

$$\eta(n) = \theta(n) \tag{2.4}$$

where $localZ(n)$ is the Z coordinate of the node in the corresponding local coordinate system. After all the surfaces are projected, we start assembling the grids by matching the faces and the nodes on the corresponding surfaces of the two grids.

2.3.2 Surface Matching

First we match the faces of the side surfaces, projected on the $\psi-\eta$ plane. Consider matching the faces and nodes of the surface $PlaneRsurf1$ of the rotating grid with the faces and nodes of the surface $PlaneSsurf1$ of the stationary grid (figure2.2 and figure 2.3), in the projected plane. For searching a point on these surfaces, the standard area coordinate search can be employed in the projected plane. Each face and node of surface $PlaneRsurf1$ is matched with an appropriate face and node on the surface $PlaneSsurf1$.

2.3.3 Surface Matching Algorithm

Let an array $matchRf(Rf)$ be a face on $PlaneSsurf1$ to which the face $'Rf'$ of $PlaneRsurf1$ matches with. Let $ieRf(j, Rf)$ and $ieSf(j, Sf)$, for $j = 1, 2, 3$, be the arrays containing the three face neighbors of a face Rf and Sf on $PlaneRsurf1$ and $PlaneSsurf1$ respectively, as shown in figure 2.6. As seen in 2.6, the faces Rf and Sf

have their vertices marked with an index ranging from 1 to 3 in an order depending on the orientation of the face. Let us denote the node number of a vertex marked with $index = j$, on faces Rf and Sf, by $ipRf(j, Rf)$ and $ipSf(j, Sf)$, respectively. Let an array $matchPRf(n)$, denote a node on the surface $PlaneSsurf1$ to which a node $'n'$ on surface $PlaneRsurf1$ will be matched with.

First, search for the centroid of the face, with local face number one, on the projection of surface $PlaneSsurf1$ and assign the location-face number to $matchRf(1)$. Once local face number one is matched to a face on $PlaneSsurf1$, the local grid connectivity data on these surface is utilized for matching the other faces of $PlaneRsurf1$ to appropriate faces of $PlaneSsurf1$.

Figure 2.6: (a) Faces on the rotating surface and, (b) faces on the stationary surface face vertex indices and the corresponding convention for face neighbor data.

Let $nFaceRsurf1$ be the total number of faces on the surface grid on $PlaneRsurf1$ and $nNodeRsurf1$ be the total number of nodes on this surface grid. Let us denote a symbolic function which searches for the centroid of a face $Rface$ on the projection of surface $PlaneRsurf1$, by a function named $AreaCoordSearch(Rface, InitGuess)$. This function returns the local face number of the location-face on surface $PlaneSsurf1$. $InitGuess$ is the initial guess-location supplied to the function. Let $S1$, $S2$ and $S3$ be the area coordinates of a node $'P'$, from the projection of surface $PlaneRsurf1$, on a triangular face of projected $PlaneSsurf1$. The convention for $S1$, $S2$ and $S3$ is shown in figure 2.7. Let $Smax$ denote the absolute maximum among the three area coordinates $S1$, $S2$ and $S3$.

Algorithm:

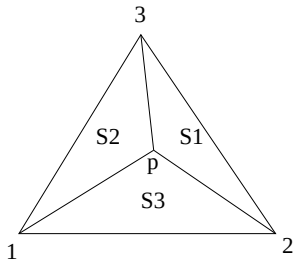

Figure 2.7: Convention for the area coordinates $s1, s2$ and $s3$ of a point 'p' based on the indices 1, 2 and 3 of the vertices of the face

Create an empty array called $FaceList$

Rface=1

InitGruess=1

matchRf(Rface)=AreaCoordSearch(Rface,InitGuess)

FaceListSize=1

Facelist(1)=1

DO $pointer = 1, FaceListSize, 1$

$Rface = FaceList(pointer)$

$facematched = matchRf(Rface)$

$iefm1 = ieSf(1, facematched)$

$iefm2 = ieSf(2, facematched)$

$iefm3 = ieSf(3, facematched)$

$ieRf1 = ieRf(1, Rface)$

$ieRf2 = ieRf(2, Rface)$

$ieRf3 = ieRf(3, Rface)$

$n1 = ipRf(1, Rface)$

$n2 = ipRf(2, Rface)$

$n3 = ipRf(3, Rface)$

Compute area coordinates $S1$, $S2$ and $S3$ of the node $n1$ with

respect to the face $facematched$ on the projected surface.

$Smax$ is absolute maximum of $S1$, $S2$ and $S3$

IF $Smax = S1$

$matchPRf(n1) = ipSf(1, facematched)$

$matchPRf(n2) = ipSf(3, facematched)$

$matchPRf(n3) = ipSf(2, facematched)$

IF$(ieRf1 > 0)$ $matchRf(ieRf1) = iefm1$

IF$(ieRf2 > 0)$ $matchRf(ieRf2) = iefm3$

IF$(ieRf3 > 0)$ $matchRf(ieRf3) = iefm2$

ENDIF

IF $Smax = S2$

$matchPRf(n1) = ipSf(2, facematched)$

$matchPRf(n2) = ipSf(1, facematched)$

$matchPRf(n3) = ipSf(3, facematched)$

IF$(ieRf1 > 0)$ $matchRf(ieRf1) = iefm2$

IF$(ieRf2 > 0)$ $matchRf(ieRf2) = iefm1$

IF$(ieRf3 > 0)$ $matchRf(ieRf3) = iefm3$

ENDIF

IF $Smax = S3$

$matchPRf(n1) = ipSf(3, facematched)$

$matchPRf(n2) = ipSf(2, facematched)$

$matchPRf(n3) = ipSf(1, facematched)$

IF$(ieRf1 > 0)$ $matchRf(ieRf1) = iefm3$

IF$(ieRf2 > 0)$ $matchRf(ieRf2) = iefm2$

IF$(ieRf3 > 0)$ $matchRf(ieRf3) = iefm1$

ENDIF

IF $ieRf1 > 0$ and $ieRf1$ is not already in $FaceList$ array

$FaceListSize = FaceListSize + 1$

$FaceList(FaceListSize) = ieRf1$

ENDIF

IF $ieRf2 > 0$ and $ieRf2$ is not already in $FaceList$ array

$FaceListSize = FaceListSize + 1$

$FaceList(FaceListSize) = ieRf2$

ENDIF

IF $ieRf3 > 0$ and $ieRf3$ is not already in $FaceList$ array

$FaceListSize = FaceListSize + 1$

$FaceList(FaceListSize) = ieRf3$

ENDIF

ENDDO

After matching all the faces of the the planar surfaces, the faces and nodes of the cylindrical surface $CylRsurf$ are matched with the faces of $CylSsurf$. The searching algorithm on the projection of surface $CylSsurf$ needs a some adjustments which have been described in the following section.

2.4 Area Coordinate Search

The area coordinate search on the two side surfaces is trivial as these surfaces are planar in nature. However, the algorithm for searching on the projection of the cylindrical surface needs some adjustments. This can be attributed to the way these surfaces have been projected on the $\psi - \eta$ plane. Since these cylindrical surface have been cut along the line of $\theta = 0$, as mentioned earlier, it is possible to have some faces cut and get inverted on the projected plane, which means that these faces will have some nodes with θ close to zero and some with θ close to 2π. This situation is illustrated in figure 2.8(a), which shows an exaggerated view of some faces which fall on the line along which the surface is cut. The figure 2.8(b) shows some such faces on the projected plane. Here, an

angle is said to be close to zero if it is less than π and it said to be close to 2π if it is at least π.

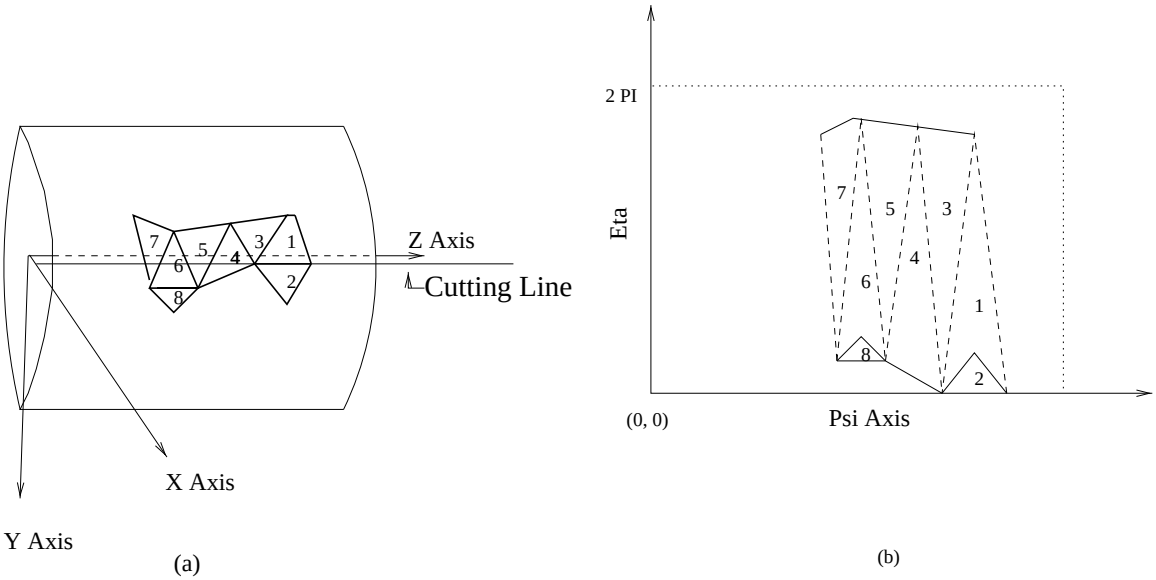

Figure 2.8: (a) Cylindrical surface faces 1-7 are lying on the line of $\theta = 0$, (the line of intersection of X-Z local plane with the cylindrical surface), along which the cylindrical surface is cut for projection. (b) Faces 1,3,4,5,6 and 7 get Inverted after projecting the cylindrical surface.

As seen in figure 2.8(a), the faces 4,5,6 and 7 are cut when the surface is cut along the line with $\theta = 0$ due to which these faces get inverted as shown in figure 2.8(b). We set θ to zero for all the nodes on this cutting line and due to this even the faces which have a node on the cutting line and have another node with θ close to 2π, get inverted. Faces marked as '1' and '3' are examples of this.

While performing the search operations on such surfaces, a reference angle is assigned to the point whose location is being searched. When searching for the centroid of a face which remained un-cut during projection, the reference angle is the average of the angles of the three nodes of the face. Suppose, we are searching for the centroid of a face Rf of projection of surface $CylRsurf$, on the projection of surface $CylSsurf$. Let us denote the $\psi - \eta$ coordinates of the three nodes $n(i)$ of the face Rf by $(\psi(i), \eta(i))$ and denote

the angles associated with the face vertices by $\theta(i)$, for $i = 1, 2, 3$. Denoting the centroid as $(\psi 0, \eta 0)$,

$$\psi 0 = \frac{1}{3}(\psi(1) + \psi(2) + \psi(3)) \tag{2.5}$$

$$\eta 0 = \frac{1}{3}(\eta(1) + \eta(2) + \eta(3)) \tag{2.6}$$

The reference angle θ_{ref}, assigned to the centroid will be,

$$\theta_{ref} = \frac{1}{3}(\theta(1) + \theta(2) + \theta(3)) \tag{2.7}$$

If this face Rf got cut during the projection of the surface and if, say, $\theta(1)$ is close to zero and $\theta(2)$, $\theta(3)$ are close to 2π, the η coordinate of the centroid is modified as,

$$\eta 0 = \frac{1}{3}((2\pi + \theta(1)) + \eta(2) + \eta(3)) \tag{2.8}$$

The reference angle θ_{ref} associated with the centroid is,

$$\theta_{ref} = 2\pi \tag{2.9}$$

Or, instead of replacing $\eta(1)$ of equation 2.6 by $(2\pi + \theta(1))$ in equation 2.8, the coordinates $\eta(2)$ and $\eta(3)$ of equation 2.6 can be replaced by $(\theta(2) - 2\pi)$ and $(\theta(3) - 2\pi)$ respectively and the η coordinate of the centroid is computed as,

$$\eta 0 = \frac{1}{3}(\eta(1) + (theta(2) - 2\pi) + (theta(3) - 2\pi)) \tag{2.10}$$

In this case, the reference angle θ_{ref} will be set as,

$$\theta_{ref} = 0 \tag{2.11}$$

Now we search for this centroid on the surface $CylSsurf$ using area coordinate search algorithm. If in the search path we encounter a face Sf which has been cut, some temporary adjustments are made for the coordinates of the three nodes of this face. These adjustments are similar to the ones that were made while computing centroid of such a face on the projected surface. Let $n(i)$ be the node of this face with $(\psi(i), \eta(i))$ as the coordinates and $\theta(i)$ as the their associated angles, for $i = 1, 2, 3$. If $\theta(1)$ is close to zero while the angles for the other two nodes are close to 2π, while searching a point with reference angle as θ_{ref} some temporary changes to the coordinates of this face are made. These changes for this face are,

IF($\theta_{ref} < \pi$)Then

$\eta 1 = \eta(1)$

$\eta 2 = theta(2) - 2\pi$

$\eta 3 = theta(3) - 2\pi$

ELSE IF($\theta_{ref} >= \pi$)Then

$\eta 1 = theta(1) + 2\pi$

$\eta 2 = \eta(2)$

$\eta 3 = \eta(3)$

ENDIF

When the search path encounters a face which was cut during projection, these corrected η coordinates $\eta 1$, $\eta 2$ and $\eta 3$ are used in place of $\eta(1)$, $\eta(2)$ and $\eta(3)$ as the η coordinates of the three nodes of the face. The figure 2.9 illustrates the process of searching for a point on such a surface. the figure 2.9(a) shows two points $P1$ and $P2$ which are being searched with an initial guess location-face to start the search. The possible search paths for $P1$ and $P2$ are shown in 2.9(b) and 2.9(c). Both the points $P1$ and $P2$ are located on faces which have been cut during projection, as shown in 2.9(a). As discussed earlier, while searching for the points points $P1$ and $P2$ which have their corresponding reference angles close to *zero* and close to 2π , some suitable adjustments

in the η coordinates are made when the search path meets a face which was cut during projection as shown in the figure 2.9(b) and 2.9(c).

2.4.1 Search Path Correction

Due to temporary adjustments needed in the face-node coordinates of the faces which have been cut during projection, it is possible for the search algorithm to fall in a vicious circle unless the search path is carefully selected. Consider the case depicted in figure 2.10 where a point P is to be searched. Face $face0$ was cut during the projection and so it gets inverted. As explained earlier, the face $face0$ can take the shape of the face containing nodes $n1^{'}$, $n2^{'}$ and $n3$ or the one containing the nodes $n1$, $n2$ and $n3^{'}$, depending on the need. Consider the situation in which the search path while searching for the point P, comes to the face marked as $face0$. Since the point P is close to 2π, this face $face0$ takes the shape of the face containing nodes $n1^{'}$, $n2_{'}$ and $n3$. In such a case, based on the sign of the area coordinates of the point $P1$ with respect to this face, we have two possible directions to advance the search. These two directions are indicated in figure 2.10 as $direction(1)$ and $direction(2)$. If we take $direction(1)$, face $face1$ will be the next face to be searched and this way we may just miss the point P. Instead, if we take the second route, $direction(2)$, we will get closer to finding the location of this point. A similar situation is possible while looking for a point with it's η coordinate near zero, in which case, we may just miss the point by being thrown to a face near $\eta = 2\pi$. So, while searching for a point which is in the upper half of the projected plane, the direction of search should be decided in such a way that, once the search is in the upper half of the plane, it is not thrown back to the lower half of the plane. A similar precaution is to be taken while searching for a point in the lower half of the projected plane.

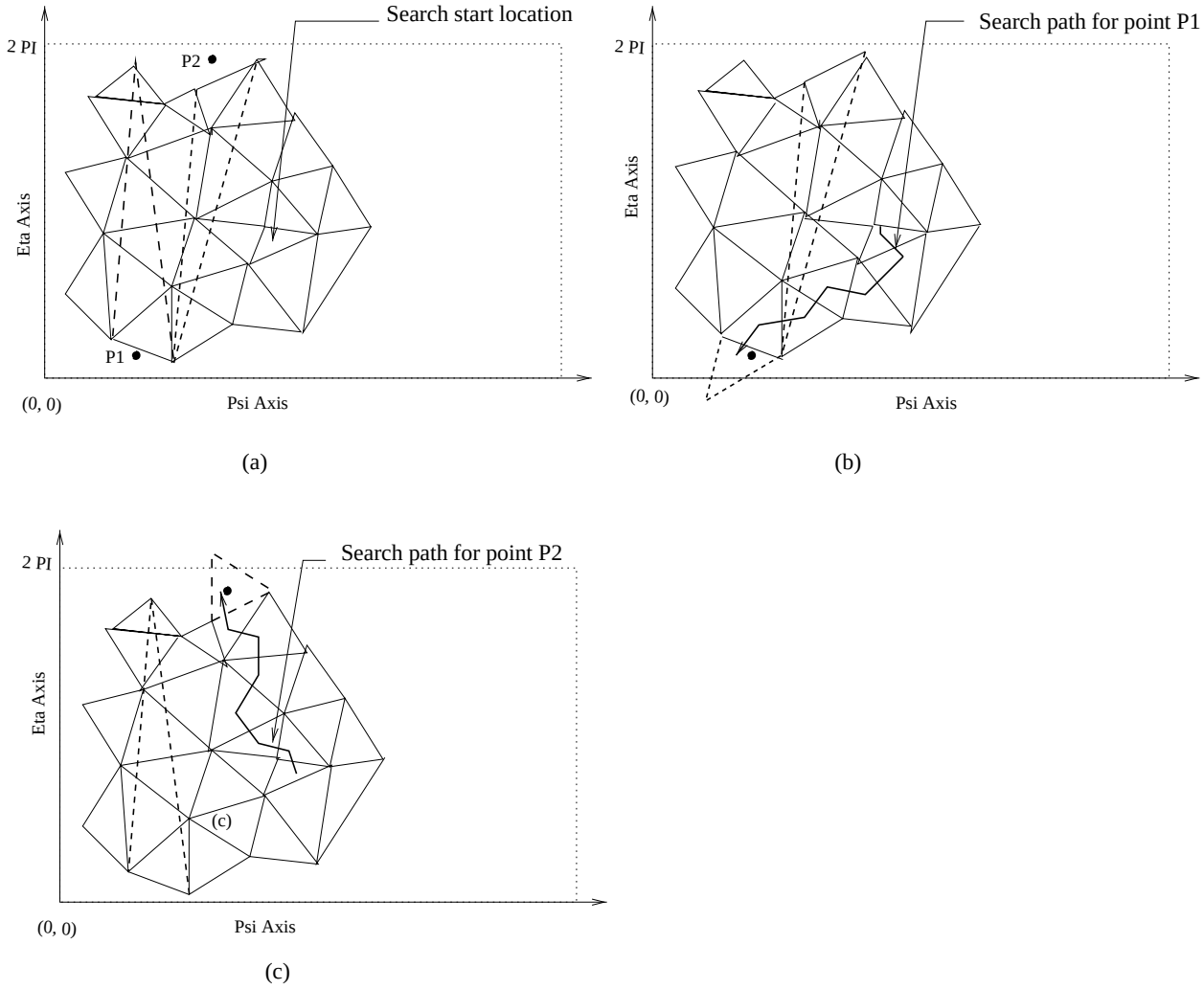

Figure 2.9: (a) Points $P1$ and $P2$ are to be searched on the projected plane with the search start location indicated. (b) A search path for locating point $P1$ with a temporary adjustment for the inverted face on the path based on the $\psi-\eta$ coordinates of the point being searched which has η coordinate near zero. (c)A search path for locating point $P2$ with a temporary adjustment for the inverted face, lying on the search path, based on the $\psi-\eta$ coordinates of the point $P2$ which has its η coordinate near 2π.

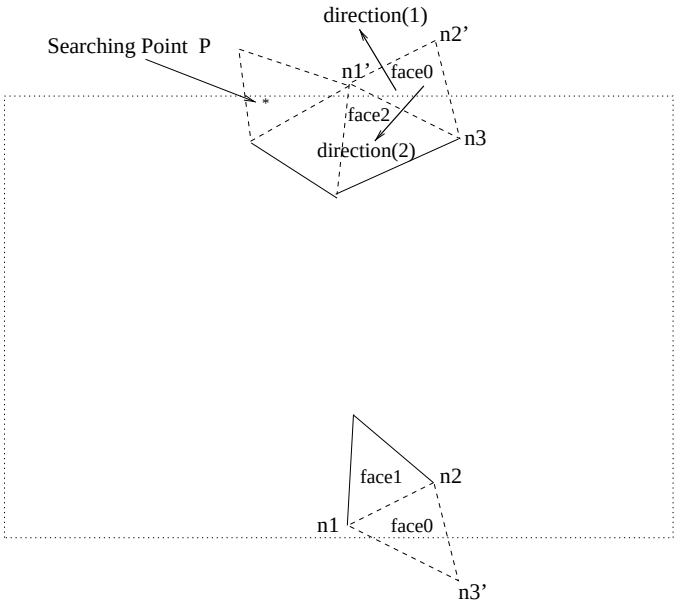

Figure 2.10: Two possible search-path directions, $direction(1)$ and $direction(2)$ from face $face0$, while searching location of point P. If path $direction(1)$ is chosen the next face to be searched will be $face1$ while for $direction(2)$ the next face to be search will be $face2$.

2.5 Applicability of the Present Method

After all the faces and the nodes of all the rotating surfaces are matched with the faces on the corresponding stationary surfaces, both the grids are combined to form a single grid-block. After rotating a grid block, only faces and cells near the interface of these two grids need to be updated. In the process of updating, new faces and nodes may be created.

With the way of combining the two grids, it is acceptable to enclose the rotating component with a cylindrical surface with varying radius along the axis. The only requirement is that after projecting the surfaces in $\psi - \eta$ plane, there should not be any folding of faces due to distortion. In case of multiple rotating components, the present work is applicable as long as it is possible to create disjoint cylindrical grids around each rotating component. Figure 2.11 shows a few possible grid domain-shapes around the rotating components. At the present stage of development, any of these or similar grid

domain- shapes are acceptable as long as each surface can be projected from the local to the $\psi - \eta$ plane, using the present method for projection, without any folding of faces due to distortion.

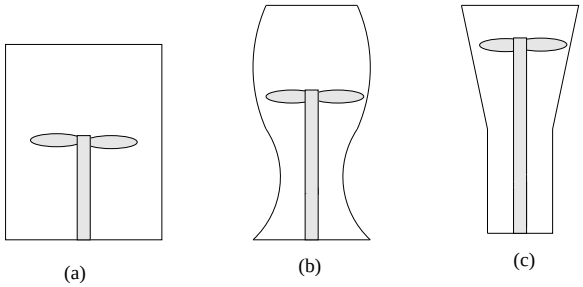

Figure 2.11: Some possible grid domains-shapes around the fan and the shaft, which can be used for the present stage of application-development.

CHAPTER III

GRID ROTATION STRATEGY

Before discussing the grid rotation and the following update strategy for a 3-D geometry, we shall first consider a simple two dimensional case. Although the procedure in actual three dimensional case is significantly more complicated to implement, the two dimensional case in the following section will give an understanding of the overall procedure followed in 3-D case.

3.1 A Simple 2-D Example

Consider a two dimensional fan enclosed by a circular boundary as shown in figure 3.1. First, this circular boundary is discretized and an unstructured grid is generated. Figure

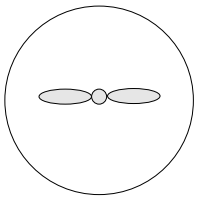

Figure 3.1: 2-D fan enclosed by a grid boundary

3.2(a) represents a symbolic grid around this fan. The same edge grid on the circular boundary is used in the generating the unstructured grid in the stationary region seen in figure 3.2(b). For convenience, we name the stationary grid as $grid1$ and the rotating grid as $grid2$. A local coordinate axis-system is created for grid2, with it's origin at the center of the circular boundary and the $X - axis$ passing through some selected point on the curve. The $Y - axis$ is created to form a two dimensional orthogonal coordinate

system. The same coordinate system is assigned to the circular hole in $grid1$, as shown in figure 3.2(b). As mentioned in chapter II, the grid in figure 3.2(a) is placed in the circular cavity in the grid of figure 3.2(b) and the grid is updated to form the single block initial grid as shown in figure 3.2(c).

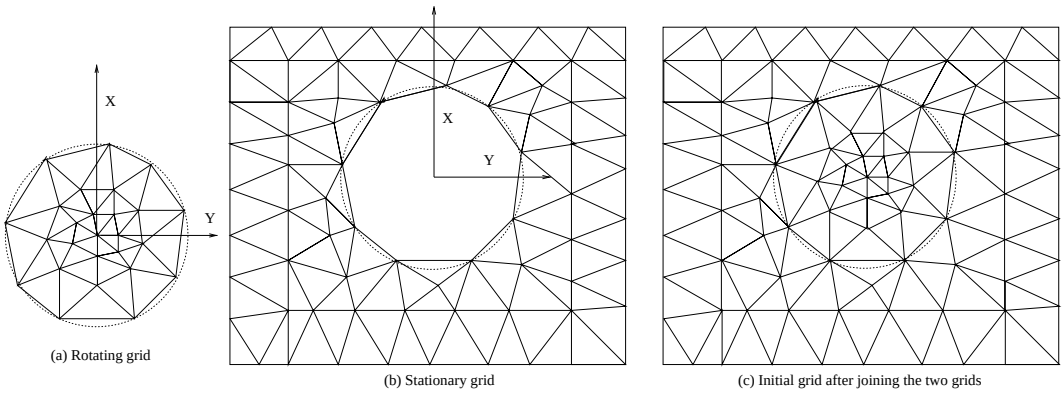

Figure 3.2: (a) 2-D Rotating grid (grid2), (b)Circular cavity, in the 2-D stationary grid (grid1), for housing grid2, (c) Initial single block grid obtained by assembling the two grids.

When $grid2$ is rotated by certain angle about the center of rotation which is the center of the circular boundary in this case, the outer boundary nodes of $grid2$ will slide along the circular contour, (shown in figure 3.3 by a dotted circular boundary), by this angle. Due to this, the edges of $grid2$ which are connected to it's outer boundary nodes are likely to penetrate in to $grid1$ as shown in figure 3.3. The dotted triangulation in this figure shows the grid position before rotation.

To update the grid in figure 3.3, for each node of the circular boundary of both the grids, an angle θ which the node makes with the $X - Axis$ of the local coordinate system of $grid1$, is computed. To explain the process of updating, consider two faces $rf1$ and $rf2$ of $grid2$ which have penetrated in to $grid1$. The figure 3.4 shows two faces $rf1$ and $rf2$ (both shaded) of $grid2$ which have their edges penetrating in to the triangles of $grid1$. In this figure, the prefix $'s'$ in the face number indicates a face of the stationary grid ($grid1$) while prefix $'r'$ indicates a face of the rotating grid ($grid2$). The table 3.1 gives the face data for all the faces shown in figure 3.4.

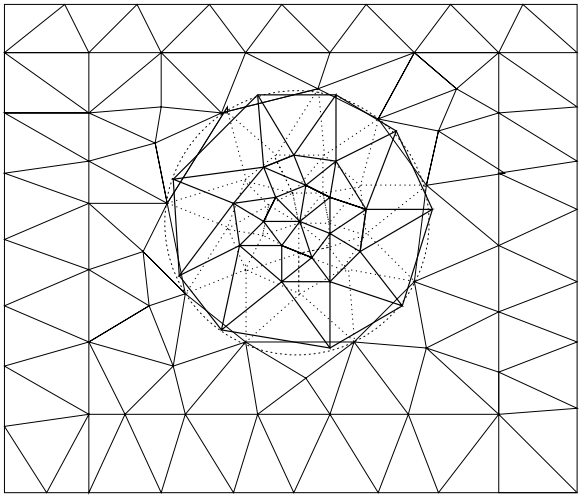

Figure 3.3: An intermediate stage after rotating grid2 and before updating in the 2-D case, showing the penetration of edge node of $grid2$ in to $grid1$ with the original unrotated position of $grid2$ shown by dotted lines.

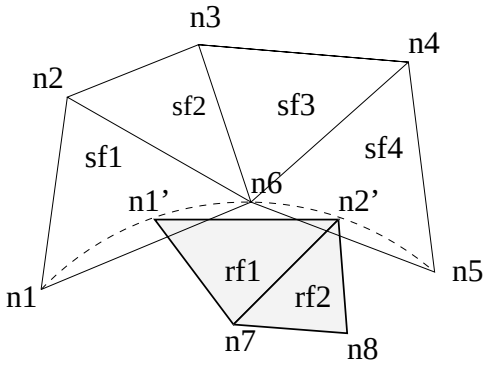

Figure 3.4: $grid2$ faces $rf1$ and $rf2$, after rotation, penetrating in to $grid1$

Table 3.1: Face-node data before updating

Face number	number of nodes	face nodes
$sf1$	3	$n1, n2, n6$
$sf2$	3	$n2, n3, n6$
$sf3$	3	$n3, n4, n6$
$sf4$	3	$n4, n5, n6$
$rf1$	3	$n1', n2', n7$
$rf2$	3	$n7, n2', n8$

Let $\theta(ni)$ denote the angle that a node $'ni'$, makes with the local $X - Axis$ of $grid1$ where $\theta(ni) >= 0$ and $\theta(ni) < 2\pi$. The edge grids on the circular curve of $grid1$ and $grid2$ are cut from the position of $\theta = 0$ and opened to make a projection in one dimension. On the projected line, the coordinates ψ of these nodes shown in figure 3.4 can be assigned as,

$$\psi(n1) = \theta(n1) \tag{3.1}$$

$$\psi(n5) = \theta(n5) \tag{3.2}$$

$$\psi(n6) = \theta(n6) \tag{3.3}$$

$$\psi(n1^{'}) = \theta(n1^{'}) \tag{3.4}$$

$$\psi(n2^{'}) = \theta(n2^{'}) \tag{3.5}$$

After this, the locations of projected nodes $n1^{'}$ and $n2^{'}$ are searched along the projection of the interface edge of $grid1$. As seen in figure 3.5, the node $n1^{'}$ can be located on the edge between nodes $n1$ and $n6$, while $n2^{'}$ can be found on the edge with nodes $n6$ and $n5$. let $u1$ and $u2$ be the parametric coordinates of the two points on the respective edges. Two new nodes, with the same parametric line coordinates, are introduced on the two associated edges of $grid1$. The two nodes $n1^{'}$ and $n2^{'}$ are replaced by these two newly created nodes $n9$ and $n10$, as shown in figure 3.6. All the faces of $grid2$ containing $n1^{'}$ and $n2^{'}$ are updated. The face $rf1$ is updated to contain the node $n6$ in an appropriate place as shown in figure 3.6.

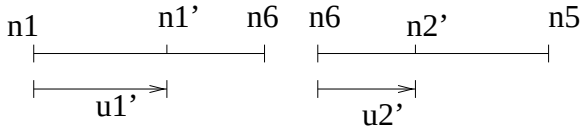

Figure 3.5: Location of nodes of a projected edge of face rf on the projected edges of $grid1$

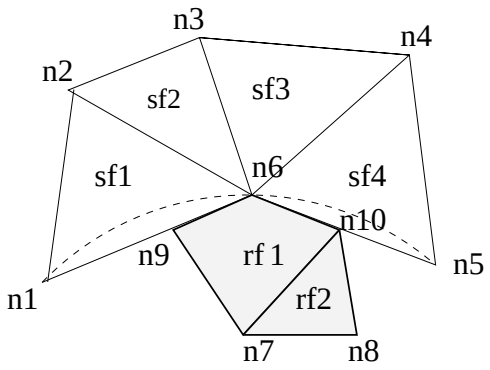

Figure 3.6: Faces after updating

After this update, the face data in table 3.1 is modified and it is given in table 3.2

Table 3.2: Face-node data after updating

Face number	number of nodes	face nodes
$sf1$	4	$n1, n2, n6, n9$
$sf2$	3	$n2, n3, n6$
$sf3$	3	$n3, n4, n6$
$sf4$	4	$n4, n5, n10, n6$
$rf1$	4	$n9, n6, n10, n7$
$rf2$	3	$n7, n10, n8$

Similar procedure is carried out for all the faces on the circular boundary to update the grid in figure 3.3. The updated grid is shown in figure 3.7 where all the faces which have undergone modifications are marked with an asterisk sign ($*$).

It can be noticed that during the updating process, only the interface nodes of the rotating grid are adjusted while those on the corresponding interface edge of the stationary grid remain unchanged. The face areas of the stationary grid do not change except those edges on the circular boundary which may be broken and new nodes may appear on them. All the changes in face areas occur only in the faces of the rotating grid. An analogous philosophy will be followed in the three dimensional situation. In 3-D situation, similar to the 2-D case, all the face area and cell volume changes occur only in the faces and cells in the rotating grid which are connected to the interface. In

3-D case, the surface faces of the stationary grid may be broken to form new faces but the sum of total face areas of the newly created faces and the updated old face, is same as the area of the old face before breaking. This is because the new nodes are placed on the faces of the stationary grid while the faces and cells of rotating grid are adjusted to contain this node. Similar to the case of the face-areas of stationary grid in 2-D, the cell-volumes of the 3-D stationary grid remain unaffected with, possibly, a few more nodes added to the interface cells.

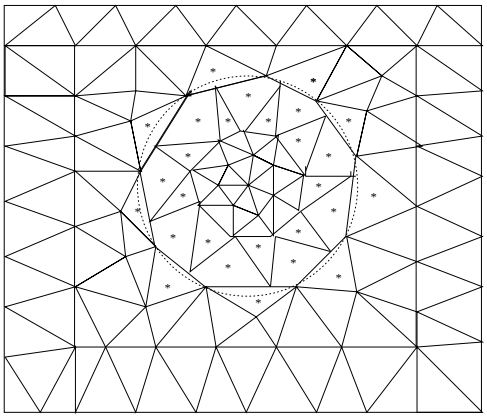

Figure 3.7: Updated 2-D grid after rotation of grid2

3.2 3-D Example

For the Present application in 3-D, we have three interface surfaces in the grid block surrounding the fan and three corresponding interface surfaces in the stationary grid, as shown in figure 2.2 and figure 2.3. We shall keep referring to the grid block around the fan-shaft combination as rotating grid or $grid2$ while the stationary grid block will be recognized by the name $grid1$, with the same surface nomenclature as indicated in figure 2.2 and figure 2.3.

When $grid2$ is rotated, the surface $PlaneRsurf1$ and $PlaneRsurf2$ of $grid2$ will slide on the surfaces $PlaneSsurf1$ and $PlaneSsurf2$ respectively. The cylindrical surface $CylRsurf$ of $grid2$ will slide on surface $CylSsurf$ of $grid1$. While updating the grid

after rotation, all the decision making processes are carried out on the projected surfaces which are essentially two dimensional and then the corresponding changes are made in the actual 3-D grid. These decision making processes involve search operations and decisions of creating new nodes and faces.

3.2.1 Grid Update After Rotation

Before rotation, each node of an outer surfaces of $grid2$ was coinciding with some node on the corresponding stationary surface of $grid1$. After rotation these surface nodes move from their original positions and the grid needs to be updated. Figure 3.8 shows an outline of the sequence of operations involved in this procedure. The update procedure has four basic steps of operations to be carried out as indicated in the figure 3.8 and is described in the following part of this section.

Step1: First step is to rotate all the nodes of the assembled single block grid that originally came from the rotating grid. Along with this, the projected surfaces of the rotating grid are also adjusted to incorporate the effect of the rotation. The two planar surface are simply rotated about the local $Z-axis$ to give the new projected plane after rotation. For the cylindrical surfaces, the ψ coordinates of nodes remain unaffected by this rotation. To obtain the new projection for cylindrical surfaces of rotating grid, the rotation angle is simply added to the η coordinates and if this sum, for a node, is larger than 2π then 2π is reduced from it's η coordinate.

Step2: Each node of the updated rotating surface is located on the corresponding stationary surface and if, based on some decided tolerance level, it is too close to a node on the stationary surface then it is merged with the associated node of the stationary surface. For some of the nodes of the planar rotating surface, it may not be possible to locate the position of these nodes on the corresponding surface of the stationary grid, as shown in figure 3.9. The figure 3.9 shows two stationary faces $sf1$ and $sf2$ on the cylindrical and a planar side surface of the stationary grid. Faces $rf1$ and $rf2$, shown

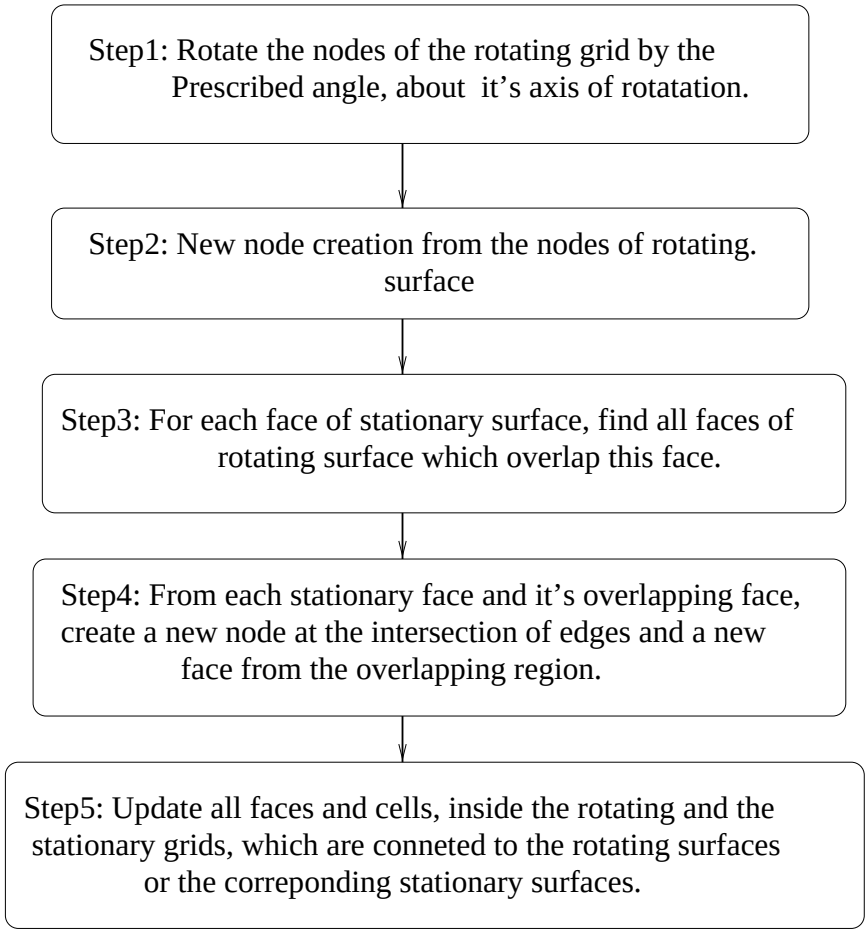

Figure 3.8: Steps of the algorithm followed for updating the grid data after rotation.

by dotted lines, belong to the rotating grid when it is rotated by some angle. It can be seen that the node $n2^{'}$ of the face $rf2$ can not be located on the planar side surface but it is still possible to locate it on the projection of the cylindrical surface. So the process, in this step, is started with the cylindrical surface first and then the planar side surfaces. For each node of the rotating surface which is not merged with any node of the stationary grid, a new node in the grid is created. For example, for node $n2^{'}$ of the rotating cylindrical surface, the location is the face $sf1$, so a new node is created on the face $sf1$ and $n2^{'}$ is replaced by this new node.

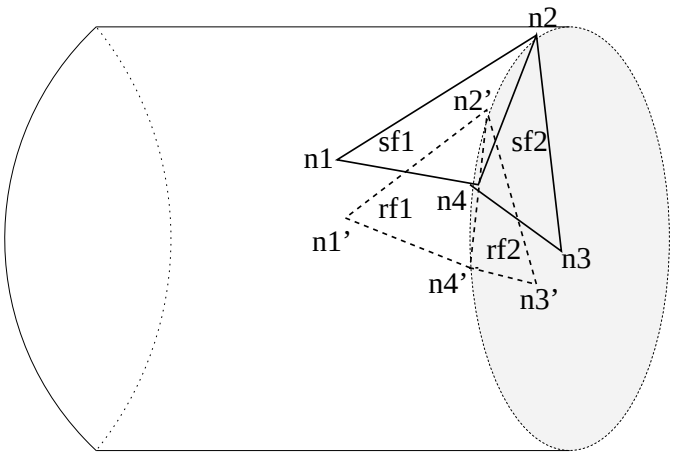

Figure 3.9: Faces $sf1$ and $sf2$ belonging to the stationary grid and faces $rf1$ and $rf2$ from the rotating grid, on the cylindrical and planar side surface(shaded), after $grid2$ is rotated.

Step3: In this step, new faces and further new nodes are created in the grid. First, for every face on a stationary surface, all the faces of the rotating grid are obtained which overlap this face on the projected plane. Consider figure 3.10, new nodes are created in the grid for every intersection of edges. The stationary face (shaded face in figure 3.10) is broken by the overlapping faces and new faces are created from each overlapping region. All these newly created nodes are in the plane of the parent face of the stationary grid which was broken by the overlapping faces of the rotating surface.

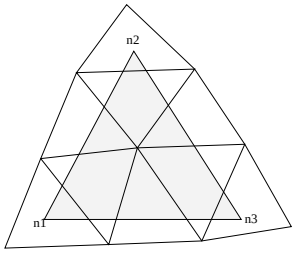

Figure 3.10: Intersection of a stationary surface face sf(shaded face) with the overlapping faces on the a rotated surface in the projected plane.

Step4: This is essentially the final step of the updating process. During this process, the faces and cells which are connected to an edge/node on the rotating or the corresponding stationary surface are updated to incorporate the addition of new nodes and faces in the grid. During this process further new faces are created. For the faces in the interior of the rotating grid which have an edge on the interface, the intersection nodes lying on such edges of these faces may not be coplanar with these faces, so these interior faces need to be broken appropriately. For reasons attributed to the method employed for handling some of the tolerance related issues discussed in the following section, the faces in the rotating grid block which have an edge on the planar interface surface, may also, sometimes, need to be broken.

Consider figure 3.11, which shows the intersection points on the edge of a rotating cylindrical surface-face *Rf* and the faces $f1$, $f2$, $f3$ in the interior of the rotating grid. As shown in figure 3.12, the interior faces $f1$, $f2$, $f3$ are broken in to several triangles since the intersection points may not be coplanar with these old faces.

Since the new nodes are placed on the faces or the edges of the stationary grid interface and it is the faces of the stationary grid interface which are broken to create new faces on the interface, these new nodes are always coplanar with their parent faces. Because of this, the faces in stationary grid block which are connected to the interfaces have all the new nodes, to be added, coplanar with them and hence need not be broken.

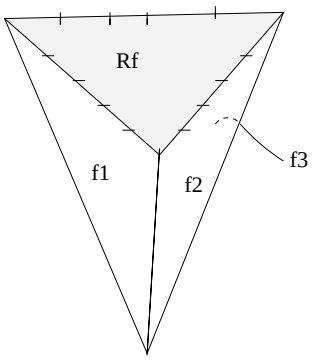

Figure 3.11: A face Rf on the rotating cylindrical surface $CylRsurf$ showing intersection points, with the faces of corresponding stationary faces, and the three faces $f1, f2, f3$ in the interior of the rotating grid ($grid2$) which have an edge on the rotating surface.

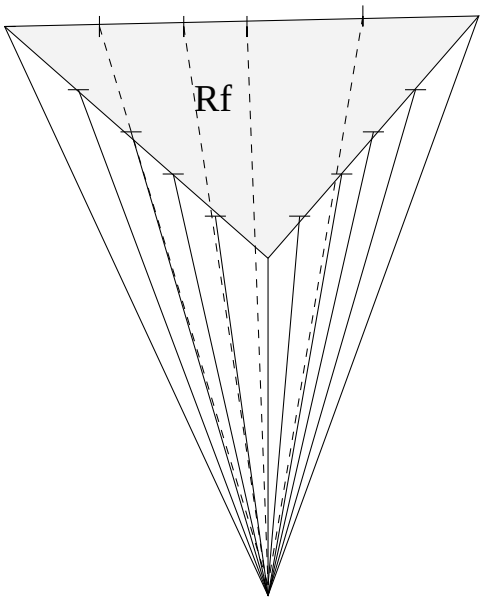

Figure 3.12: Each face inside rotating grid ($grid2$), connected to an edge of a face rf on the cylindrical interface, is broken in to triangular faces using the intersection nodes.

This way no non-interface face in the stationary grid is broken and whenever needed, the new nodes are simply added to the data of these faces and cells.

3.2.2 Tolerance Related Issues

Tolerance is one of the most critical part of any geometry related computation and should be dealt with judiciously to avoid serious problems. In the present work, the most important role played by tolerance is during deciding the creation of new faces and getting the nodes which create these faces. Creation of a new face involves a series of decision making process and all the decisions made should be consistent with each other. For example, while trying to find the intersection of two edges, if the two edges are 'nearly' parallel to each other, it creates a numerically ill-conditioned system of linear equations and the computed intersection coordinates may not be reliable. Consider figure 3.13 which shows two faces partially overlapping. We need to find the overlapping region which means getting the nodes which form this region. Assume that edge $ed3$ and edge $e5$ show an intersection point at location shown by point n'. In this case, it is necessary that no edge on the surface containing the face $f1$ shows an intersection point with edge $ed5$ at node $n1$, within the selected tolerance. In the same way, if the edges $ed2$ and $ed5$ give an intersection at node $n1$ based on some decided tolerance, every edge on the surface containing the face $f1$ should show an intersection with edge $ed5$ at not $n1$ only. Similarly, if within the tolerance selected for deciding whether a point is completely inside a face, the point $n1$ is shown to lie completely in the interior of face $f2$, it is imperative that the edge $ed2$ show an intersection with edge $ed5$ at a point other than it's end node $n1$. There are several such consistency related issues that need to be handled appropriately to avoid problems.

To briefly point out how such situations were tackled, consider two planar surfaces $surf1$ and $surf2$ which in essentially represent the interface surface grids in the projected plane. Before embarking on creating new faces from the overlap-regions of faces on these

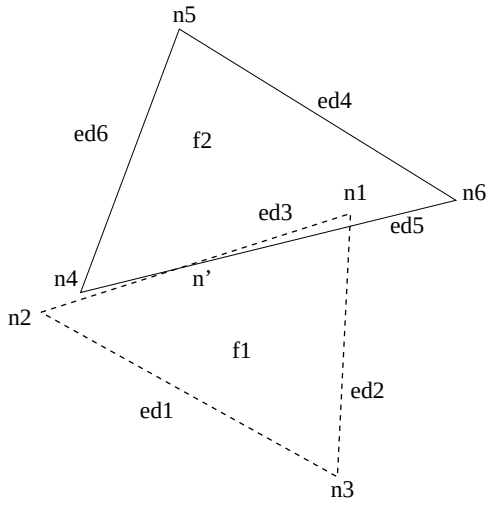

Figure 3.13: Edges $ed1$, $ed2$, $ed3$, belong to face $f1$ and edges $ed4$, $ed5$, $ed6$ belong to face $f2$.

two surfaces, it is necessary to deal with the above mentioned sources of troubles caused by the tolerance. The procedure for avoiding such problems is carried out in two stages. In the first stage, based on a selected tolerance for the area coordinate, all the nodes of surface $surf2$ are projected on an edge of the surface $surf1$ or merged with a node of surface $surf1$. After this stage, there will not be any node of surface $surf2$ which can be dangerously close to an edge or node of surface $surf1$ to cause problems. But, there may be nodes of the surface $surf1$ which can cause troubles. If we merge a node of surface $surf1$ with a node of $surf2$ or project it on an edge of $surf2$ by changing the coordinates of this node, it may alter the setup achieved by arranging the coordinates of some of the nodes of surface $surf2$. A similar thing as merging of a node of surface $surf1$ with a node or projection on an edge of surface $surf2$ will be done when needed, but without altering the coordinates of any node on any projected surface. This will constitute the second of the two stages for handling this issue. For this, we create a data for every edge on the two surface which will hold all the intersection points the edge has with any edge of a face on the other surface. We go through all the faces of surface $surf1$ and check if an edge, say an edge ed of $surf1$, intersects an edge of surface $surf2$

at one of it's end points. If the intersection point, within some tolerance limit, lies on one of the end points of the edge *ed*, it is said to intersect the other edge at it's own corresponding end point. In this case, this end node is placed in the data of this edge of $surf2$ as an intersection point. So, before checking whether two edges intersect, we check if the edge data of the two edges involved already have a node in common and in this case we don not compute intersection and go on to the other edges of this face. If the two edges are parallel within certain tolerance then we don not try to compute an intersection. Tolerance for merging the nodes of $surf2$ with the nodes of $surf1$ in first stage should be kept sufficiently large as compared to the tolerance set for deciding if the intersection falls at the end point of an edge, so that in second stage two edges of the two surfaces do not intersect at their end points otherwise the two nodes have to be merged and a node will need to be removed from the grid data as these surfaces represent interface surfaces of two grid blocks. If a node of $surf1$ is placed in the edge data of an edge of $surf2$, in the actual grid this edge will be broken at this node. Due to this, the intersection points lying on this edge may not be collinear with the two end points of this edge. As pointed out in the previous section, the face in the rotating grid block which have an edge on the planar interface surface may also need to be broken since the intersection nodes to be added to the face may not be coplanar with this face.

After completing both the stages, for breaking a face of an interface surface as mentioned in the previous section, before computing an intersection point for two edges it is always checked if the edge data for these two edges already have a node in common. Once all the nodes participating in forming a polygonal overlap region are found, a new face is formed and orientation of this new face is borrowed from the parent face which was broken to create this new face.

3.3 Preliminary Estimation of the Updated 3-D Grid Data-Size

When the grid is being updated after rotation, we need to create a node, face and cell data list of sufficiently big size to hold the final updated grid data. Since before updating, it is still unknown as to how many new nodes and faces will be created and what maximum number of nodes a face or a cell will have in the updated data. So some preliminary estimates are needed and if the data size exceeds these estimates, these arrays need to be resized. As pointed out in the previous section, during the update process the new nodes are introduced on the faces of stationary surfaces which have surfaces of the rotating grid sliding on them. As shown in figure 2.2 and figure 2.3, one of the planar side surfaces is a multiply connected surface due to the shaft of the fan. To find some estimate on the number of new nodes we assume that these surfaces are simply connected by ignoring the circular holes in them.

With the above assumption, suppose in figure 2.3, the $CylSsurf$, $PlaneSsurf1$ and $PlaneSruf2$ have a total of NE number of triangular faces. All these faces are broken by the overlapping faces of the corresponding rotating grid surfaces. New nodes may appear inside these faces, or on the edge of these faces. Assuming an ideal case that a node on these surfaces has six triangular faces connected to it, and a face on the stationary surface has approximately four intersection points on each edge due to overlapping faces of rotating surface. Consider figure 3.10, which shows a stationary surface face sf (shaded), overlapped by nine faces of a rotating surface and face sf has four intersection points on each edge and contains a node in its interior. With these assumptions and ideal case considerations, there will be a total of $6\,NE$ number of intersection nodes on the three relevant surfaces of $grid1$. This can be easily proved. Since, by assumption, all these surfaces of $grid1$ are simply connected, when put together they form a closed surface. So on this closed surface with only triangular faces, each face has three face-neighbors. With four intersection nodes per-edge, there are twelve intersection nodes per-face and since each edge is shared by two faces, all of the $12\,NE$ intersection nodes have been

counted twice. Hence the actual number of total intersection nodes is $12\,NE/2$. This observation can be generalized further to incorporate an arbitrary topology for surface faces. Consider a closed-simply-connected surface in 3-D. Let $NE(n)$ be the number of surface faces which have n nodes or edges. If every edge of face with n edges has $\phi(n)$ number of intersection points between the two nodes of that edge, the total number of intersection nodes TN on this surface, based on the earlier argument, can be roughly estimated as,

$$TN = \frac{1}{2}(\sum_{n=3} n\phi(n)\,NE(n)) \tag{3.6}$$

The above equation 3.6, implies that the sum,$\sum_{n=3} n\phi(n)\,NE(n)$ is an even number. Since the assumed number $\phi(n)$ can be chosen arbitrarily and independent of n and $NE(n)$, the sum $\sum_{n=3} n\,NE(n)$ should be an even number. If there are only triangular, quadrilateral, pentagonal and hexagonal faces then, to have an odd number of triangles an odd number of pentagonal faces also have to be there.

Since in the present case, all the faces on these surfaces are triangular so the total number of such intersection points would be $\frac{1}{2}3\phi(3)NE(3)$. With our assumption of $\phi(3) = 4$ and denoting $NE(3)$ by NE, the total intersection points from equation 3.6 are $6\,NE$. With the assumption of each face of stationary surface having one node of rotating surface in the interior, the total number of new nodes added to the grid would be $6\,NE + NSurfnode$, where $NSurfnode$ is the total number of nodes on all the rotating surfaces. This number can be used as a preliminary guess to estimate the size of final node data.

To estimate number of new faces generated, consider figure 3.11, which shows a face Rf on the $CylRsurf$ of $grid2$ and the cell in this grid connected to this face. This cell has three faces $f1$, $f2$ and $f3$ inside $grid2$. Similar to the previous assumption, assume that when surface $CylRsurf$ rotated, the face Rf will be overlapped by the faces of corresponding surface of stationary grid and that it will have four intersection points on each edge between the two edge-nodes. The faces inside $grid2$, connected to an edge of

surface $CylRsurf$ are broken to form new triangular faces. The inside faces of $grid2$ which are connected only to an edge of planar surfaces are not broken. So, if the face Rf has four intersection points per-edge then total number of new faces generated inside $grid2$, because of this face Rf, would be twelve as shown in figure 3.12. Besides this, with our assumption, every face on $grid1$ surface has been broken to created eight extra faces as shown in figure 3.10. So if the total number of triangular faces on the original $CylRsurf$ are $NfCylSurf$ and total number of faces on all the original outer surfaces of $grid2$ are $NfSurf$, then, based on these assumptions, the number of newly created faces can be estimated to be around $12\,NfCylSurf + 8\,NfSurf$. The estimation, based on the current assumptions, of maximum number of nodes per-face and per-cell will not be very reliable . But some rough guess can be made using the figure 3.10 for maximum number of nodes for a face to be seven and seventeen for a cell. For the present application, all the faces connected to an edge of these surface grids in both $grid1$ and $grid2$ are triangular and the cells connected to these surfaces are tetrahedrons in both the grids and the total number of cells do not get affected during the updating process.

CHAPTER IV

GENERALIZED SOLVER

For the present application for simulating rotating machines, individual grids made around the rotating components are assembled with the stationary grid as described in chapter II. After this process of assembling, a single block grid is obtained. When the rotating components undergo rotation, this grid is appropriately updated. The resulting grid may have faces and cells with arbitrary number of nodes. For CFD simulation on this updated grid, a generalized flow solver can be used, which has been briefly described in this chapter.

The integral form of non-dimensionalized Navier-Stokes equations is taken as the governing equation [10].

$$\int_{\Omega} \frac{\partial Q}{\partial t} d\Omega + \oint_{d\Omega} F(Q) \cdot \vec{n} ds = \int_{d\Omega} F^{v}(Q) \cdot \vec{n} ds \tag{4.1}$$

where Q is the conserved variable vector, $F(Q)$ is the inviscid flux vector, $F^{v}(Q)$ is the viscous flux vector, and $\vec{n}$ is the outward pointing unit normal to the control volume Ω. The non-dimensionalizations of the above equations are based on the freestream conditions. The velocity components are non-dimensionalized with respect to the total freestream velocity. To simulate the effect of turbulence, the Spalart-Allmaras one-equation model [18] can be used. The implementation details for this may be found in [10].

4.1 Data Structure

A cell-face based data structured is used to alleviate the problem with having cells with arbitrary shape and number of nodes. In the cell-face based data structure, the nodes that form the cell-face and cell numbers on either side of the cell-face is stored. For a boundary face, the adjacent cell number is stored along with an unique identifier for the particular boundary face.

4.2 Finite-Volume Formulation

Finite-volume schemes are well suited for generalized meshes because a typical generalized mesh is an agglomeration of arbitrary polyhedra. For the present work, we have used a cell-centered, finite volume scheme, in which cell-averaged flow variables are stored at the cell center. The discretized form of Eq. 4.1 is written as,

$$\frac{\Delta Q}{\Delta t} V_i = -\sum_{j=1}^{k} F_{ij} \cdot \vec{n} ds_j + \sum_{j=1}^{k} F_{ij}^{v}(Q) \cdot \vec{n} ds_j \tag{4.2}$$

where V_i is the volume of the cell, i and j represent the cells on either side of the cell-face, and F_{ij} and F_{ij}^{v} are the inviscid and viscous numerical fluxes, respectively. Roe's approximate Riemann solver [19] as an exact solution for a linearized Riemann problem, is used to compute the inviscid numerical flux passing through the cell faces

$$F_{ij} = \frac{1}{2}[F(Q_i) + F(Q_j) - |\overline{A}|(Q_j - Q_i)] \tag{4.3}$$

where $|\overline{A}|$ is the Roe-averaged matrix [19].

4.3 Gradient Reconstruction

A linear reconstruction of either the conserved or the primitive variables using a Taylor series expansion is used to obtain a higher-order accuracy in the spatial

discretization. The Taylor series expansion for a function of two variables is written as

$$Q(x,y) = Q(x_i,y_i) + (\nabla Q)_{(x_i,y_i)} \cdot \vec{\Delta r} + O\left(\vec{\Delta r} \cdot \vec{\Delta r}\right) \quad . \tag{4.4}$$

The gradient of the conserved variables at the cell center is estimated using either Gauss' theorem with the control volume taken as the cell itself or using a least square approach. In the approach based on Gauss' theorem, the value of the conserved/primitive variable vector at the nodes is estimated based on a weighted averaging procedure. The weights are taken as the inverse of the distance between the cell-center and the node to which the contribution is added.

In the least square-based gradient estimation, the variation of the flow variable inside a cell is written as a function of the flow variable within the neighboring cells. Referring to Fig. 4.1, the resulting system of equations can be written for cell 0 as

$$\begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \Delta x_3 & \Delta y_3 \\ \Delta x_4 & \Delta y_4 \\ \Delta x_5 & \Delta y_5 \end{bmatrix} \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} = \begin{Bmatrix} f_1 - f_0 \\ f_2 - f_0 \\ f_3 - f_0 \\ f_4 - f_0 \\ f_5 - f_0 \end{Bmatrix} \quad . \tag{4.5}$$

The overdetermined system given in Eq. 4.5 is solved using a least square approach. A modified Gram-Schmidt method can be used to solve the least square problem.

During the reconstruction process, local extrema may be created within the domain. This may produce spurious values in regions where there are sharp jumps in the flow variables such as shocks, contact discontinuities or expansion regions. In order to avoid this, a limiter function is employed and the Taylor's series expansion is modified using

$$Q(x,y) = Q(x_i,y_i) + \phi\,(\nabla Q)_{(x_i,y_i)} \cdot \vec{\Delta r} + O\left(\vec{\Delta r} \cdot \vec{\Delta r}\right). \tag{4.6}$$

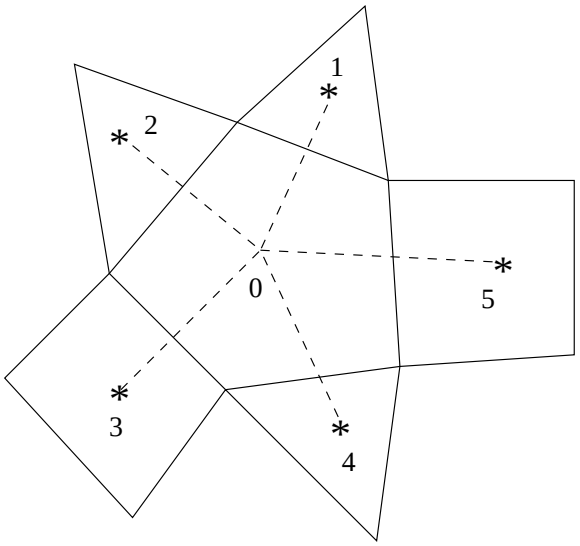

Figure 4.1: Sample mesh for least square approach for gradient estimation

where ϕ is the limiter function and its value is limited between zero and one [10, 20, 21].

4.4 Implicit Scheme

In the case of implicit schemes, the numerical flux crossing the cell face is a function of the conserved variables at the $(n+1)^{th}$ time level. The flux vector has to be linearized before the evaluation of the flux crossing the cell faces. After linearization, the resulting linear system can be written as,

$$\left[\frac{V_i}{\Delta t}I + \sum_{j=1}^{k}\left(\frac{\partial H_{ij}}{\partial Q_i} - \frac{\partial H_{ij}^v}{\partial Q_i}\right)^n\right]\Delta Q_i^n$$
$$+\sum_{j=1}^{k}\left[\left(\frac{\partial H_{ij}}{\partial Q_j} - \frac{\partial H_{ij}^v}{\partial Q_j}\right)^n \Delta Q_j^n\right] = -\Re_i^n \tag{4.7}$$

where

$$H_{ij} = F_{ij}\cdot\vec{n}ds \qquad \text{and} \qquad H_{ij}^v = F_{ij}^v\cdot\vec{n}ds \ . \tag{4.8}$$

The convective flux Jacobian matrices can be estimated using approximate analytical Jacobians, by taking the Roe-averaged matrix $|\overline{A}|$ to be constant, or by a numerical approach [22]. The matrix system resulting from the above equation is solved using a simple relaxation scheme.

4.5 Turbulence Modeling

The simulation of many complex features of flows of practical importance needs to account for the flow's turbulent behavior. The laminar viscosity is usually a function of temperature and is estimated using Sutherland's formula [23]. The turbulent viscosity is a function of the flow and is usually evaluated using an empirical model. The turbulent viscosity can be estimated using the Spalart-Allmaras one-equation turbulence model [18] and for modelling Reynolds stress, the Boussinesq hypothesis [23] can be employed. The Spalart-Allmaras one-equation model encompasses a solution of a second-order partial differential equation for the variable $\overline{v}$. The turbulent kinematic viscosity is estimated from by applying a damping function. The non-dimensional form of the Spalart-Allmaras one equation turbulence model in the vector invariant form, without tripping terms, can be written as

$$\begin{aligned} \frac{\partial \overline{v}}{\partial t} &= \vec{V} \cdot \nabla \overline{v} + C_{b1} \tilde{S} \overline{v} + \frac{1}{\sigma Re_L} \nabla \cdot (v + \overline{v}) \nabla \overline{v} \\ &+ \frac{C_{b2}}{\sigma Re_L} \nabla \overline{v} \cdot \nabla \overline{v} - \frac{C_{\varpi} f_{\varpi}}{Re_L} \left(\frac{v}{d} \right)^2 . \end{aligned} \tag{4.9}$$

The different variables and functions appearing in the above equation can be summarized as

$$\upsilon_t = \overline{\upsilon} f_{\nu 1} \qquad f_{\nu 1} = \frac{\chi^3}{\chi^3 + C_{\nu 1}^3} \qquad \chi = \frac{\overline{\upsilon}}{\upsilon}$$

$$f_{\nu 2} = 1 - \frac{\chi}{1 + \chi f_{\nu 1}} \qquad \tilde{S} = S + \frac{1}{Re_L}\left(\frac{\overline{\upsilon}}{\kappa^2 d^2}\right) f_{\nu 2}$$

$$f_{\varpi} = g\left[\frac{1 + C_{\varpi 3}^6}{g^6 + C_{\varpi 3}^6}\right]^{\frac{1}{6}} \qquad g = r + C_{\varpi 2}\left(r^6 - r\right)$$

$$r = \frac{1}{Re_L}\left(\frac{\overline{\upsilon}}{\tilde{S}\kappa^2 d^2}\right) \qquad C_{\varpi 1} = \frac{C_{b1}}{\kappa^2} + \frac{1 + C_{b2}}{\sigma}$$

$$C_{b1} = 0.1355; \quad \sigma = \frac{2}{3}; \quad C_{b2} = 0.622; \quad \kappa = 0.41; \quad C_{\varpi 2} = 0.3; \quad C_{\varpi 2} = 2.0; \quad C_{\nu 1} = 7.1 \tag{4.10}$$

The details of the implementation of Spalart-Allmaras one equation model can be found in [10].

4.6 Parallelization

The parallelization of the code can be achieved by decomposing the physical domain into different regions. METIS [24] can be used to decompose the domain into different blocks. The graph of the mesh is used to perform the decomposition. Each block is assigned to a different processor and the discrete equations are solved independently in each block. After each iteration, flow variables at the block interfaces are passed between processors. Message Passing Interface (MPI) can used to pass information across the block interfaces.

CHAPTER V

RESULTS

The present methodology for handling the grid after rotation of grid blocks, was tested for single and multiple rotating components. For the testing purpose, an un-ducted SR7 prop-fan with eight blades was used. For testing the integrity of the updated grid data, the updated grid was checked to locate folded faces, negative volumes and gaps in the grid where a gap means that some face of a cell may be missing from the grid data.

Some preliminary tests were conducted to study the performance of this method of grid update when coupled with a CFD solver. Tests were conducted to observe the behavior of free stream when an empty cylinder is being rotated in the grid. Behavior of the CFD solution near the grid block interface was observed for a rotating un-ducted SR7 prop-fan.

5.1 Grid Data Integrity Tests After Rotation

Two test cases were considered for testing the grid data integrity of the updated grid. In the first test case, an un-ducted SR7 prop-fan was rotated by 360 degrees with rotation steps of 0.5 degrees. To test the working of the code for multiple rotating components, two un-ducted SR7 fans were rotated by 360 degrees with rotation steps of 0.5 degrees in opposite directions.

5.1.1 Single Rotating-Grid Block

The figure 5.1 shows an un-ducted SR7 fan with eight blades. The table 5.1 shows the grid data sizes for the two blocks of grids used for testing.

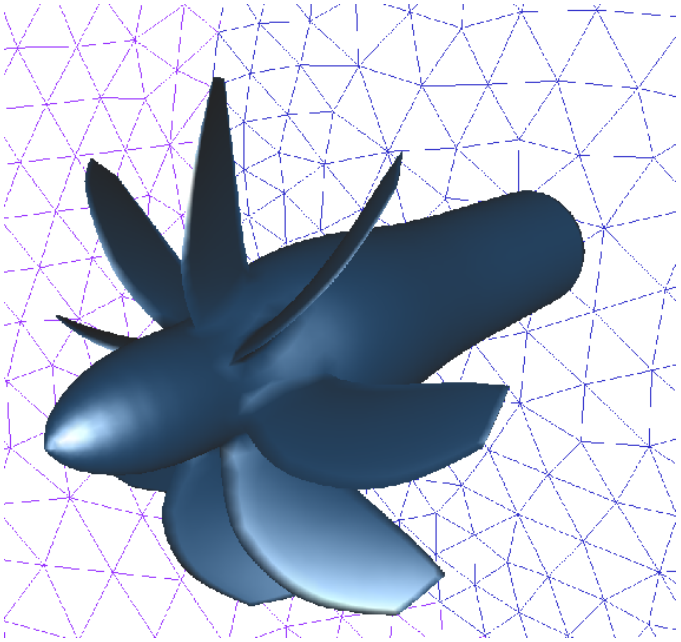

Figure 5.1: An un-ducted SR7 prop-fan with eight blades.

Table 5.1: Grid data size of the blocks.

Block	Number of nodes	Number of faces	Number of cells
Stationary	2022	19840	9509
Rotating	157097	1105708	495133

When the rotating block undergoes a rotational motion, its interface surfaces slide over those of the stationary grid. The figure 5.2 shows the projection of the cylindrical surfaces sliding over each other. The grid shown in blue color in figure 5.2 is from the stationary grid block while the other one (Red in color) belongs to the rotating grid block. This figure shows the faces of a rotating surface, overlapping the faces of a stationary surface, after undergoing a rotation by 60 degrees. Due to rotation and projection, some

faces of the rotating surface get cut as mentioned in chapter II. As seen in this figure, the inverted faces of the rotating surfaces have been flipped depending on the location of the face of the stationary grid which they overlap. The figure 5.3 shows two planar interface surfaces sliding over each other. The surface grid shown in blue color belongs to the stationary grid while the other surface belongs to the rotating grid block.

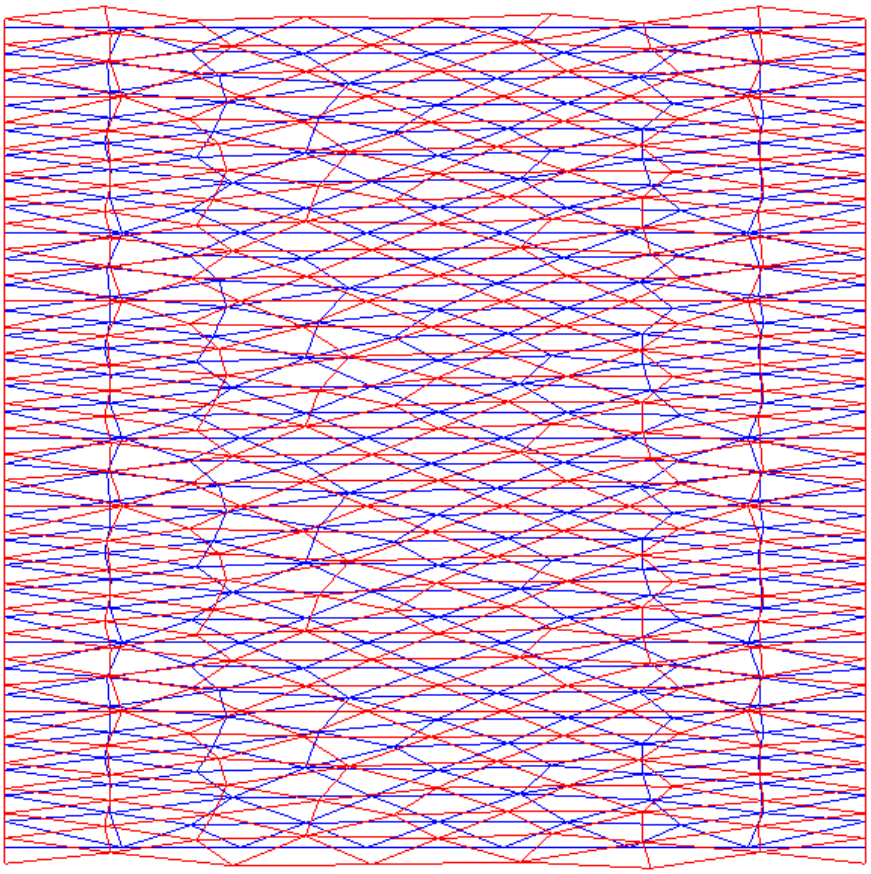

Figure 5.2: Cylindrical surfaces sliding over each other in the projected plane.

The figure 5.4 shows a typical cell on the interface of a stationary grid block. As described in chapter III, the interface faces of the stationary grid are broken due to the overlapping faces of the corresponding rotating surfaces. The non-interface faces of rotating grid block which have an edge on the cylindrical interface, are broken in to triangles while non-interface faces in the rotating grid block which have an edge on the planar interface will be broken only if an intersection point lying on it's edge is not

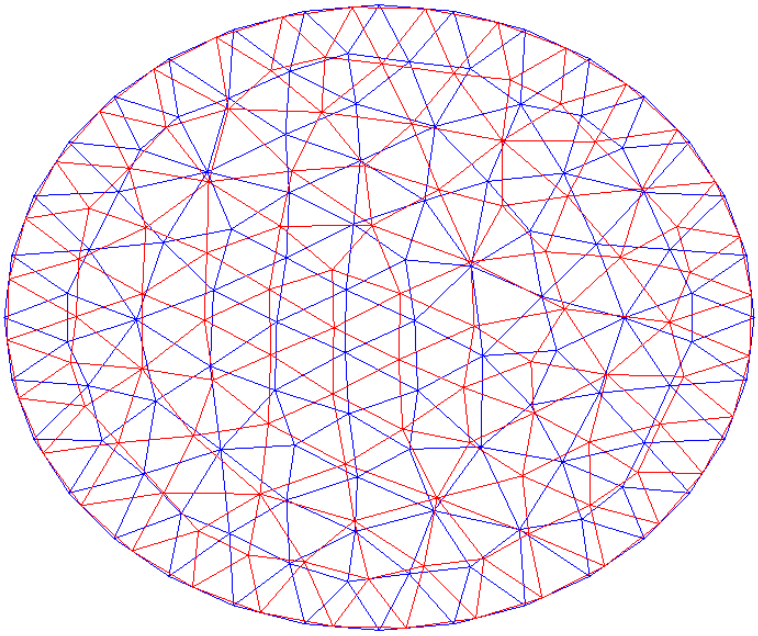

Figure 5.3: Planar interface surfaces sliding over each other.

collinear with the two end nodes of this edge. The figure 5.5 shows a cell in the rotating grid block which has one face on the cylindrical interface and another one on the planar interface. In this case, the faces on the interface are broken along with the faces that have an edge on the cylindrical interface.

The rotating grid block was set to undergo a series of rotations of 0.5 degrees per step, for a complete revolution. Each time the grid is rotated, new nodes and faces may be added on the interfaces. The figure 5.6 and figure 5.7 show the variations of the number of nodes and faces in the updated grid as the rotation progresses. The nature of these variations shown in figures 5.6 and figure 5.7 are highly dependent on the nature of the interface surface grids.

5.1.2 Multiple Rotating-Grid Blocks

For testing the performance of the code for multiple rotating components, two SR7 prop-fans were rotated in opposite directions with rotation steps of 0.5 degrees. The figure 5.8 shows the two geometries used for testing. Table 5.2 gives the grid data sizes

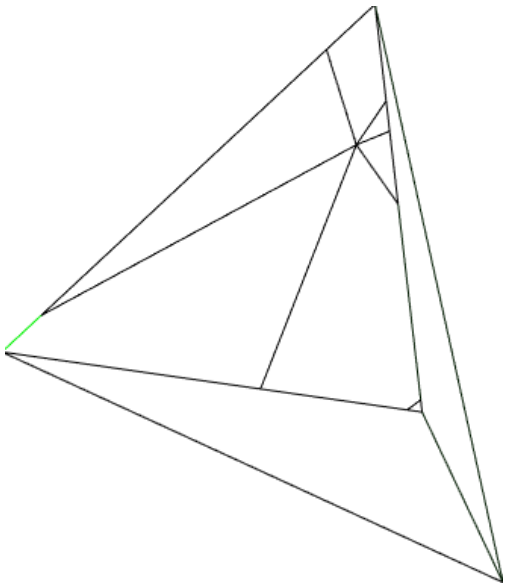

Figure 5.4: An interface cell in the stationary grid.

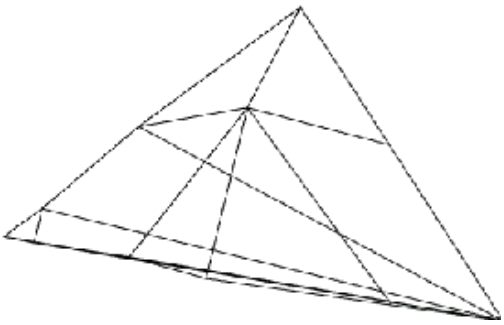

Figure 5.5: A cell with faces on the cylindrical and the planar surfaces of the rotating grid block.

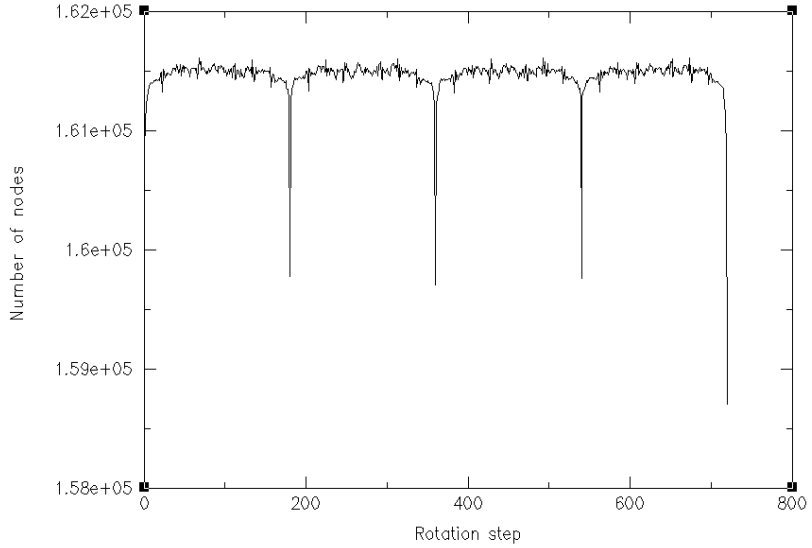

Figure 5.6: Number of nodes in the updated grid after each rotation step of 0.5 degrees for the single rotating body test.

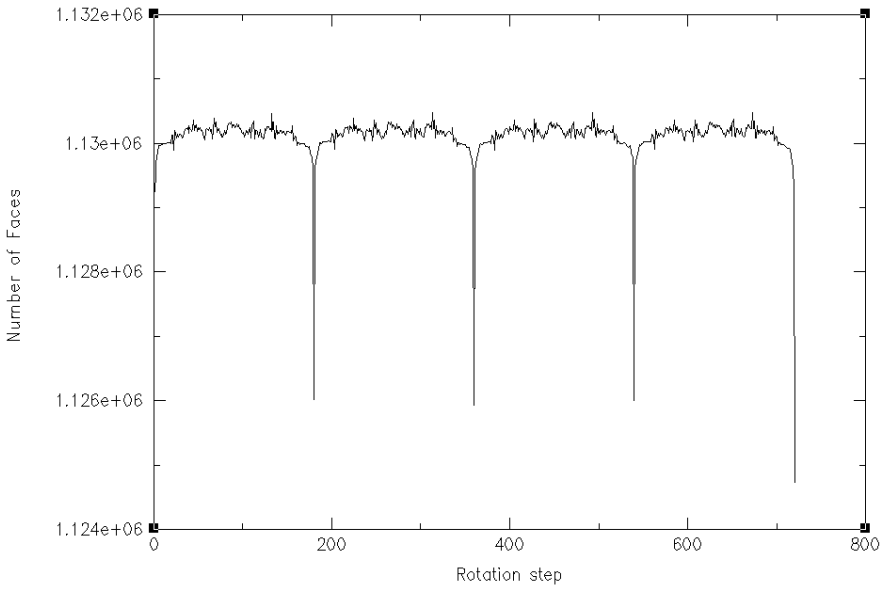

Figure 5.7: Number of faces in the updated grid after each rotation step of 0.5 degrees for the single rotating body test.

of the grid blocks. The rotating grid block, block2, in this table belongs to the upper geometry shown in the figure 5.8 while the rotating grid block, block3, belongs to the lower geometry.

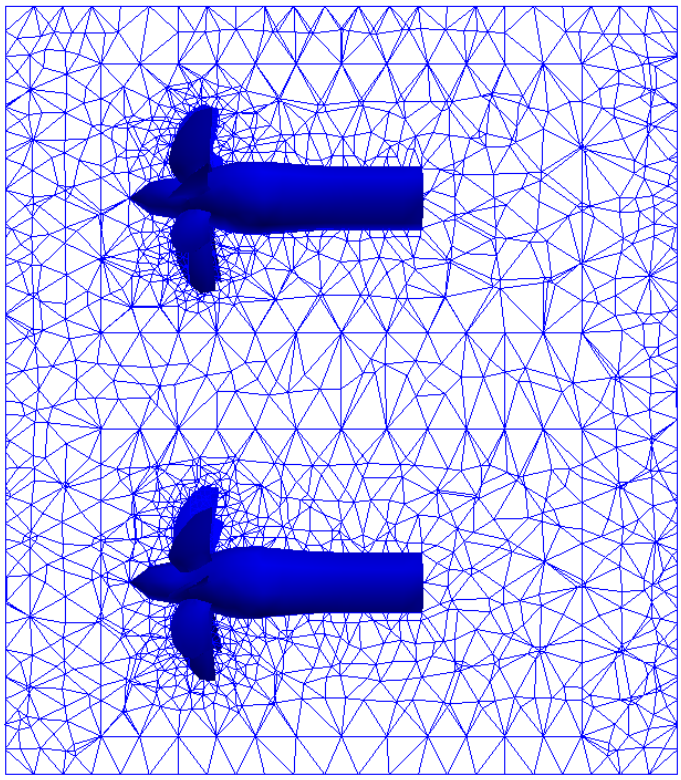

Figure 5.8: Two un-ducted SR7 prop-fans rotating in opposite directions.

Table 5.2: Grid data size multiple rotating body test.

Block	Number of nodes	Number of faces	Number of cells
Stationary grid block1	8858	84387	40077
Rotating grid block2	2654	26438	12666
Rotating grid block3	3068	31011	14920

The figure 5.9 and the figure 5.10 show the variation of number of nodes and the faces in the updated grid data, after each rotation. After each rotation step, the updated grid was tested to locate any negative volumes or gaps.

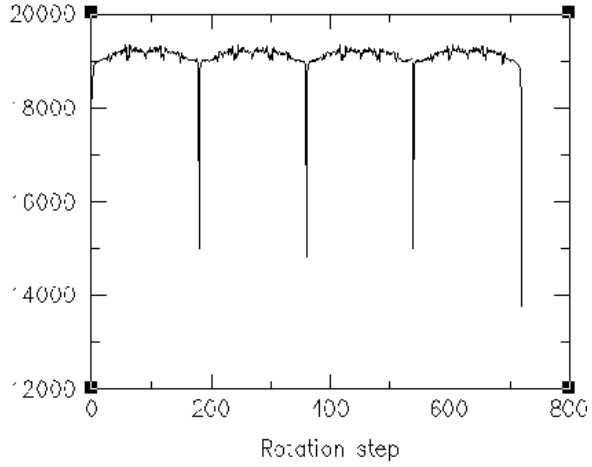

Figure 5.9: Number of nodes in the updated grid after each rotation step of 0.5 degrees for the multiple rotating body test.

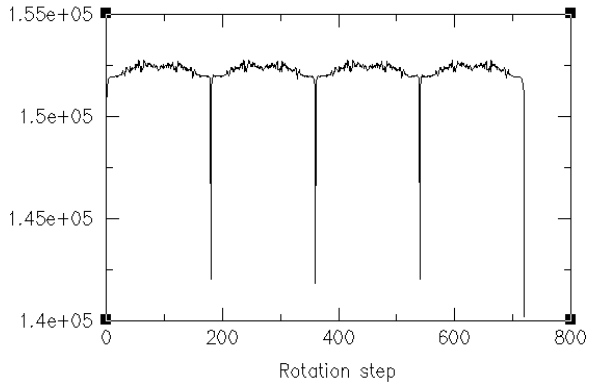

Figure 5.10: Number of faces in the updated grid after each rotation step of 0.5 degrees for the multiple rotating body test.

5.2 CFD Solution Behavior at the Grid-Block Interfaces

With the present methodology for handling the grid to incorporate the relative rotational motions among the grid blocks, some preliminary tests were conducted to test the behavior of the CFD solution near the interfaces. First a shock tube simulation was conducted with an empty cylinder rotating inside the tube with it's axis of rotation along the length of the tube. A free stream test was conducted to see the behavior of the solution at the grid block interfaces and it's effect on the conservation near the interfaces.

5.2.1 Shock Tube Test

The figure 5.11 shows the dimensions of a cylindrical shock tube. An empty cylindrical grid, located in the middle of the shock tube, was rotated continuously to see it's effect on the solution results. For the CFD simulation, a second order accurate scheme was used.

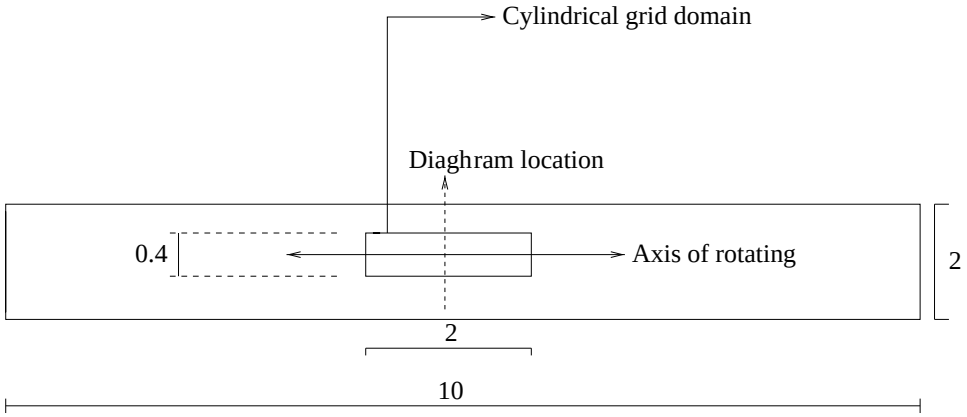

Figure 5.11: Dimensions of the shock tube and the rotating cylindrical geometry.

For comparison of data, the test was also conducted by keeping the cylindrical grid stationary. The results with stationary and non-stationary cylinders were compared with the analytical solutions.

The figure 5.12, figure 5.13 and figure 5.14 show the distribution of pressure, velocity and density, respectively, on a line touching the cylindrical interface, after 1.4196 seconds.

These figures compare the analytical values with the situations when the cylinder is rotating and when it is not rotating. It can be seen that there are some deviations from the analytical values and these are partly due to poor grid resolution in some places, but the results with rotating and non-rotating cylinders match closely.

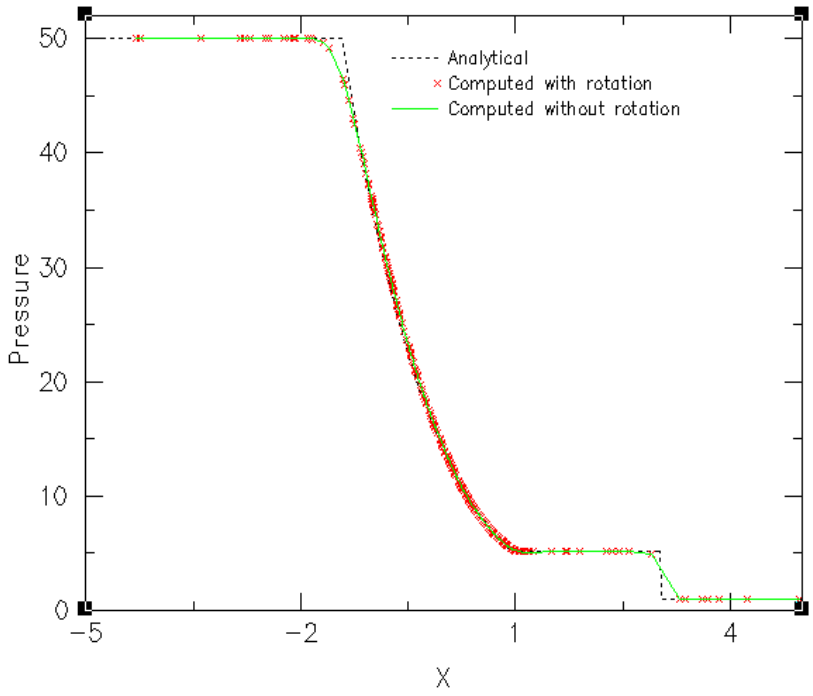

Figure 5.12: Comparison of analytical and computed pressure distribution in the shock-tube after 1.4196 seconds.

5.2.2 Free Stream Test Case

For this test case, a cylindrical rotating grid block was created which had no solid body inside it. This block was rotated to see if the free stream is preserved when the CFD solver is coupled with the present method of grid updating. The table 5.3 shows the grid sizes for the stationary grid and the rotating empty cylindrical grid block.

The solver was run for three complete revolutions of the cylinder with a CFL of 100 and free stream mach number of 0.7. The table 5.4 shows the range of the variations of the flow variables within the domain, after three revolutions. The stationary and the

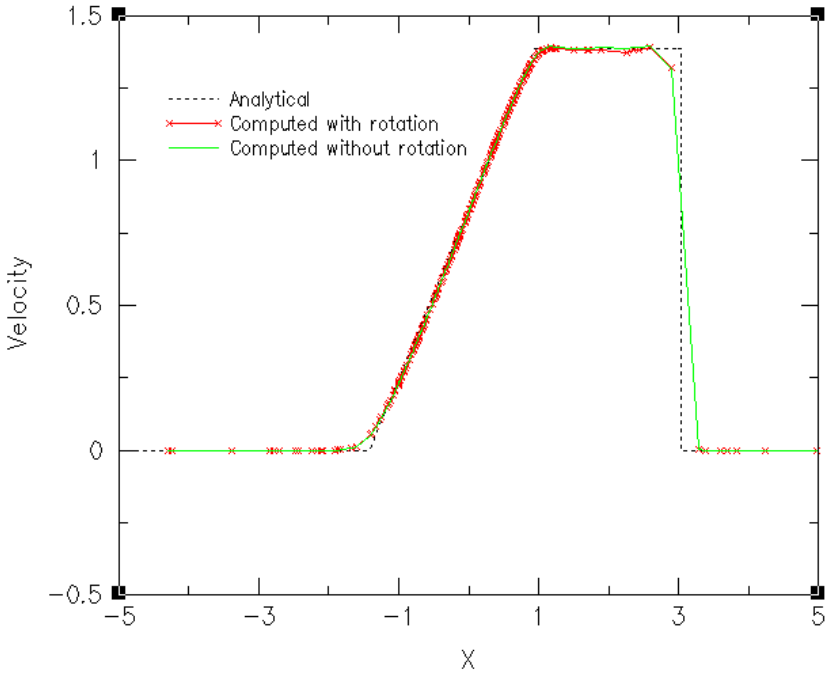

Figure 5.13: Comparison of analytical and computed velocity distribution in the shock-tube after 1.4196 seconds.

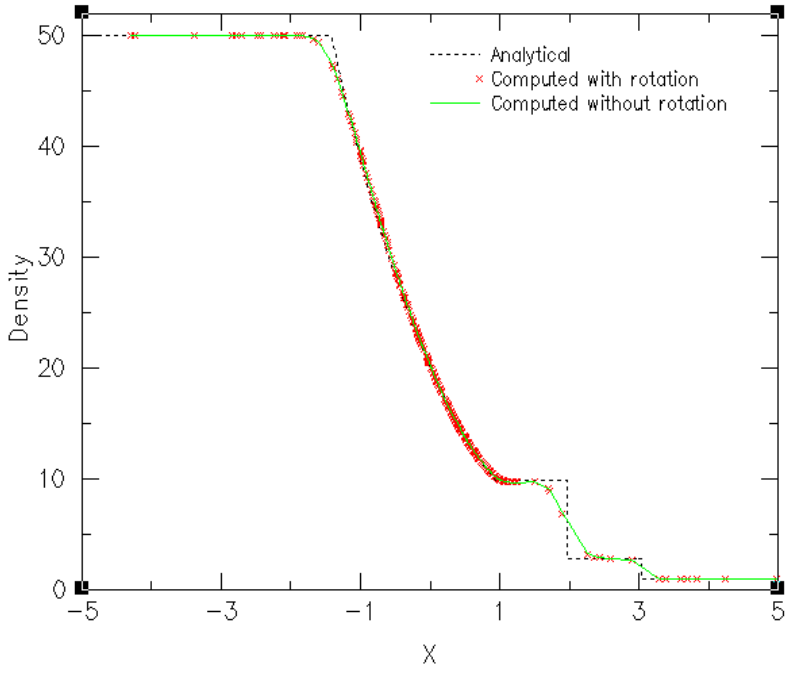

Figure 5.14: Comparison of analytical and computed density distribution in the shock-tube after 1.4196 seconds.

Table 5.3: Grid data size of the blocks for free-stream test grid.

Block	Number of nodes	Number of faces	Number of cells
Stationary	4289	41318	19677
Rotating	1954	18176	8611

rotating grid blocks were aligned in the X-axis direction and the inlet flow was in the direction of positive X-axis. The rotation of the grid block was performed about the X-axis. The variables, ρ, u, v, w and P represent the density, velocity components along the X-axis, Y-axis, Z-axis and the pressure, respectively.

Table 5.4: Grid data size of the blocks for free-stream test grid.

Flow variable	Minimum	Maximum
ρ	1.223351	1.226932
u	238.001984	238.29499
v	-0.201275	0.144972
w	-0.185158	0.15192
P	101206.523	101374.773

The figure 5.15 shows the pressure distribution on the outer surface of the domain. This figure indicates some disturbances on the outer boundaries of the domain. The figure 5.16 shows a cutting plane along the axis of the rotating cylinder and the interface is visible on this cutting plane. The figure 5.16 shows the pressure distribution on this cutting plane with maximum possible variation indicated in the table 5.4. At present, reasons for the disturbances seen in the figure 5.17 are not well understood. In future, a detailed investigation is needed to explain these disturbances on the interfaces.

The figure 5.18 shows the variation in the number of faces, nodes and cells in the updated grid data following each rotation. Each complete revolution corresponds to about 645 rotation steps.

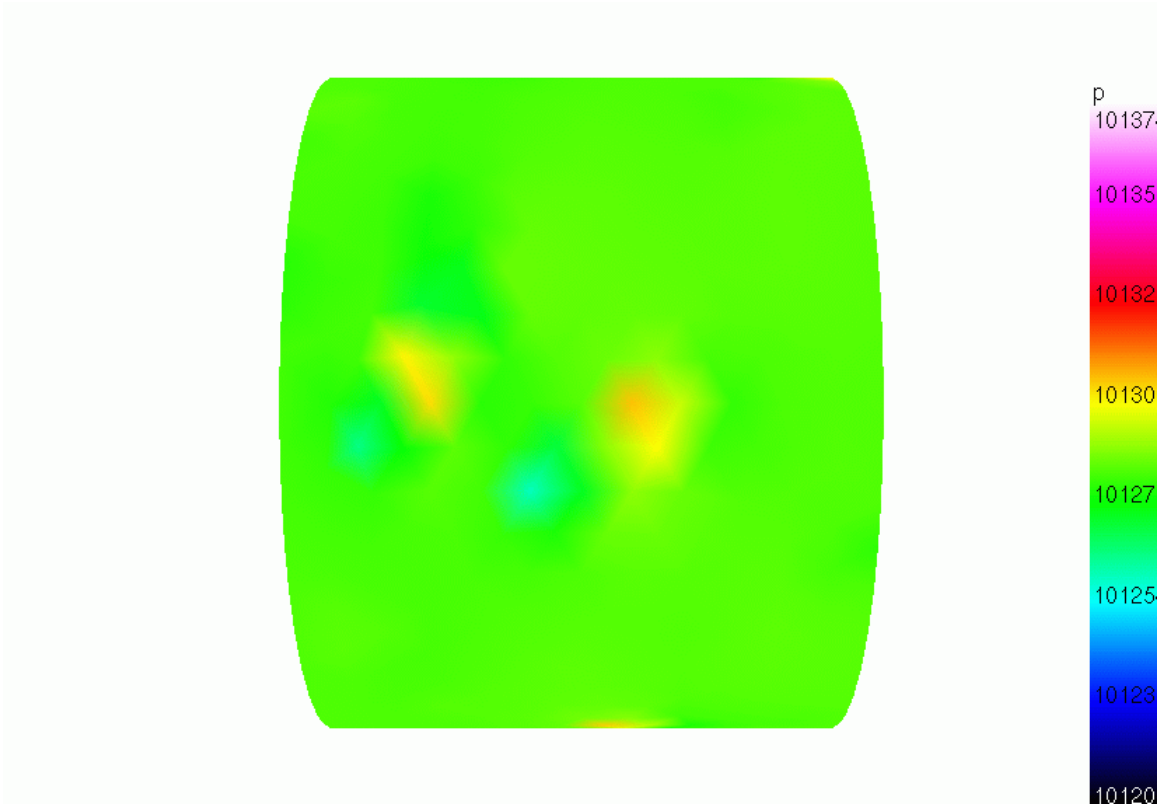

Figure 5.15: Pressure distribution on the outer boundary of the domain for the free stream test grid.

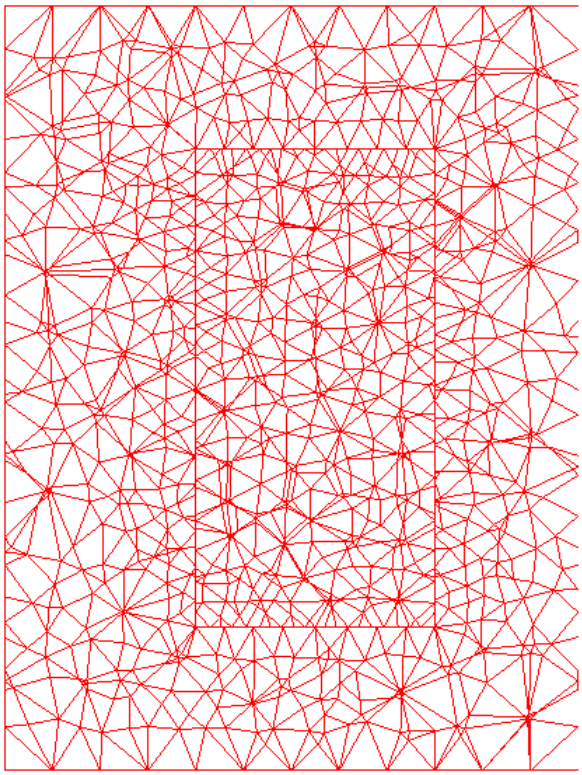

Figure 5.16: A cutting plane showing the interface of the grid blocks for the free stream test grid.

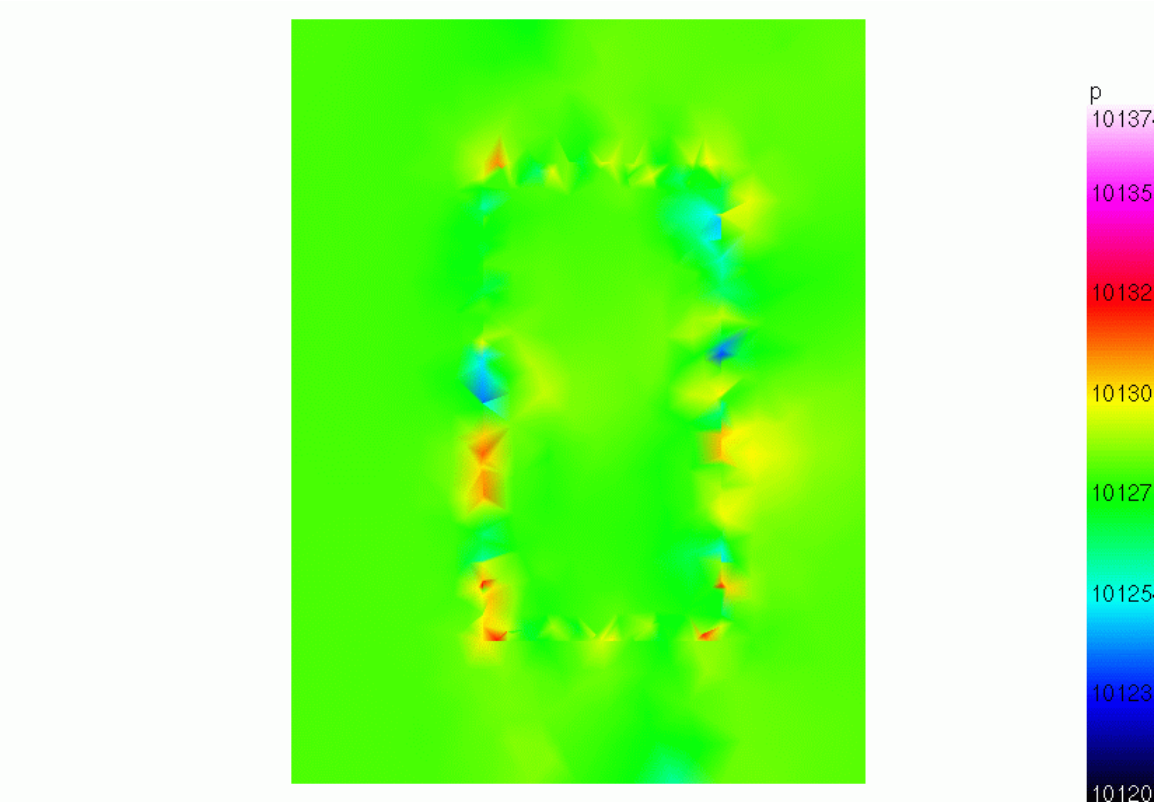

Figure 5.17: A cutting plane showing the pressure distribution in the domain for the free stream test.

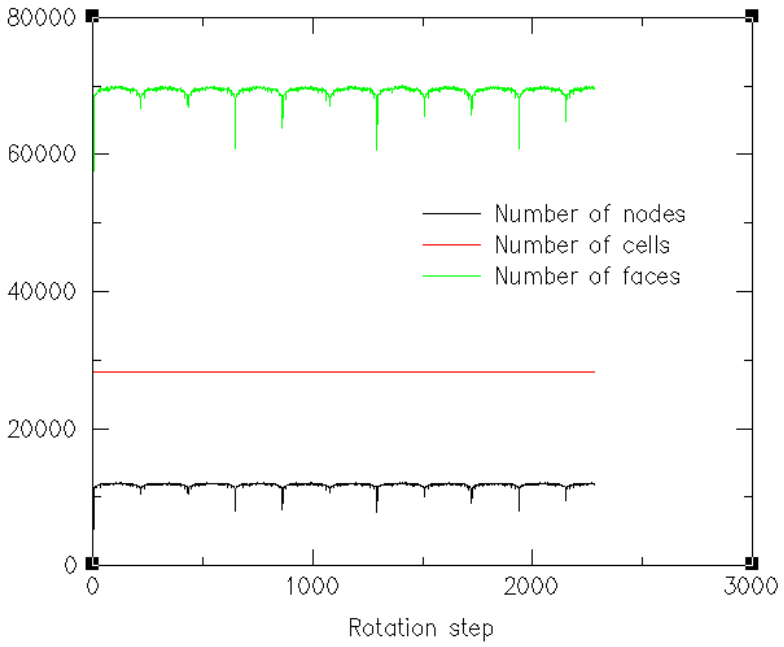

Figure 5.18: Variation in number of nodes, faces and cells with rotation for the free stream test grid.

5.2.3 Test on an Un-ducted SR7 Prop-Fan Geometry

A generalized solver was run with a rotating un-ducted SR7 prop-fan with an inlet Mach number of 0.7 and an advance ratio of 3.06 with the CFL number set to 50. The table 5.5 gives the grid data for both the grid blocks. The figure 5.19 shows a cutting plane where the interface between the two blocks can be seed and the figure 5.20 shows the pressure distribution on this cutting plane after 250 solution iterations which corresponded to a rotation by 1.26584 degrees. As seen in the figure, 5.20, there are no visible disturbances on the interfaces, however more exhaustive testings with better visualization tools is need to study the effect on the interfaces.

Table 5.5: Grid data size of the blocks for un-ducted SR7 prop-fan geometry.

Block	Number of nodes	Number of faces	Number of cells
Stationary	2022	19840	9509
Rotating	157097	1105708	495133

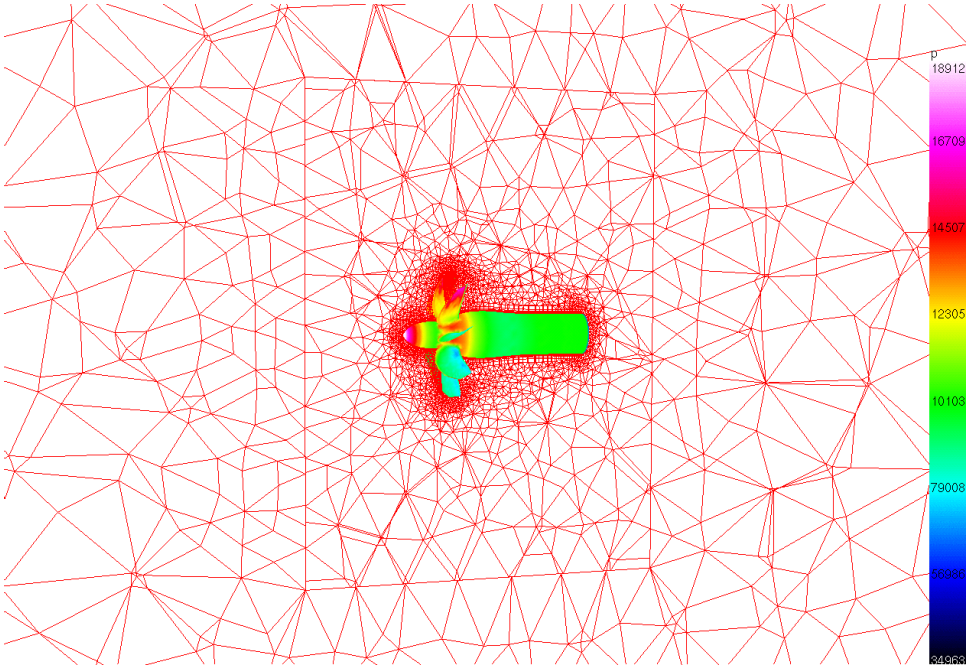

Figure 5.19: A cutting plane showing the interface around the geometry.

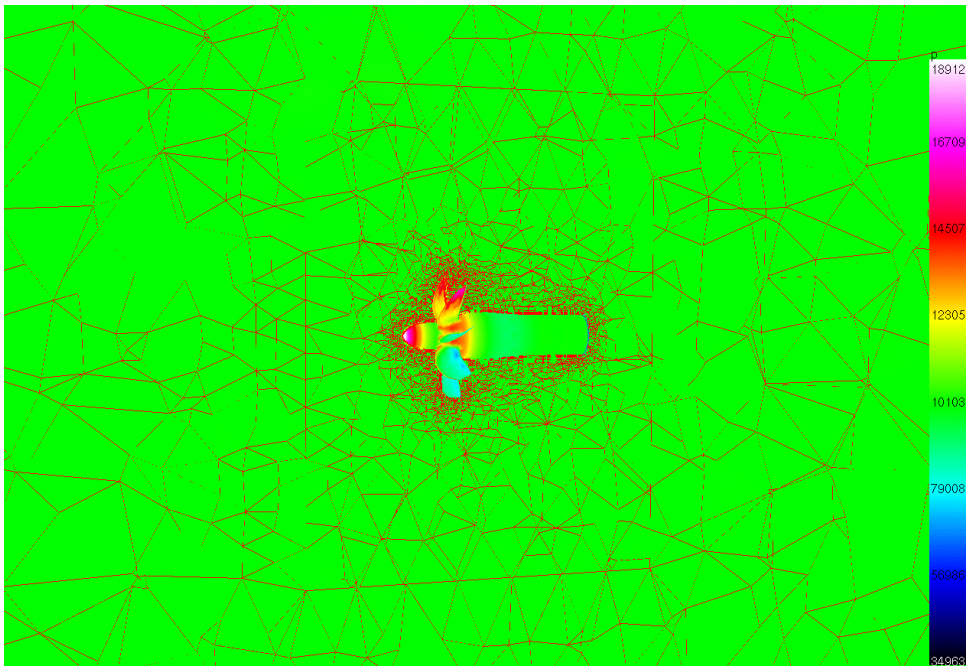

Figure 5.20: Pressure distribution around and across the interface and the geometry, after 250 solution iterations corresponding to a rotation by 1.26584 degrees.

CHAPTER VI

SUMMARY AND CONCLUSIONS

A hybrid grid based technique was developed for handling the grid on block interfaces in the CFD simulations of rotating machineries. The current methodology was developed for cells with any number of nodes but the present work is limited to grids which have only tetrahedral elements on the interfaces. The methodology was tested for a single and two rotating bodies to examine the grid-data integrity of the single block grid created after each rotation. The application developed during this work is capable of handling multiple rotating components with arbitrarily aligned axes of rotation. The method for grid updating, after each rotation, was found to be working as expected for the test cases considered. Some preliminary tests were conducted to study the behavior of the CFD solution on the interface surfaces.

Some solution disturbances on the interfaces were noticed when the present method for grid update was coupled with a CFD solver. An extensive validation with a CFD solver is needed in future to investigate the sources of these disturbances and a parallel version of the present work is also due in future to enable efficient testings on large grid data sets.

REFERENCES

[1] J. M. Janus, *Advanced 3-D CFD Algorithm for Turbomachinary.* PhD thesis, Mississippi State University, May 1989.

[2] J. P. Chen, *Unsteady Three-Dimensional Thin Layer Navier-Stokes Solutions for Turbomachinary in Transonic Flow.* PhD thesis, Mississippi State University, December 1991.

[3] J. F. Thompson, "A general three-dimensional elliptic grid generation system on a composite block structure," *Comp. Methods Appl. Mech. Engrg.*, vol. 64, pp. 377–411, 1987.

[4] N. P. Weatherill, "A method for generating irregular computaional grids in multiply connected planar domains," *Int. J. Num. Meth. Fluids*, vol. 8, pp. 181–197, 1988.

[5] D. L. Marcum and N. P. Weatherill, "Unstructure grid generation using iterative point insertions and local reconnection," *AIAA J.*, vol. 33, pp. 1619–1625, 1995.

[6] J. A. Shaw, "Hybrid grids," in *Handbook of Grid Generation* (J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds.), CRC Press, Boca Raton, FL, 1998.

[7] Y. Kallinderis, "Hybrid grids and their applications," in *Handbook of Grid Generation* (J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds.), CRC Press, Boca Raton, FL, 1998.

[8] R. Noack and J. Steinbrenner, "A three-dimensional hybrid grid generation technique," *AIAA paper, 95-1684CP, 12'th AIAA Computational Fluid Dynamics Conference*, San Diego, CA, June 1995.

[9] D. S. Thompson, S. Chalasani, and B. K. Soni, "Generations of generalized grids by extrusion from surface meshes of arbitrary topology," in *Proceedings of the 7'th International Conference on Numerical Grid Generation in Computational Field Simulations* (M. Cross, P. R. Eiseman, J. Hauser, B. K. Soni, and J. F. Thompson, eds.), pp. 1061–1070, International Society for Grid Generation, Mississippi State, MS, September 2000.

[10] R. P. Koomullil and B. K. Soni, "Generalized grid techniques in computational field simulation," in *Proceedings of the 6'th International Conference on Numerical Grid Generation in Computational Field Simulations* (M. Cross, P. R. Eiseman, J. Hauser, B. K. Soni, and J. F. Thompson, eds.), pp. 521–531, International Society for Grid Generation, Mississippi State, MS, August 1998.

[11] A. Khawaja and Y. Kallinderis, "Hybrid grid generation for turbomachinery and aerospace applications," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 145–166, September 2000.

[12] D. Lindquist and M. Giles, "Generation and use of unstructured grids for turbomachinery," in *Proceedings of the Computational Fluid Dynamics Symposium on Aeropropulsion*, NASA CP-10045, 1990.

[13] K. G. U. Ghia and R. Ramamurti, "Hybrid c-h grids for turbomachinery cascades," in *Advances in Grid Generation* (K. Ghia and U. Ghia, eds.), pp. 143–150, ASME Publication , New York, 1983.

[14] S. S. Neerarambam, *A Parallel Turbomachinary Flow Solver and Application to Distortion-Induced Compressor Rotor Stall.* PhD thesis, Mississippi State University, May 1998.

[15] Z. Jaworski, M. L. Wyszynski, R. S. Badham, K. N. Dyster, I. P. T. Moore, A. W. Nienow, N. G. Ozcan-Taskin, and J. McKemmie, "Sliding mesh CFD flow simulation for stirred tanks," une 1995.

[16] M. Rai, "A relaxation approach to patched-grid calculations with the Euler equations," *Journal of Computational Physics*, vol. 66, pp. 99–131, December 1986.

[17] A. R. Ghosh, "Solutions of three-dimensional thin layer Navier-Stokes equations in rotating coordinates for flow through turbomachnary," Master's thesis, Mississippi State University, December 1996.

[18] P. R. Spalart and S. R. Allmaras, "A one-equation turbulence model for aerodynamic flows," *AIAA Paper 92-0439, AIAA 30th Aerospace Meeting*, Reno, NV, January 1992.

[19] P. L. Roe, "Approximate Reimann solvers, parametric vectors and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[20] T. J. Barth and D. C. Jesperson, "The design and application of upwind schemes on unstructured meshes," *AIAA Paper 89-0366, AIAA 27th Aerospace Meeting*, Reno, NV, January 1989.

[21] V. Venkatakrishnan, "On the accuracy of limiters and convergence to steady state solutions," *AIAA Paper 93-0880, AIAA 31st Aerospace Meeting*, Reno, NV, January 1993.

[22] D. L. Whitfield and L. Taylor, "Discretized Newton-relaxation solutions of high resolution flux-difference schemes," *AIAA Paper 91-1539, AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, HI, June 1991.

[23] Z. U. A. Warsi, *Fluid Dynamics: Theoretical and Computational Approaches.* CRC Press, Inc., 1993.

[24] G. Karypis and V. Kumar, *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices , Version 4.0.* University of Minnesota, 1998.