National Technical University of Athens

2D FLOW SIMULATION WITH CHIMERA GRIDS

by

Theodoros Dimas

A thesis submitted in partial fulfillment of the requirements for the degree of

Computational Mechanics

November 2020

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

ABSTRACT

2D Flow Simulation with Chimera grids

by Theodoros Dimas

Supervisor: Professor Spyridon Voutsinas
Department of Mechanical Engineering

In this study, a zonal interface generation algorithm was developed and implemented in a cell-centered, finite volume solver in order to simulate a flow over an airfoil which is embedded in a Cartesian grid. Chimera or overset grids acquire the flexibility of having a simple topology mesh over complex individual geometric features and then combined, they create a global simulation.

While the classic Chimera grid approach requires interpolations of the values in the overlapping regions of the grids, in this study an alternative method was followed: The two meshes are patched in their common boundary creating one grid, which takes into consideration the advantages of the patched grid scheme and the overset grid scheme and therefore a new zonal interface boundary scheme is created. The biggest advantage of this method is that it ensures global conservation. The algorithm can unite structured-structured grids and structured-unstructured as well.

Furthermore, the zonal interface generation algorithm was created in a way that the expansion of the algorithm to moving Chimera grids is feasible. Finally, the algorithm was successfully implemented in the Eulerian Solver MaPFlow and was tested for a steady state 2D flow simulation of a mesh over an airfoil which is embedded in a Cartesian grid.

Table of contents

Table of Figures

0. Introduction

To handle complex geometries and flow physics in Computational Fluid Dynamics analysis, a multi zonal approach is highly favored over the single-zonal. In the multizonal approach, the physical flow domain is divided into several geometrically simple sub-domains (zones), in which independent meshes can be easily generated. The multizonal approach has also the advantage that the mesh in a certain zone of the flow domain can be easily refined if necessary. Furthermore, multizonal algorithms can be easily adapted to take advantage of multi-processing parallel computers.

There are two different multizonal algorithms depending on whether the zonal boundaries exactly match or arbitrarily intersect each other. The former is called the Patched grid approach and the second is called Overset (or Overlapped) grid approach.

In the first chapter, the differences between the classic Chimera grid approach and the fully conservative Chimera method are shown. In the second chapter, the methodology of applying the fully conservative Chimera grid to moving grids is explained. In chapter 3 the algorithm which was developed for composing the Chimera grids into one is described. In chapter 4 the governing equations, as well as the discretization of the Eulerian solver in which the above algorithm was implemented, is shown. In chapter 5, the cases in which the zonal interface generation algorithm was tested are illustrated and a 2D Flow steady-state flow simulation for an airfoil NACA 0012 at 10^o angle of attack embedded in a Cartesian grid of length of 25 chords, is presented. In chapter 6, the conclusions derived, are presented and some potential future work based on this study is suggested.

Chapter 1

Chimera Grid Approach

The overset approach was first developed by Benek, Steger, and Dougherty [2] as a method of accommodating complex geometry consisting of multiple bodies. They referred to their scheme as “Chimera” after the mythological Greek character composed of many different animal features. The motivation behind the Chimera schemes was that relatively simple grid topologies could be used around individual geometric features and then combined to create a global simulation. The result is a system of patched grids, which together cover the problem domain. Relative motion between various bodies was handled smoothly since grid components could slide past one another.

The major difficulty in assembling a practical overset scheme is the efficient and accurate implementation of domain connectivity. Domain connectivity provides each individual grid in the domain with information about the surrounding flow, coupling the grid system. Conventionally, domain connectivity has been accomplished by linear interpolation along inter-grid boundaries, as originally proposed by Benek, Steger, and Dougherty. Efficient algorithms have been developed to identify donor and recipient nodes for use in interpolation. Many researchers have pointed out that formal conservation is lost in the global sense when interpolation is used.

Basic concept of the Chimera Grid method is the following:

- ✓ The embedded mesh introduces an artificial boundary or “hole” into the mesh in which it is embedded.
- ✓ The holes are not included in the solution of the flow in the major mesh.
- ✓ Communication among the grids is restricted to the transfer of the boundary data.
- ✓ Flow is solved separately in each mesh.

More specifically, after deciding on the data structure and reading the data from the meshes we obtain one major mesh and one or more minor meshes (meshes which are embedded in the major mesh)(fig.1) . Afterwards, a cutting hole process is constructed which creates holes in the points of the major mesh which are overlapped by the minor mesh(es). However, it should be noted that with the hole creation, the overlapping region is reduced not erased because in this region the interpolations take place. The fringe points, which are the points of the major mesh that are next to the hole points, become the boundary points of the major mesh (fig.2).

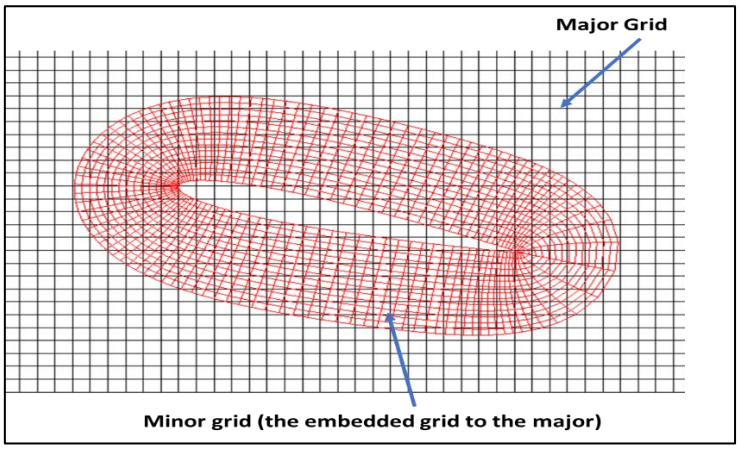


Figure 1 Major and minor mesh

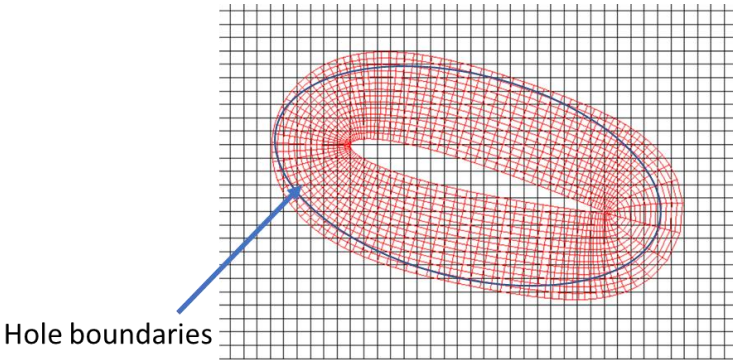


Figure 2 Boundary hole in the major mesh

The solution values on the outer boundary of the embedded grid and the corresponding hole boundaries are determined by interpolation (fig.3). For example, a point on the boundary of the embedded grid is located and the points in the major grid, which are the closest, are searched in order to interpolate the solution from this point to the outer boundary point of the embedded mesh. The same “nearest neighbor” procedure is followed for a hole boundary to get the solution value from the nearest point of the embedded mesh. Last but not least, the extents of the overlap must be sufficient enough to assure that such points exist and large enough so that the boundaries forming the overlapping region have a little influence on the solution in the regions of the interpolation.

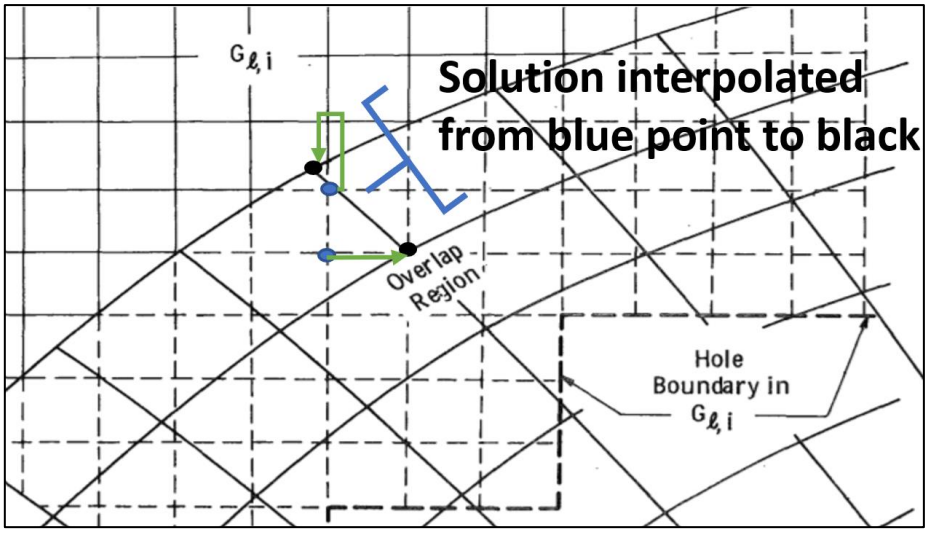


Figure 3 Example of the interpolation of the solution data between the boundary of the minor grid and the nearest point of the majo grid

With regards to boundary conditions Atta proposed a Neumann boundary condition in the hole boundaries of the major grid and a Dirichlet boundary condition in the boundary of the minor mesh. More specifically in the outer boundary of the minor mesh the flow variable $\Phi(\rho, u,v,w,p)$ is specified while in the hole boundary the normal component of Φ is specified (fig.4).

Atta found that:

- Increasing overlap area decreases number of iterations for convergence but increases computing effort.
- Cartesian grid (major grid) should overlap 15-20% of the curvilinear grid (minor grid).

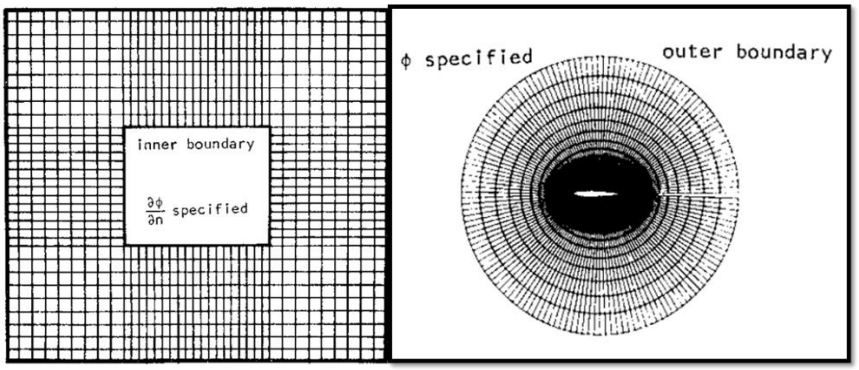


Figure 4 Boundary conditions on the outer boundary of the minor mesh and inner boundary of the major mesh

In general, a Chimera grid approach algorithm consists of the following:

- Compute flow field in minor grid for a number of iterations.
- After that, we interpolate the normal velocity component in the boundaries of the major grid (hole boundaries).
- Compute flow field in major grid for several iterations.
- Interpolation of the flow variables in the outer boundary of minor grid.

1.1 Fully conservative Chimera Grid Method

Since interpolations are used between grids to exchange information in the Chimera Grid scheme, the overset grid approach is intrinsically non-conservative. Wang et al [3 4] proposed a fully conservative chimera method by taking into consideration the advantages of the patched grid scheme and of the overset grid scheme and therefore created a new zonal interface boundary scheme.

The method is designed for the overlapped grids, but it eliminates all the disadvantages listed above. The new approach is fully conservative and has found to be stable and accurate under severe test conditions such as strong discontinuities passing through zonal boundaries. Moreover, the new method can be easily implemented into any multi-zonal finite-- volume CFD code. A high order, cell-centered finite volume TVD scheme with Roe Riemann solver is used to discretize the governing equation.

1.1.1 Basic concept of the method

Consider the following conservation laws written in integral form:

$$\int_v \frac{\partial Q}{\partial t} dV + \oint_s F \, dS = 0. \qquad (1.1)$$

The vectors F and Q are given by

$$F = \begin{Bmatrix} \rho v_n \\ \rho u v_n + p n_x \\ \rho v v_n + p n_y \\ \rho w v_n + p n_z \\ (e+p) v_n \end{Bmatrix} \qquad Q = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{Bmatrix} \qquad (1.2)$$

where ρ, p, e and u, v, w are density, pressure, total energy, and Cartesian velocity components respectively, $\vec{n} = (n_x, n_y n_z)$ is the unit normal of the surface, S, and $u_n = \vec{u} \cdot \vec{n}$ and Pressure p.

In the finite-volume approach the physical domain is further divided into small **cells** called control volumes. Let a semi-discrete numerical scheme for a control volume ΔV in V be written as

$$\frac{\partial Q}{\partial t} \Delta V = -\sum_f F_f dS_f \qquad (1.3)$$

where Q now represents the cell averaged conservative variables, and the summation index f denotes all surrounding faces of ΔV. Scheme (3) is conservative in V if it satisfies the below equation

$$\sum_{\Delta V \epsilon V} \frac{\partial Q}{\partial t} \Delta V = -\sum_{f \epsilon \Gamma} F_f dS_f \qquad (1.4)$$

Where Γ is the boundary of the physical domain V.

In case we have two general overlapping zones A and B shown in Figure 5, and assume conservative schemes are used in both regions, then we have :

$$\sum_{\Delta V \epsilon B} \frac{\partial Q_B}{\partial t} \Delta V = -\sum_{f \epsilon \Gamma_B} F_f dS_f \qquad (1.5)$$

$$\sum_{\Delta V \epsilon A} \frac{\partial Q_A}{\partial t} \Delta V = -\sum_{f \epsilon \Gamma_A} F_f dS_f \qquad (1.6)$$

Where Γ_A and Γ_B are the boundaries of A and B. O is the overlapping region between A and B, $O = A \cap B$ and AO=$A \cap O$ and BO=$B \cap O$.

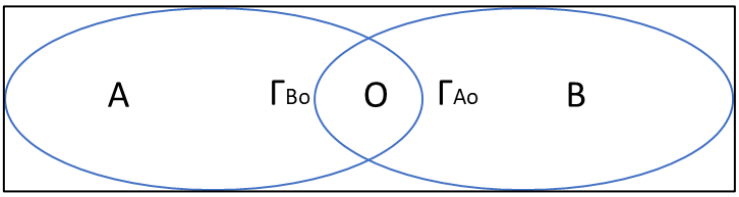


Figure 5 Overlapping regions A and B

Since only unique physical variables are possible in region O, the following physical condition should be satisfied.

$$\sum_{\Delta V \epsilon O} \frac{\partial Q_{BO}}{\partial t} \Delta V = \sum_{\Delta V \epsilon O} \frac{\partial Q_{AO}}{\partial t} \Delta V \qquad (1.7)$$

A conservative scheme for the whole domain formed by A and B (A+B-O) must satisfy.

$$\sum_{\Delta V \epsilon A+B-O} \frac{\partial Q}{\partial t} \Delta V = \sum_{f \epsilon \Gamma_A + \Gamma_B - \Gamma_{AO} - \Gamma_{BO}} F_f dS_f \qquad (1.8)$$

Where Γ_{BO} is part of the boundary of B which is overlapped by A, and Γ_{AO} is defined similarly. From Equations (5)-(8), the following conservative condition is obtained.

$$\sum_{\Delta V \in O} \frac{\partial Q}{\partial t} \Delta V = - \sum_{f \in \Gamma_A + \Gamma_B - \Gamma_{AO} - \Gamma_{BO}} F_f dS_f \tag{1.9}$$

Therefore, since equations (5) – (8) are necessary and sufficient conditions of conservation, (9) is also a necessary and sufficient condition of conservation in the overlapping regions. If two regions are patched together, then the necessary and sufficient condition is the conservation of total flux along the common boundary BO or AO.

$$\sum_{f \in \Gamma_{AO}} F_f dS_f = - \sum_{f \in \Gamma_{BO}} F_f dS_f \tag{1.10}$$

The two conditions given in (8) and (10) are the necessary and sufficient conditions of conservation for general overlapping regions. Its implementation is extremely difficult due to the following reasons:

- Two new internal boundaries Γ_{AO} and Γ_{BO} need to be generated in each domain to enforce these conditions.
- The simultaneous enforcement of both (8) and (10) is extremely difficult if not impossible.

Because of the difficulties enforcing conservation for overlapping zones, a new approach was proposed by Wang which **transfers the problem from enforcing conservation for overlapping zones to enforcing conservation for patched zones**. This new and general zonal interface treatment satisfies the condition given in (10) in a local sense. Global conservation is automatically guaranteed. The general boundary interface scheme is derived for arbitrary overlapping meshes. Patching meshes are treated as a special case.

1.1.2 Calculation of the flux at the patching boundary

The flux through each Smallest Faces element is calculated according the flow variables just to the left Q_L and to the right Q_R of the element as illustrated in figure 9.

Since flow variables are only available at cell centers, in a cell-centered approach, interpolation from the cell center to the cell face is required. For instance, in figure 9, Q_R is interpolated in face 6-7 of the boundary from the crossed cell of the major grid, while Q_L is interpolated in face 6-7 of the boundary from the boundary cell of the minor grid. Interpolation suggested by Wang, was a linear MUSCL type scheme which is being shown in a later section.

After obtaining the fluxes along the new patching boundary, the flux in each boundary face (BB) of the minor grid is calculated by summing the fluxes in each SF element (fig.9). Therefore, the total flux in a boundary cell of the minor grid, is the summation of the flux in the boundary face and all the fluxes along the other faces of the boundary cell.

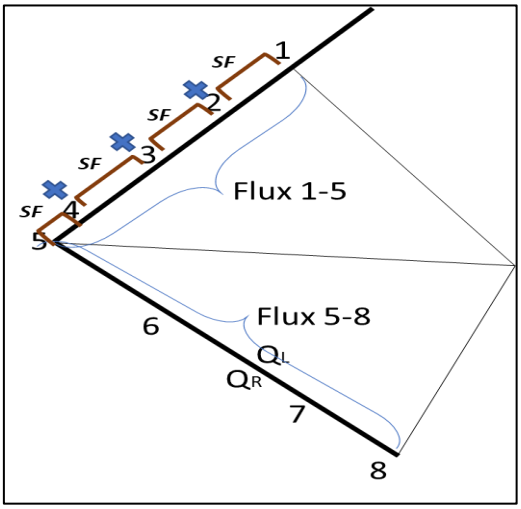


Figure 6 Flux 1-5 : Summation of the flux of the Smallest faces SF

The total flux in a crossed cell of the major grid is described below:

Since the cells of the major grid are intersected by the boundary Γ_{BO}, these crossed cells (with orange color in fig.8) are now polygonal control volumes (named i-cells) which have the same indices as the crossed cells but different geometry. The flow properties stored in the crossed cells are assumed to be the cell center values of the i-cells. The total flux through an i-cell can be expressed as:

$$\sum_{icell} \boldsymbol{F_f} = \boldsymbol{W_{i+1/2,j}F_{i+1/2,j}} + \boldsymbol{W_{i-1/2,j}F_{i-1/2,j}} + \boldsymbol{W_{i,j-1/2}F_{i,j-1/2}} + \boldsymbol{W_{i,j+1/2}F_{i,j+1/2}} + \boldsymbol{F_{CA}} \quad \textbf{(1.11)}$$

Where $\boldsymbol{F_{i+1/2,j}}$ etc. are fluxes through the crossed cell which are calculated in the 'normal' way, $\boldsymbol{W_{i+1/2,j}}$ etc. are area weights for the corresponding i-cell and F_{CA} is the flux through the CA element within the crossed cell.

Therefore, with this weighting the fluxes through all i-cells can be casted the same form no matter the geometrical shape those cells cam have. An example of the flux calculation through an i-cell is given in figure 10.

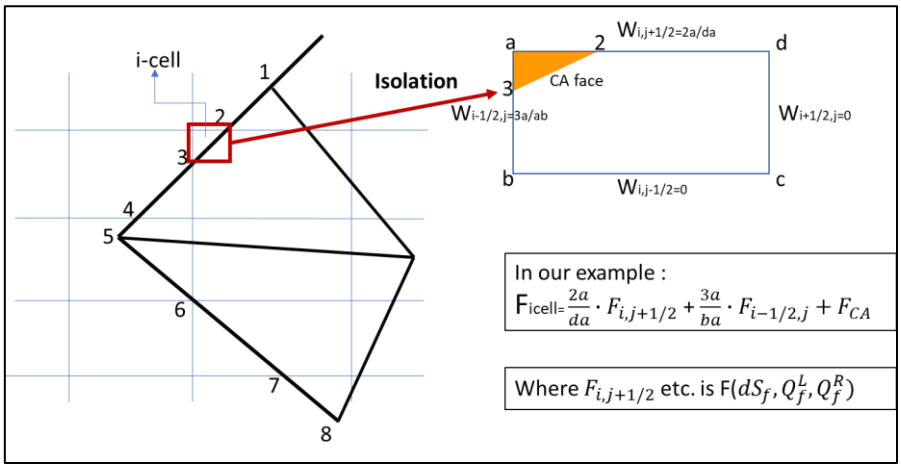


Figure 7 Total Flux calculation through an i-cell

1.2 Fully conservative Structured/Unstructured Chimera Grid Scheme

Wang et al [4] extended the conservative interface for Chimera grids to overlapped structured/ unstructured grids. The difficulty with the structured CFD approach is the generation of structured grid for complex geometries. On the other hand, unstructured grid approach has tremendous geometric flexibility in that a grid does not need to conform to a predefined topology. Its disadvantages include relatively expensive memory and CPU overhead.

With this approach, unstructured grids can be produced around geometrically complex components, for which it may be difficult to generate structured grids. Then the unstructured grids are overlapped with a major background structured grid or even a Cartesian grid to form a global Chimera grid.

1.2.1 Tested cases

Wang tested this conservative Chimera grid method in multiple cases: Steady and unsteady, 2D and 3D, transonic and subsonic. The test cases demonstrated that the conservative Chimera is accurate and possesses excellent convergence properties.

For the sake of completeness, one test case of **transonic flow over staggered Biplane is presented below.**

This case demonstrates the hybrid(structured/unstructured) Chimera approach for external flow problems. In figure 11, we can see the three-zone overlapped grids for the biplane configuration. The two elements are NACA 0012 airfoil, staggered ½ chord length in the chordwise and pitch directions. Mach number is 0.7 and angle of attack is $a=0^0$. Pressure contours of the conservative Chimera approach and of an adaptive mesh from literature are displayed in figures 12 and 13 and there is good agreement between them. Machine zero convergence was achieved.

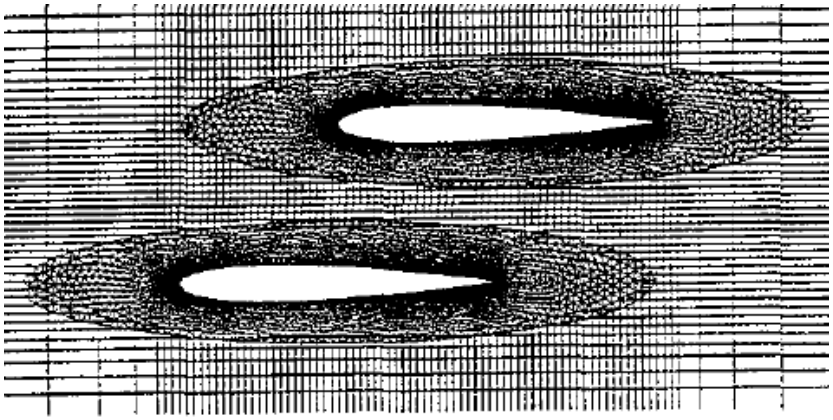


Figure 8 Three zone Hybrid Overlapped Mesh for Bi-plane

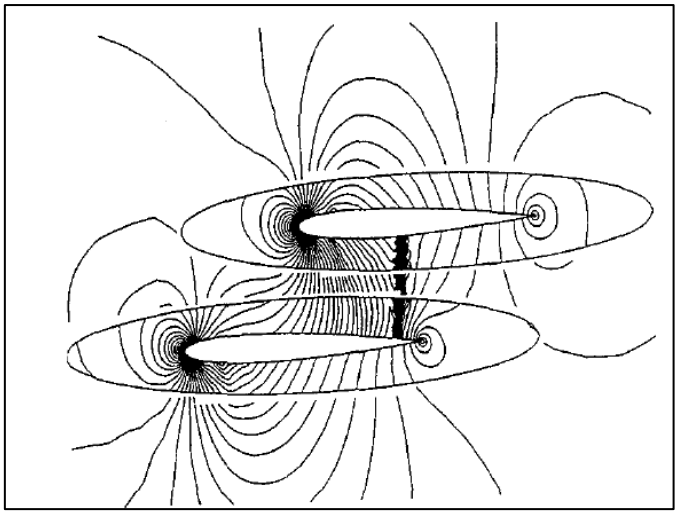


Figure 9 Pressure Contours of Chimera grid

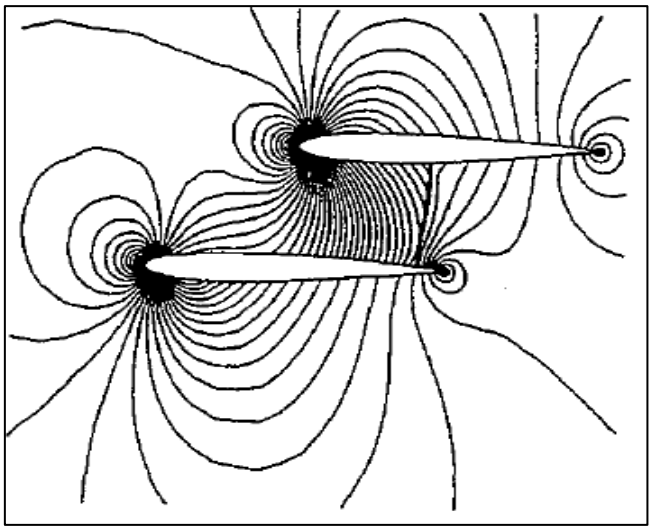


Figure 10 Pressure contours of Adaptive mesh

Chapter 2

2. Fully conservative Chimera method on moving grids

In this chapter, the further development of the conservative interface algorithm for stationary Chimera grids, which was described in **Chapter 2** and is developed in **Chapter 4,** to handle moving Chimera grids.

The fully conservative zonal interface algorithm for overlapped (Chimera) grid has been extended to handle moving body flows by Wang [5]. To ensure flow conservation, the motion of the computational grid needs to be considered. Several new issues arising from moving and deforming grids, e.g., vanishing and new-born cell problems, are addressed with a cell-merging-unmerging technique. The grid velocities are determined in a way which satisfies free stream preserving (the Geometric Conservation Laws).

2.1 Approaches for moving grids

In general, there are three major types of computational approaches used in CFD to handle moving bodies:

a. Grid Deformation Approach

In this approach, the computational grid around the moving body is "adjusted" at each time step such that it conforms to the new body position. The grid topology and total number of control volumes are preserved. It has been used in both structured and unstructured grid-based flow solvers.

Advantages of the approach are that the flow solver can be easily made fully conservative and that flow variable interpolations are unnecessary. The disadvantage of the approach is that the scale of the motion of moving bodies must be small in comparison with the scales of body-to-body and body-to-boundary distances, and in most cases grid rotations are not allowed. Otherwise the relative motion can severely distort the mesh and even cause grid line crossings, breaking down the flow solver. The cause of the problem is the fixed grid topology imposed on grid deformation.

b. Grid Remeshing/Adaptation Approach

With this approach, the grid near the moving body is regenerated at each time step according to the new position of the moving body. This approach eliminates the limitation on grid topology and thus grid quality around the moving body can be maintained. Furthermore, grid motions of arbitrary scales are allowed. The main drawback of this approach is that flow variables must be interpolated from the old to the new grid at each time step and it is exceedingly difficult to interpolate the flow variables in a conservative manner.

c. Chimera Grid Approach

The previous two approaches fall in the category of patched (non-overlapped) grid methods, in which different zones must share some common interfaces. In the Chimera grid approach, no limitations are placed on zonal interface matches. Therefore, component grids can move relative to each other in an arbitrary fashion. Grid adjustments or grid regeneration are thus avoided. Flow variables are interpolated between overlapped grids to exchange information. However, it also suffered from the conservation-violating data interpolation across grid interfaces.

In order to satisfy the conservation laws, cell-merging-unmerging technique is used to deal with the so-called vanishing and new-born cell problems caused by the relative motions between Chimera grids. Special attentions were paid to the so-called Geometric Conservation Laws, which states that a uniform free stream must be preserved given arbitrary grid motions.

2.2 Selection of Control Volumes-Cell Merging

The governing equation in integral form for an arbitrary moving control volume is (as it is described in Chapter 5) is given by the formula (5.9):

$$\frac{\partial}{\partial t}\int_{D(t)} \vec{U}\, dD + \oint_{\partial D(t)} (\overrightarrow{F_c} - V_g \vec{U} dS - \overrightarrow{F_u}) dS = \int_{D(t)} \vec{Q}\, dD \quad (3.1)$$

where Vg=$\vec{u}_{grid} \cdot \vec{n}$, D is the control volume,)$\vec{U}$ is the vector of the Conservative Flow Variables, $\overrightarrow{F_c}$ are the convective fluxes, $\overrightarrow{F_u}$ are the viscous fluxes.

After the successfully generation of the Zonal Interface as described in Chapter 2, care needs to be exercised in choosing the right control volumes so that (3.1) is applicable. Two potential problems can occur, i.e., the vanishing and newborn cell problems. They

are illustrated in Figure 14. At time step n, cell B is a cut cell. It becomes a hole cell at time step n+1, i.e., cell A disappears between n and n+1. On the other hand, cell B is a cut cell at n but becomes a normal cell at n+1. Therefore, a new cell is created in the computational domain. For both situations, (3.1) cannot be applied because the control volumes between n and n+1 are not well defined.

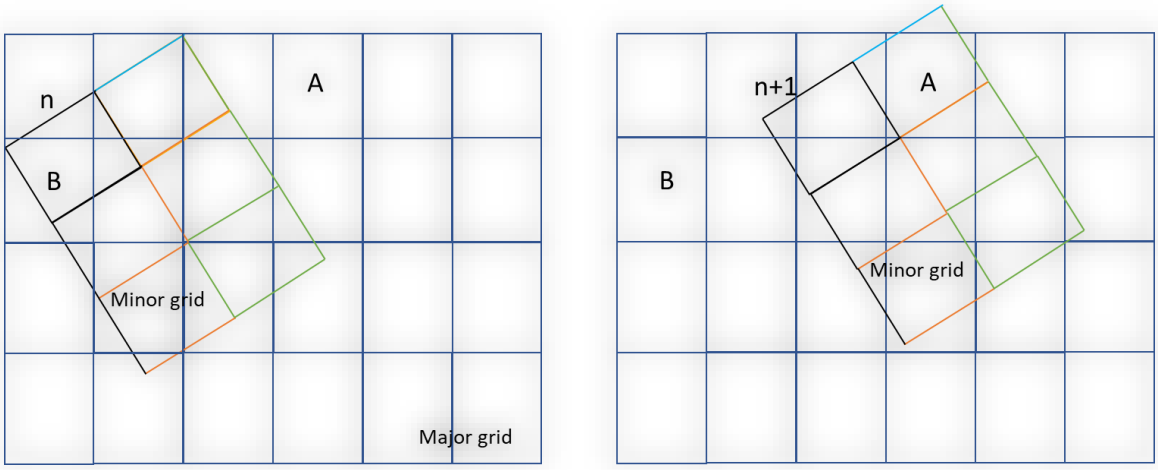


Figure 11. Illustration of vanishing and New-born cell problems for moving chimera grids

A technique called cell-merging is used here to eliminate the problems[6]. The basic idea of merging is the combination of adjoining cells so that all boundaries between them are ignored for finite-volume integration purposes, thus forming new, larger cells. After cell merging, the following criterion must be satisfied:

• The patch boundary does not cross any corner vertex of a merged cell between time step n and n+1.

By satisfying this condition, the topology of a merged cell stays the same within one grid motion step. Therefore (3.1) can be directly applied to the merged cell. The conservative variables of a merged cell are obtained by volume-averaging from the underlying cells which compose the merged cell. In practical implementations, minimum number of cells should be merged to reduce loss of resolution due to larger control volumes resulting from cell merging. One observation is that cells should be merged in the direction as perpendicular to the patch boundary as possible. The following cell merging algorithm has been developed and implemented

1) Identify all grid vertices in the major grid which are swept over by the outer boundary of the moving minor grid and record all such vertices in a point list.
2) For each point in the point list, choosing one of the two possible merging directions based on the local unit normal of the patch boundary.

3) Merge the four cells which share the point into two larger cells in the chosen direction and eliminate the point from the point list.
4) If there is at least one point left in the list, go to Step 2. The control volumes for the minor grid are identical to grid cells.

2.2.1 Zonal Interface Generation

Consider the two overlapping regions in figure 5. Let A to be the major grid and B the minor grid. The outer boundary of the minor grid, Γ_{BO} is chosen to divide the whole physical domain (A + B) into two non-overlapping zones (A-O) and B as shown in figure 6. Therefore, the conservation condition for the overlapping zone becomes flux conservation on the common boundary $\Gamma_{BO.}$

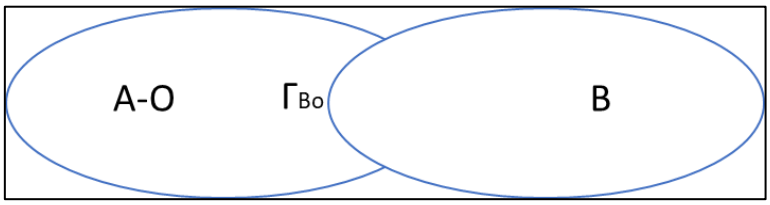


Figure 12 Zonal interface Generation

The **key step** in this proposed approach is to find the **intersection surface of two arbitrary overlapping zones.**

In the next figure, the major mesh is a structured mesh while the minor consists of unstructured mesh. The common boundary is the outer boundary Γ_{BO} of the minor mesh which consists of the boundary points of minor grid and the points at which intersects the cells of the major grid.

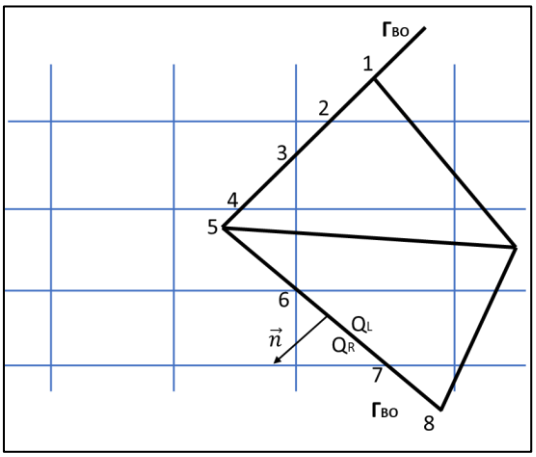


Figure 13 Points of the common boundary of the two meshes

There are three different types of boundary faces created by this common interface generation.

1. Boundary cells of minor grid, e.g. 1-5 and 5-9. These are called BB elements.
2. Part of ΓBO that is included within one cell in the major grid. These are called CA elements e.g. 3-4 and 4-5-6.
3. The smallest faces that compose the boundary Γ_{BO}. These are called SF elements, e.g. 1-2,2-3,3-4,4-5 .

Obviously, one CA and one BB element is composed of at least one SF element. The generation of the boundary interface in the major grid involves determining all BB, CA and SF elements and their mutual relations.

Furthermore, there are three different type of cells which are created in the major grid after this common boundary generation. As you can see in the figure 7 these are:

1. Normal cells: Cells that are located outside the minor grid.
2. Crossed cells: Cells of the major grid which are crossed by boundary $\Gamma_{BO.}$
3. Hole cells: Cells of the major grid which are inside minor grid (overlapping region).

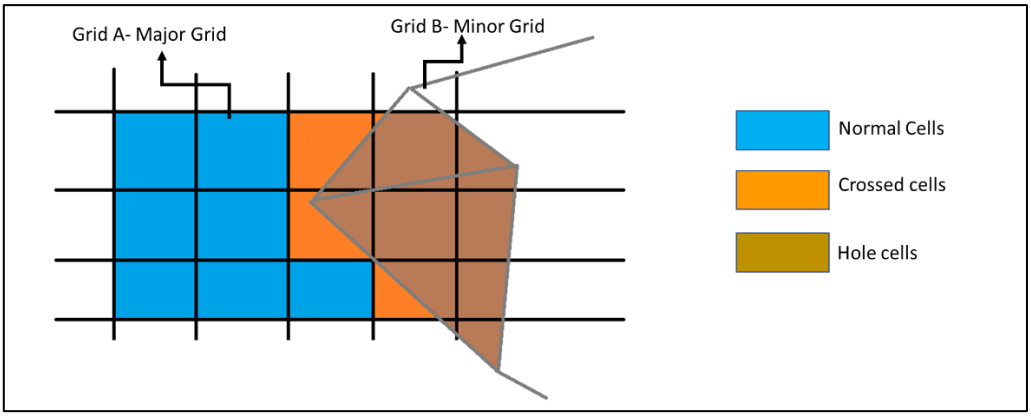


Figure 14 Cells Types in major grid

Flow variables in the normal cells are updated in the 'normal way' while hole cells are not taken into consideration in the solution of the flow.

2.3 Flux calculation on the Patch boundary

As described in the previous chapter in order to calculate fluxes on the common boundary three separate steps are taken to obtain the fluxes : reconstruction, Riemann solver and calculation of grid velocity.

2.3.1 Calculation of grid velocity

In this subchapter, we will focus on the calculation of grid velocity.

For moving grids, the flux vector consists of a contribution from the grid velocity (the term $V_g\vec{U}dS$ where Vg=$\vec{u}_{grid}\cdot\vec{n}$. in Eq. 3.1). In the calculation of grid velocity Vg, it is critical that any arbitrary grid motions do not disturb a uniform free stream; Otherwise, any grid motion may induce spurious flows, destroying the credibility of the flow solver. This statement is sometimes referred to as the so-called Geometric Conservation Law (GCL). Let U = constant in (3.1), we obtain

$$\frac{D^{n+1}-D^n}{\Delta t}=\sum_f(V_g\cdot ndS)_F \qquad (3.2)$$

Eq. (3.2) is the GCL in a discretized form. It involves only the geometric properties and must be satisfied in the discrete sense. Consider a Chimera grid system with a stationary major grid and a moving minor grid. The only cells, which are deforming in the major grid are the cut cells (after possible cell merge). The only moving faces of the cut cells are the SF (smallest faces) which are bounded by these cells. Therefore, the GCL for a cut cell can be expressed as

$$\frac{D^{n+1}-D^n}{\Delta t}=\sum_{Sf}(V_g\cdot ndS)_{SF} \qquad (3.3)$$

where SF arc the smallest faces bounded by the cut cell, and V"+ I and V" are the volumes of the cut cell at time n +1 and n respectively. Each smallest face has a distinctive face velocity. If (3.3) is satisfied, the GCL is guaranteed. However, for the purpose of calculating the face velocity, we assume that the smallest faces within a cut cell all have the same velocity. Then the following equation is obtained.

$$V_{gn}=\frac{D^{n+1}-D^n}{\Delta t\,|\sum_{SF}(ndS)_{SF}|} \qquad (3.4)$$

Eq. (3.4) is actually used to calculate the face velocities of all the smallest faces inside a cut cell for updating that cut cell in the major grid. It is therefore obvious that the GCL is satisfied for the cut cell. When updating the outer boundary ' cells of the minor grid, the grid velocities need to be calculated differently. Otherwise, the GCL is usually

not satisfied. It is obvious that all smallest faces located in the same boundary segment of the minor grid have the same grid velocity. Therefore, the velocity of the patch boundary segment is calculated and then distributed to the smallest faces. The grid velocity of a line segment is simply the volume swept by that segment between time levels n and n+1 divided by the time step and segment length, i.e.,

$$V_{gn} = \frac{D^{n+1} - D^{n}}{\Delta t \cdot \Delta s} \quad (3.5)$$

With this formula, it is easy to verify that the GCL is satisfied for all the minor grid cells. Since grid velocities are calculated differently when updating the major and minor grid, one may argue that the whole scheme is non-conservative. However, it can be proved that the following equation is true:

$$\left[\sum_{Sf}(V_g \cdot ndS)_{SF}\right]_{Major} = \left[\sum_{Sf}(V_g \cdot ndS)_{SF}\right]_{Minor} \quad (3.6)$$

The left and right side of (10) are simply the total volume swept by the patch interface between time step n and n+ 1. They are identical due to the way the grid velocities are calculated in the major and minor grid.

2.4 Cell-Unmerging

After the flow variables are updated at time step n+ 1, the merged cells around the patch boundary are unmerged, which reverses the cell-merging operation. The purpose of this operation is to reidentify members of merged cells as separate entities to recover the loss of resolution due to cell merging. The individual flow variables of grid cells are recomputed from the merged cells upholding flow conservation principles. Flow gradients of merged cells are used in the computation to further increase resolutions.

Chapter 3

3. Mesh Patching-Methodology and data structure

In this chapter, the data structure of the code for the composition of the meshes is explained and the process of making the composition is described.

3.1 Data structure of the mesh composition code

3.1.1 Data structure for the nodes

In order to develop the mesh-composition code, two squares meshes were selected: minor mesh, which is embedded in the major mesh, was set in various angles (fig. 14) and the major mesh is always a Cartesian mesh. At this point, it should be noted that this code, could compose a structured minor mesh with a structured major mesh and an unstructured minor mesh with a structured major mesh.

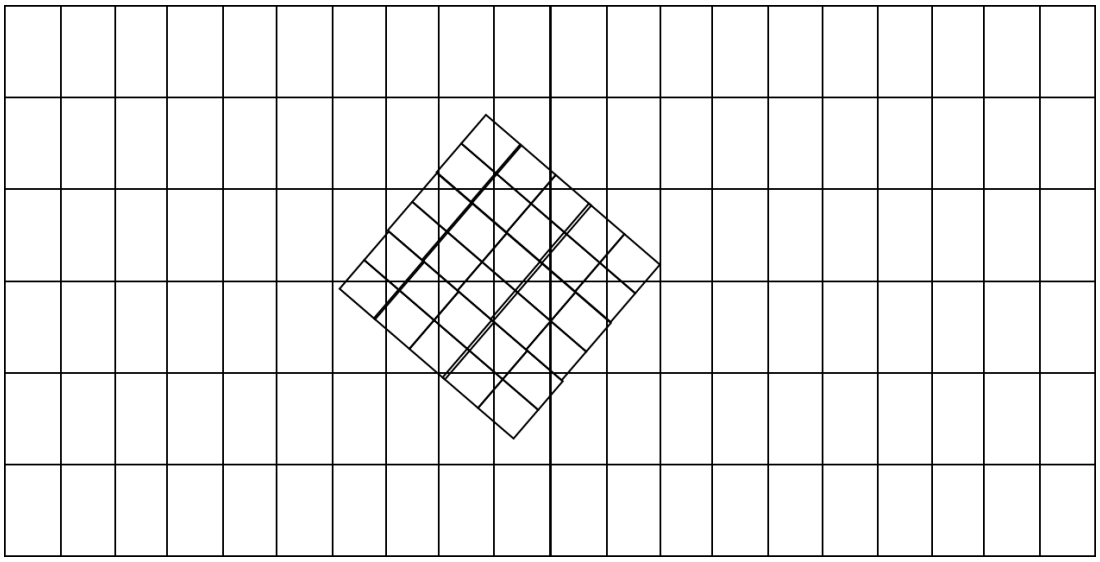


Figure 15.Major and minor mesh used for the development and testing of the mesh composition algorithm

Regarding the nodes of the major mesh there are four different cases:

1. Hole nodes: These are the nodes which are located in the overlapping region and once detected should be not taken into account for the new mesh.

2. Common nodes: These are the nodes of the major mesh, which are coincident with the nodes of the minor mesh in the outer boundary of the minor mesh.
3. Normal nodes: The nodes of the major mesh outside of the overlapping region.
4. Intersection nodes: The new nodes, which are shaped by the intersection of the two meshes.
5. Cross points: The nodes of the major mesh, which are located on the outer boundary of the minor mesh.

The five different types of nodes are illustrated in the figure 15 below.

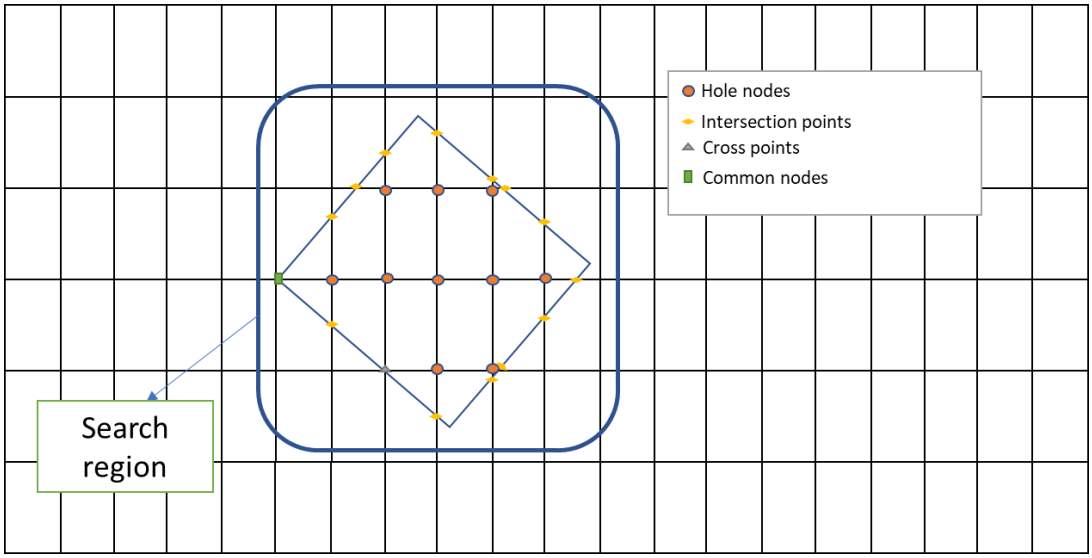


Figure 16Data structure of the major grid points

To classify the points of the major grid into the above types, at first, a search area is designed which is a big box whose maximum and minimum coordinates values are as below:

Ymin_search region=ymin (outer boundary coordinates)

Xmin_search region=xmin(outer boundary coordinates)

Xmax_search region=xmax(outer boundary coordinates)

Ymax_search region=ymax(outer boundary coordinates)

Therefore, the distance of the points of the major grid, which are inside the search region, to each node of the outer boundary of the minor mesh is calculated. Finding the minimum distance, the closest node of the outer boundary of the minor mesh is now know, as well as the next node from the closest node and the previous node from it. In case there is a distance, which is zero, or inside a tolerance, which is set from the user, the node of the outer boundary is a common node to the node of the major mesh.

Otherwise, it must be defined if this point of the major mesh is inside the hole or outside of it. This is taking place by using cross products as shown below.

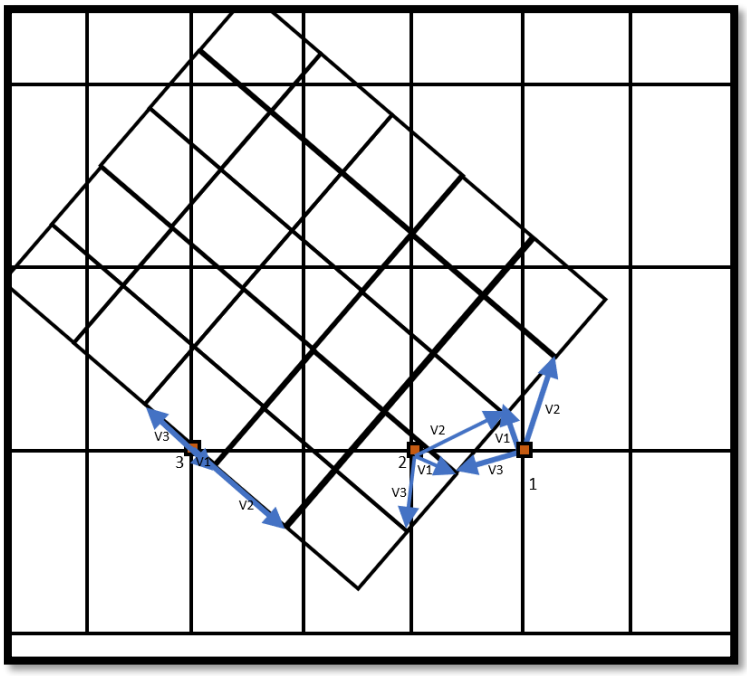


Figure 17.Checking process for the points

As shown in figure 16, for every point 3 Vectors are calculated. V1 is the vector from the point of the major mesh to the closest node of the outer boundary of the minor mesh. V2 is the vector from the point to the next of the closest node of the outer boundary and V3 is the vector from the point to the previous of the closest node of the outer boundary. If the cross product V2xV1 is >0 and the cross product V3xV1<0 the point is outside the outer boundary of the minor mesh and is defined as normpoint (point 1 in figure 16). If the cross product V2xV1<0 and V3xV1>0 then the point is inside the outer boundary of the minor mesh and is defined as a hole point(point 2 in figure 16). Finally, if the cross product V2xV1 = 0 or V3xV1=0 then this point is on the outer boundary of the minor mesh and is defined as crosspoint.

3.1.2 Data structure for edges

After the definition of the data types of the nodes of the major grid, next step is definition of the edges' types. There are seven different types of edges:

Edges with type 0: These edges are called hole edges, because they belong to the overset area. Both nodes of these edges are hole nodes.

Edges with type 1: These edges are called normal edges, because both of their nodes are normal nodes.

Edges with type 2: These edges are called intersected edges, because they are intersecting with the edges of the outer boundary of the minor mesh. One node of this edge is normal node and the other is a hole node.

Edges with type 3: These edges are called common edges and both of their nodes are common nodes to an edge of the outer boundary of the minor mesh.

Edges with type 5: These edges are called cross edges because they do not cross the edge of the outer boundary. One of their nodes is normal node and the other is cross point.

Edges with type 7: These edges are called common node edges because one of their nodes is normal and the other one is a common node.

The above edges are illustrated in the figure below.

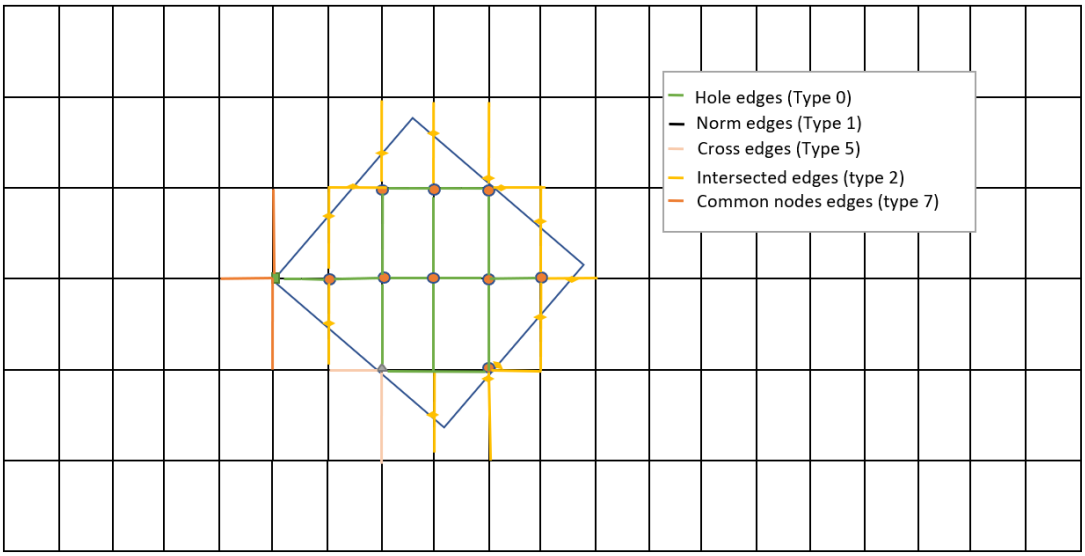


Figure 18.Types of major grid edges

The next table (Table 1) is describing in detail all the cases for the definition of the above types.

Types	Holepoints	Normpoints	Crosspoints	Common nodes
Ftype 0	2		2	
Ftype 0	1		1	
Ftype 0	1			1
Ftype 0			1	1
Ftype 1		2		
Ftype 2	1	1		
Ftype 5		1		1
Ftype 7		1		1
Ftype 3				2

Table 1. Ftypes cases

3.1.3 Data structure of the cells

After the definition of the edges' types, the next step is to determine the cells types based on the ftypes. There are four different types of cells :

Ctype 1: All edges are normal edges.

Ctype 0 : All edges are hole edges.

Ctype 2 : At least one edge is normal and at least one edge is hole edge. However, all the combinations, which make up this type, are described in table 2.

Ctype 3: This type contains of two hole edges and a combination of the other ftypes.

Figure 18 shows all the cases, which define the c-type of the cells, and **figure 19** shows the cells after the union of the two meshes. Furthermore, table 2 contains all the information about the c-type gathered.

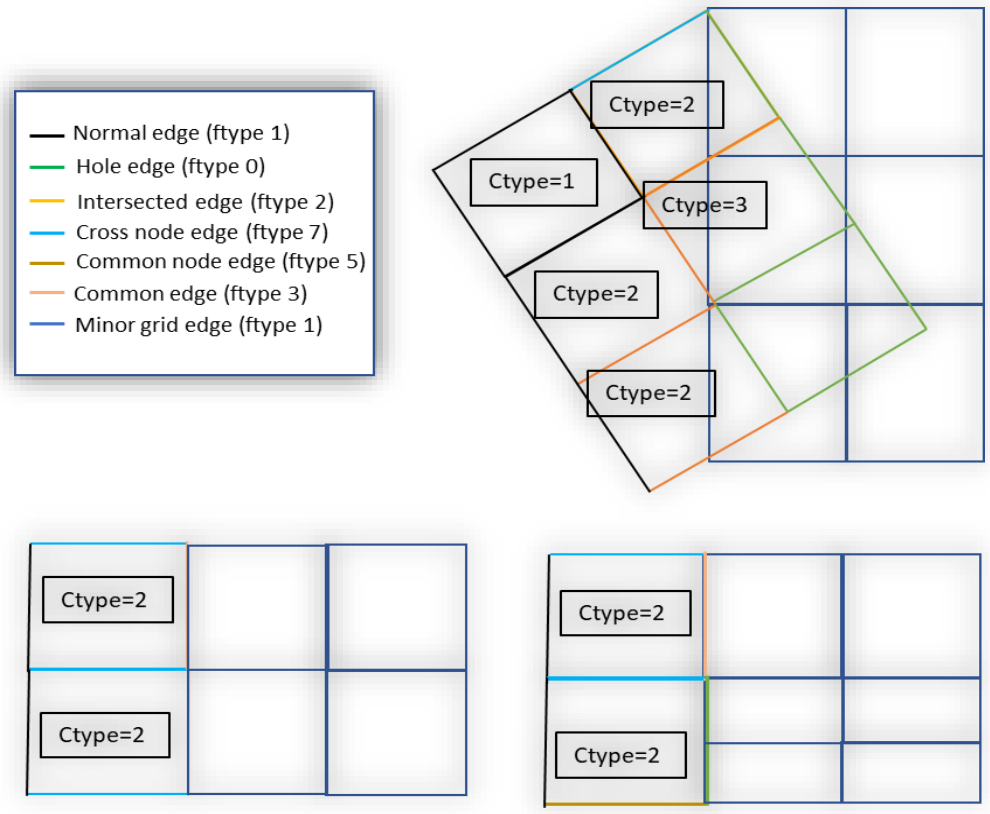


Figure 19. C-type cases for the cells

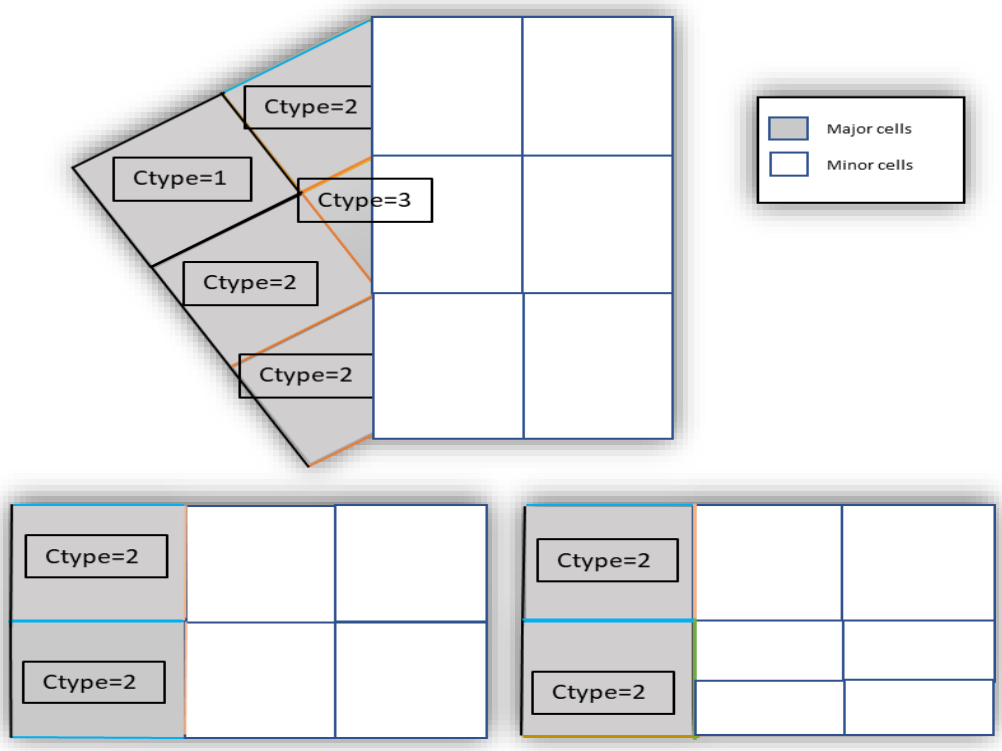


Figure 20.Cells of major and minor mesh after the union

Types	Edges with ftype 0	Edges with ftype 1	Edges with ftype 2	Edges with ftype 3	Edges with ftype 5	Edges with ftype 7
Ctype 0	4					
Ctype 0	3			1		
Ctype 1		4				
Ctype 2	1	1	2			
Ctype 2	1	1				2
Ctype 2	1	1			2	
Ctype2		1		1		2
Cype 3	2		2			
Ctype 3	2					2
Ctype 3	2				2	

3.1.4 Data structure of 'Smallest faces'

As their name reveals these are the smallest faces, which are created by the intersection of the edges of the two meshes. Figure 21 shows how smallest faces are created after the union of the two meshes.

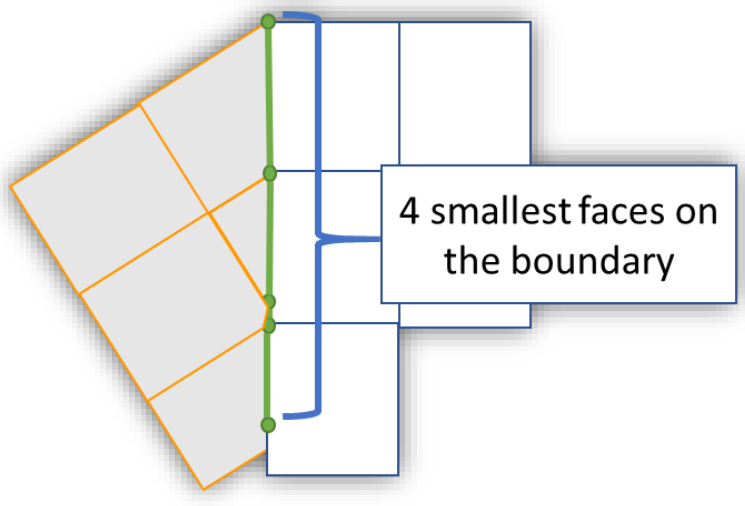


Figure 21 Smallest faces on the outer boundary of the minor mesh

3.2 Algorithm for the mesh composition

In the below sub-chapter, the process for the mesh composition is described. It contains of four steps.

3.2.1 Step 1-Outer boundary and points structure definition

The minor grid outer boundary points are set in counterclockwise direction. Then the process, which is described in 3.1.1, is followed to find the different types of nodes in the major mesh.

3.2.2 Step 2- Intersection points

Based on the data structure of the nodes, the faces types are defined. Then, the edges of type 2 are scanned and the intersection points with the edges of the outer boundary are found. More details of the process of finding the intersection point between two edges are given in the Appendix A1. The intersection points and the crosspoints (nodes of the major grid that are coincident on the outer boundary) are saved in an array *intersect_po*, which is a derived datatype containing information about the coordinates of the intersection points and the crosspoints as well as the edge of the outer boundary they intersect/touch. Another important derived datatype is the array *cutcellindex*, whose size is equal to the total number of cells of the major mesh and contains the two points of a cell, which intersect the outer boundary of the minor mesh. Figure 22. Gives an example of the **intersect_po** and **cutcellindex** arrays. At this point is important to be noted that the numbering of the intersection points and the crosspoints is put after the total number of nodes of the minor mesh, so that these points are the consequence of minor mesh nodes.

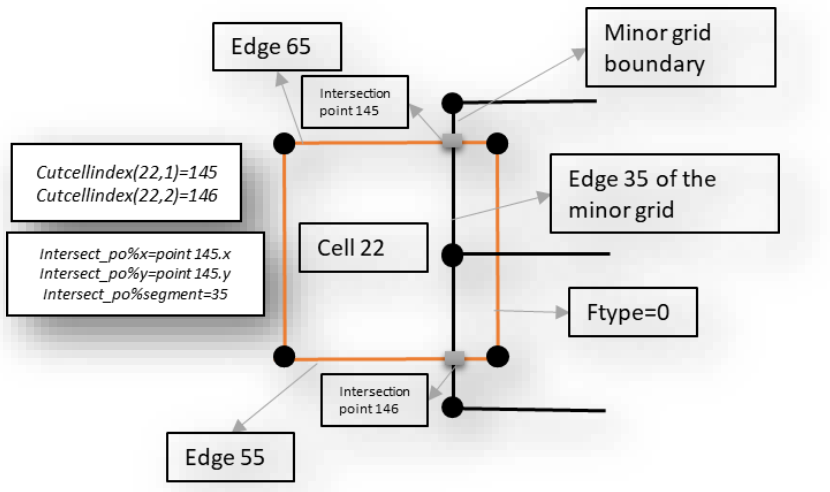


Figure 22Example of intersect_po and cutcellindex arrays

3.2.3 Smallest-faces search

Next milestone is the correct completion of the smallestfaces data type, which is common to the type of the edges of both meshes.

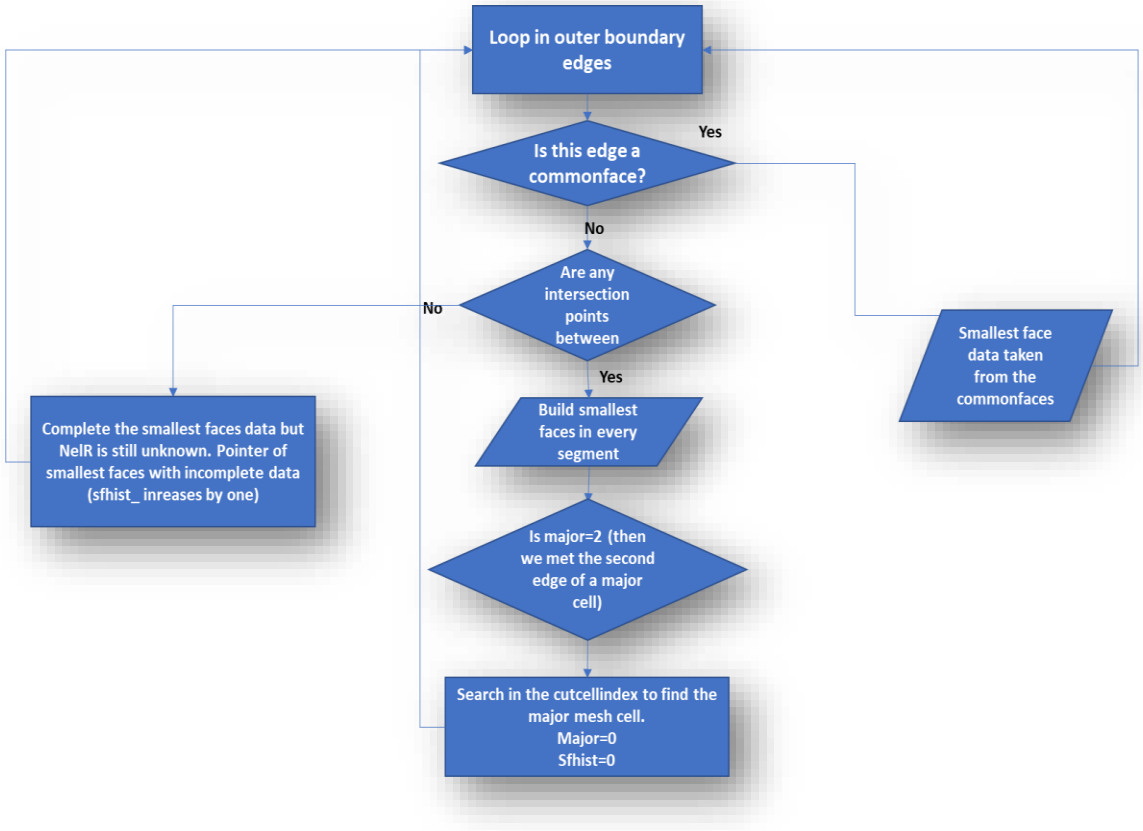


Figure 23. Algorithm for constructing smallestfaces

What is shown in figure 22, it is going to be described in this paragraph. The above algorithm was developed to construct the smallest faces.

Especially, every edge of the outer boundary of the minor mesh is scanned. If this edge belong to commonfaces (which means that this edge coincide with an edge of the major mesh) then the smallest face is easily completed as all the data (Right cell, left cell, nodes) is known.

In case the edge is not a commonface, then through intersect_po the number of intersection points on that edge is found. If it is zero, then the smallest face takes all the data of the edge of the minor mesh without the information of the right cell and the pointer, which shows the number of the smallest faces without a proper right cell (sfhist), is increased by 1.

In case the edge has x intersection points, x+1 smallest faces will be created in that specific edge of the outer boundary. When the flag **major** is 2, it means that we have encountered an intersection point of the second edge of a major cell (e.g. point 146 in

figure 21) and the algorithm is going to search the cutcellindex array to find the right cell of the major mesh (smallest faces are in counterclockwise direction looking from inside the minor mesh, therefore the cells of the major mesh are always on the right of a smallest face). Whenever the right cell is found, the NelR data is filled in the total of **sfhist** smallest faces.

3.2.4 New numbering and data transfer

All the nodes of the major mesh, which are of type normal, are getting a new nodal numbering, which starts from the last intersection point. Therefore, value normpoint(140) is the new number of the node 140 of the major mesh. If normpoint (130) is zero it means that node 130 of the major mesh was not a normal node.

In the same way, all the edges of the major mesh, which have f-type 1,2,5,7, are having a new numbering, which starts from the last smallest face. Consequently, majorfaces(150) is the new number of the edge 150 of the major mesh.

Finally, all the cells with c-type 1,2,3 are getting a new numbering as well, which starts from the last cell of the minor mesh.

All the above, data is transferred to the final arrays which contain the connectivity data of the new composed mesh.

3.2.5 Boundary faces treatment

The edges of the minor mesh which are along the outer boundary are erased, as their position is being taken by the smallest faces which are created on the outer boundary of the minor mesh and they are interior faces from now on. The same applies for the dummy cells of the minor mesh, which are erased too.

In case, the minor mesh is around a body, the treatment is a little different. The dummy cells on the body, are going in the end of the cells array after the last dummy cell of the major mesh outer boundary. All the rest remain as they were.

Chapter 4

4. Eulerian Solver

In this chapter the Eulerian solver (MaPFlow) into whom the mesh-composition code is implemented is described. MaPFlow solves the Unsteady Reynolds Averaged Navier Stokes equations (URANS) in a multiprocessor environment using the MPI protocol. The solver is 2nd order in space and time; makes use of the Roe approximate Riemann solver for reconstruction; is equipped with Preconditioning to handle Low Mach number flows and uses the Spalart Allmaras and the k − ω SST models for turbulence closure.

4.1 Governing equations

4.1.1 Conservative form

Let D denote a volume of fluid and ∂D its boundary. By integrating the Governing equations over D, the following integral form is obtained:

$$\int_D \frac{\partial \vec{U}}{\partial t} dD + \oint_{\partial D} (\overrightarrow{F_c}\, dS - \overrightarrow{F_u}) dS = \int_D \vec{Q}\, dD \quad (4.1)$$

In (6.1) $\vec{U}$ is the vector of the Conservative Flow Variables,

$$\vec{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \quad (4.2)$$

where ρ denotes the density, (u, ν, w) the three components of the velocity field and E the total energy. $\overrightarrow{F_c}$ and $\overrightarrow{F_u}$ denote the Convective and Viscous Fluxes, respectively.

$$\vec{F_c} = \begin{pmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho (E + \frac{p}{\rho}) V \end{pmatrix} \quad (4.3)$$

$$\vec{F}_v = \begin{pmatrix} 0 \\ n_x\tau_x x + n_y\tau_x y + n_z\tau_x z \\ n_x\tau_y x + n_y\tau_y y + n_z\tau_y z \\ n_x\tau_z x + n_y\tau_z y + n_z\tau_z z \\ n_x\Theta_x + n_y\Theta_y + n_z\theta_z \end{pmatrix} \quad (4.4)$$

Where V is the contravariant velocity, V=$\vec{u} \cdot \vec{n}$ and

$$\Theta_x = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\frac{\partial T}{\partial x}$$

$$\Theta_y = u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\frac{\partial T}{\partial y}$$

$$\Theta_z = u\tau_{xz} + v\tau_{zy} + w\tau_{zz} + k\frac{\partial T}{\partial z} \quad (4.5)$$

The above system is completed with the equation of state for perfect gas :

$$p = (\gamma - 1)\rho\left[E - \frac{u^2+v^2+w^2}{2}\right] \quad (4.6)$$

4.1.2 Moving Grids

In case D changes in time, the time derivative in (4.1) will also act on D(t). In order to pass time derivation into the integral, Reynold's transport theorem is applied:

$$\frac{\partial}{\partial t}\int_{D(t)} \vec{U}\, dD = \int_{D(t)} \frac{\partial \vec{U}}{\partial t} + \int_{D(t)} \nabla(\vec{U} \cdot \vec{u}_{vol}) dD = \int_{D(t)} \frac{\partial \vec{U}}{\partial t} \oint_{\partial D(t)} \vec{U} \cdot (\vec{u}_{vol}\vec{n}) dS \quad (4.7)$$

where $\vec{u}_{vol}$ is the velocity that defines the time evolution of D(t). If $\vec{u}_{grid}$ denotes the grid velocity then $\vec{u}_{vol}$=$\vec{u}_{grid}$, hence,

$$\int_{D(t)} \frac{\partial \vec{U}}{\partial t} = \frac{\partial}{\partial t}\int_{D(t)} \vec{U}\, dD - \oint_{\partial D(t)} \vec{U} \cdot (\vec{u}_{vol}\vec{n}) dS \quad (4.8)$$

Substituting (4.8) in (4.1), provides the Arbitrary Langrangian-Eulerian(ALE) formulation of the governing equations :

$$\frac{\partial}{\partial t}\int_{D(t)} \vec{U}\, dD + \oint_{\partial D(t)} (\overrightarrow{F_c} - V_g\vec{U} dS - \overrightarrow{F_u}) dS = \int_{D(t)} \vec{Q}\, dD \quad (4.9)$$

where Vg=$\vec{u}_{grid} \cdot \vec{n}$

4.2 Spatial Discretization

In MaPFlow the flow variables are calculated and stored at cell centers. Assuming that the cell volume remains unchanged:

$$\frac{\partial}{\partial t}\int_D \vec{U}\, Dd = D\frac{\partial \vec{U}}{\partial t} \qquad (4.10)$$

where:

$$\vec{U} = \frac{1}{D}\int_D \vec{U}_{exact} dD \qquad (4.11)$$

Thus, equation (4.1) becomes:

$$\frac{\partial \vec{U}}{\partial t} = \frac{1}{D}\left[\oint_{\partial D}(\vec{F}_c - \vec{F}_v)dS - \int_D \vec{Q}dD\right] \quad (4.12)$$

The surface integral is approximated using piecewise constant fluxes over the cell faces that are calculated at their centers. For cell I,

$$\frac{\partial \overrightarrow{UI}}{\partial t} = \frac{1}{D}\left[\sum_{m=1}^{Nf}(\vec{F}_c - \vec{F}_v)_m \Delta S_m) - \overrightarrow{(QD)}_I\right] \quad (4.13)$$

where N_f is the number of faces the cell has and ΔS_m is the area of face *m*. The terms $(\vec{F}_c)_m$, $(\vec{F}_v)_m$ are the convective and viscous fluxes through face *m.*

4.2.1 Reconstruction of variables

In order to calculate the fluxes appearing in the right-hand side of (4.13), the values of all flow variables at the face centers are needed. This information is absent since all flow variables are defined at the cell centers. Passing the flow information from the cell centers to the faces is carried out by means of variable reconstruction.

Consider two cells I,J being in contact over face f. Variable reconstruction on f can be defined either starting from cell I or cell J. For compressible solvers it is assumed that across the face the flow experiences a jump defined by the left L and right R states. The L/R specification depends on the normal to f which directs from L to R.

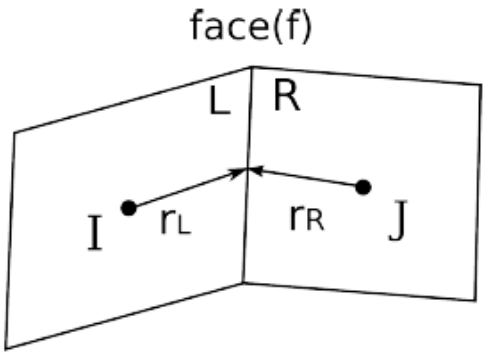


Figure 24. Reconstruction of variables on a face (f)

In MaPFlow, the Piecewise Linear Reconstruction (PLR) is used. PLR approach implies that the flow variables are linearly distributed over the control volume. Thus, the Left and Right reconstructed states are defined as follows:

$$\vec{V}_L = \vec{V}_I + \Psi_I(\nabla\vec{V}_I \cdot \vec{r}_L) \quad (4.14)$$

$$\vec{V}_R = \vec{V}_J - \Psi_J(\nabla\vec{V}_J \cdot \vec{r}_R) \quad (4.15)$$

Where $\vec{r}_L, \vec{r}_R$ denote the distance vectors pointing from the cell centers to the face center (Fig. 23) and Ψ a limiter function. In the above expression, the gradients are calculated at the corresponding cell centers using the Green-Gauss formulation :

$$\nabla\vec{V} = \frac{1}{D}\int \vec{V}\vec{n}\mathrm{dS} \qquad (4.16)$$

Which in the Cell-Centered scheme takes the form:

$$\nabla\vec{V}_I = \frac{1}{D}\sum_{J=1}^{Nf}\frac{1}{2}(\vec{V}_I + \vec{V}_J)\vec{n}_{IJ}\Delta S_{IJ} \quad (4.17)$$

In equations (4.14) and (4.15) the function Ψ is a limiter function which reduces the gradients $\nabla\vec{V}_I, \nabla\vec{V}_J$. Limiter functions are widely used in compressible solvers in order to ensure convergence in areas with strong gradients. In our case, the Venkatakrishnan limiter is used due to its good convergence properties.

4.2.2 Convective Fluxes

The discretization of the convective fluxes can be based on central, flux-vector or flux-difference schemes. Central schemes calculate the convective fluxes across faces as the arithmetic average of the values obtained at the two sides of the face plus an artificial dissipation term added to enhance stability. Flux-vector schemes are based on upwinding which respects the direction of propagation of waves. Finally, flux-

difference schemes calculate convective fluxes at cell faces solving the Riemann problem for the Left and Right states defined on the face.

The present work uses Roe's approximate Riemann solver, which is a flux- difference scheme. Roe's scheme consists of constructing the convective flux as a sum of wave contributions:

$$(\vec{F_c})_{I+1/2} = \frac{1}{2}[\vec{F_C}(\vec{V_R}) + \vec{F_C}(\vec{V_L}) - |A_{ROE}|_{I+1/2}(\vec{V_R} - \vec{V_L})] \quad (4.18)$$

Where the Left and Right states ($\vec{V_L}, \vec{V_R}$) are calculated using (4.14) and (4.15). The Roe matrix A_{ROE} has the same form as the convective flux Jacobian but instead of formally averaged values, the following Roe-averaged variables are used.

4.3 Temporal Discretization

Temporal and spatial discretization are done separately leading for every control to the following equation:

$$\frac{d(D_I\vec{U_I})}{dt} = -[\sum_{m=1}^{Nf}(\vec{F_c} - \vec{F_v})_m \Delta S_m) - \overrightarrow{(QD)_I}] \quad (4.23)$$

Temporal discretization can be either explicit or implicit. Explicit methods use the $\vec{U}^n$ known solution and march in time using the residual $\vec{R}^n$ to obtain solution at (t+ Δt). On the other hand the implicit schemes use R($\vec{U}^{n+1}$) = $\vec{R}^{n+1}$ to obtain the new solution and are favored because they allow larger timesteps. Since $\vec{R}^{n+1}$ is unknown, the following linear approximation is used:

$$\vec{R}^{n+1} \approx \vec{R}^n + \left(\frac{\partial \vec{R}}{\partial \vec{U}}\right) \cdot \Delta\vec{U}^n \, , \Delta\vec{U}^n = \vec{U}^{n+1} - \vec{U}^n \quad (4.24)$$

In MaPFlow a backward finite difference scheme is used of various orders for the time derivative.

4.3.1 Steady State Computations

The composed mesh is at first studied in steady state cases. Even when steady state simulations are considered a pseudo-unsteady technique is followed. For steady state simulations 1st order scheme is chosen to march the solution in pseudo-time until convergence is reached. At 1st order, below there is shown the final systems of discrete equations in which the system matrix defines the implicit operator of the scheme:

$$\left[\frac{D_I}{\Delta t_I}+\left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)_I\right]\Delta\vec{U}_I^n == -[\sum_{m=1}^{Nf}(\vec{F}_c-\vec{F}_v)_m\Delta S_m)-\overrightarrow{(QD)}_I]=-\vec{R}_I^n \quad (4.25)$$

In order to facilitate convergence, the Local Time Step technique is used. The time step for steady state calculation can be defined using the spectral radii of each cell. For every cell, a different time step is defined by :

$$\Delta t = CFL\frac{D_I}{(\hat{\Lambda}_c + C\hat{\Lambda}_v)_I} \qquad (4.26)$$

Where $\hat{\Lambda}_c, \hat{\Lambda}_v$ is the sum of convective and viscous eigenvalues over all cell faces.

The convective spectral radii is defined by :

$$(\hat{\Lambda}_c)_I = \sum_{J=1}^{Nf}(|\vec{u}\,IJ\ \cdot \vec{n}\,IJ| + cij)\Delta Sij \quad (4..27)$$

The viscous spectral radii is defined by :

$$(\hat{\Lambda}_v)_I = \frac{1}{DI}\sum_{J=1}^{nF}[max(\frac{3}{3\rho IJ},\frac{\gamma\ IJ}{\rho\ IJ})(\frac{\mu L}{Prl}+\frac{\mu T}{PrT})IJ(\Delta Sij)^2] \quad (4.28)$$

Chapter 5

5. Results

The construction of the zonal interface generation code between two Chimera grids, was based on two square meshes. The major mesh was of dimension 20x20 and the embedded mesh was of dimension 10x10. Both meshes were structured.

5.1 Composed meshes illustrations

In this subchapter the resulted patched meshes are illustrated in cases where the embedded mesh was set different cases. As it shown below, the zonal interface generation was successfully constructed for all the angles as the composed mesh connectivity was successfully read by MaPFlow.

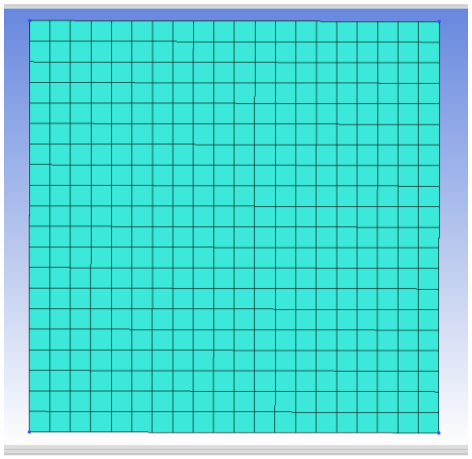


Figure25. Minor mesh at 0 degrees

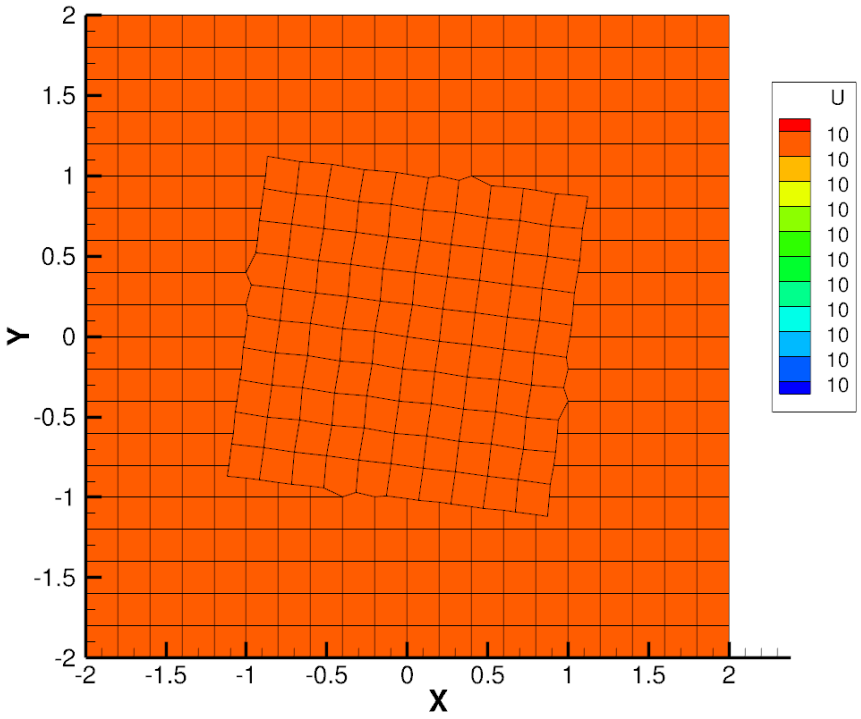


Figure 25. Minor mesh at 7 degrees

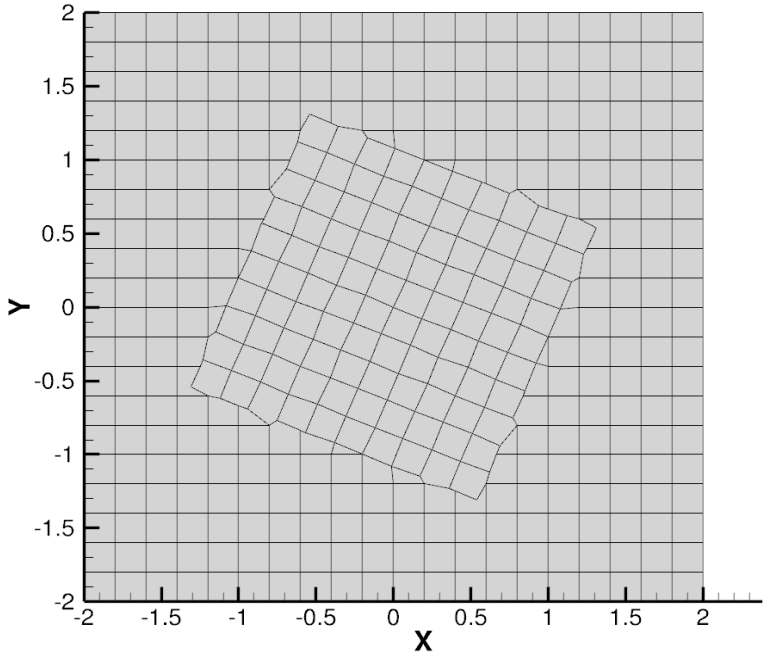


Figure 26.Minor mesh at 25 degrees

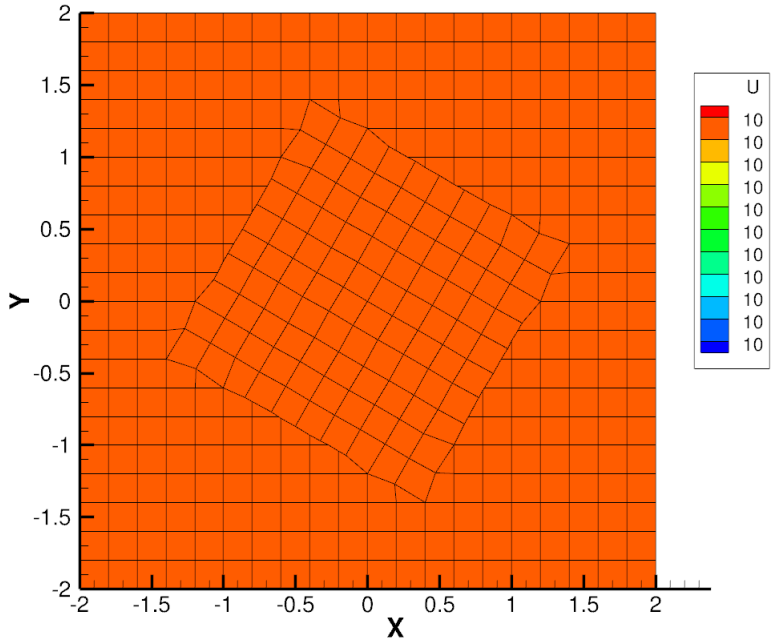


Figure 27.Minor mesh at 30 degrees

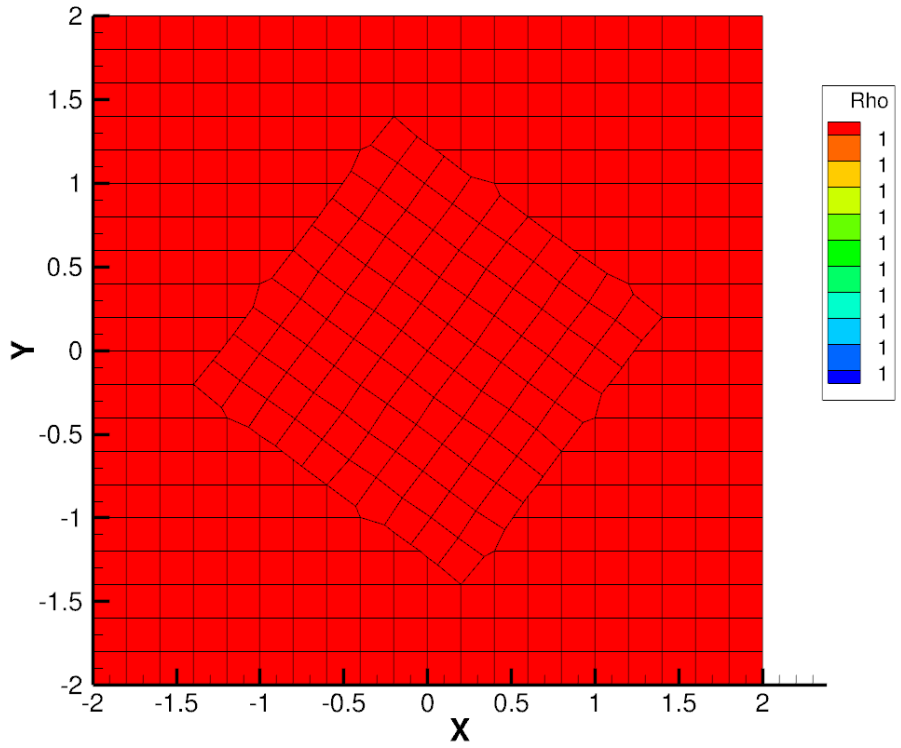


Figure 28. Minor mesh at 36 degrees

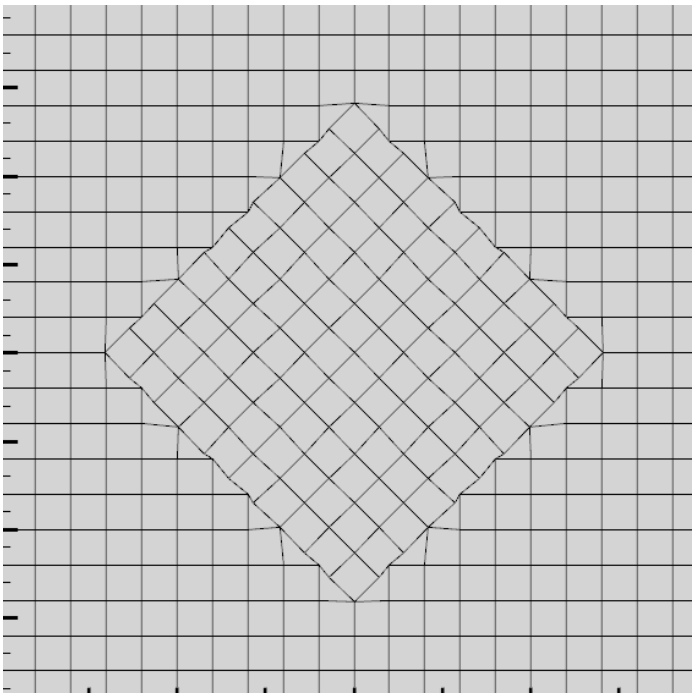


Figure 29.Minor mesh at 45 degrees

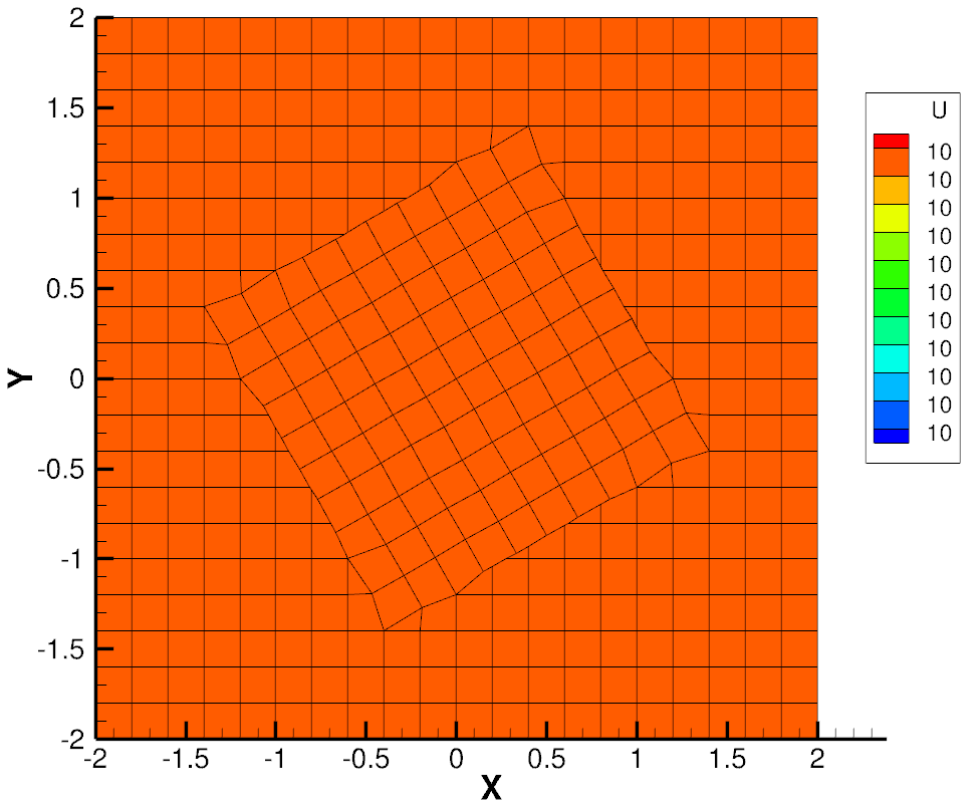


Figure 30.Minor mesh at 60 degrees

The main conclusion to be derived by the above figures is that the zonal interface generation works smoothly, without any noise, as it is noticed in the illustrated flow.

5.2 Airfoil case

The mesh over the airfoil is of length of five chords and it contains of structured and unstructured cells. The Cartesian grid, in which the airfoil mesh is embedded, is of length of 25 chords. The flow field is 2D and Non-Dimensional. In the next two figures, both meshes are illustrated. In the next table, all the inputs for the solution of the flow over the airfoil, are illustrated.

Variable	Value
Mach number	0.25
Free stream density	1
Temperature (K)	300
Starting CFL	1
Final CFL	15
Reconstruction	Piecewise Linear Reconstruction

Table 2.Simulation's Inputs

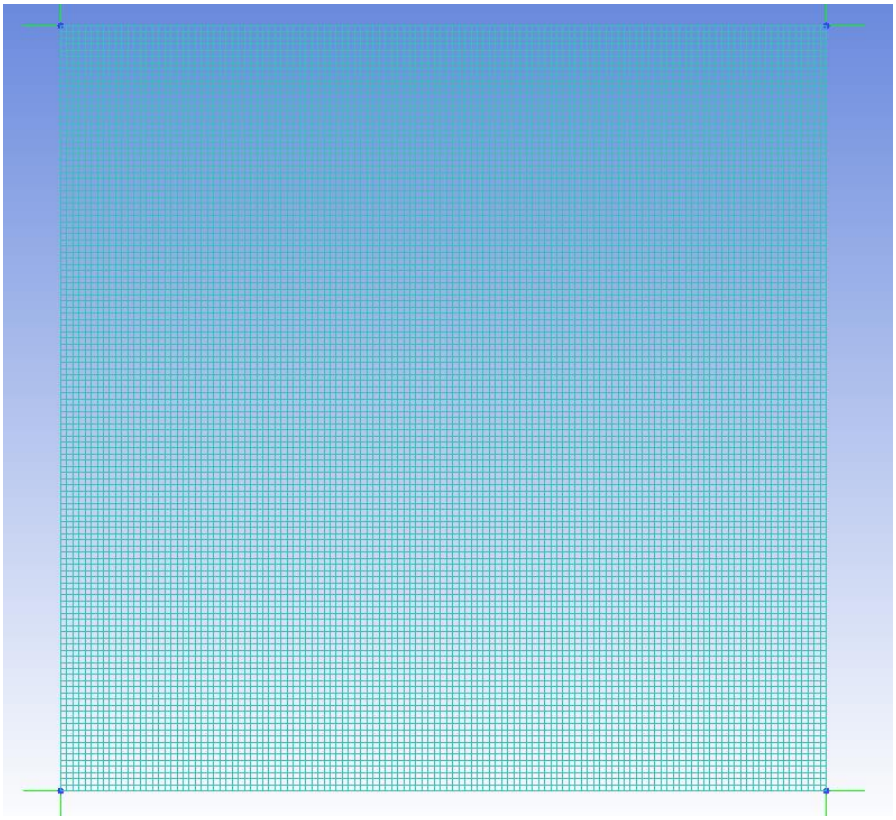


Figure 31. Cartesian Mesh

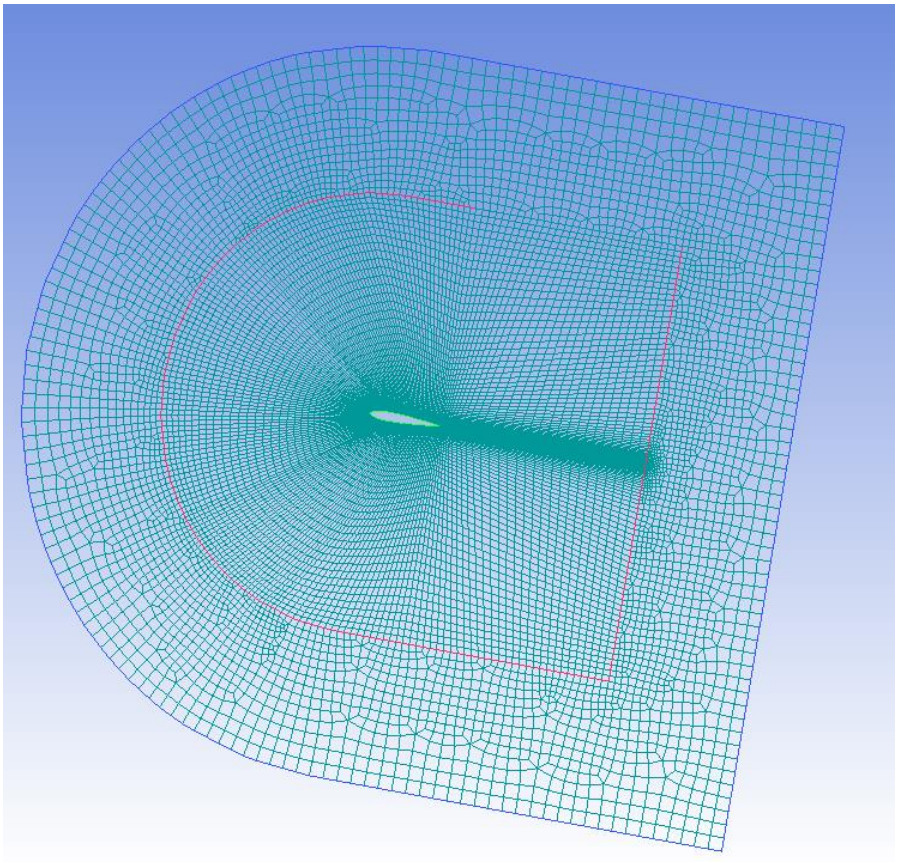


Figure 32. Airfoil Mesh

Next the steps are shown once more in order to unite the two meshes. In the first figure the hole of the major grid is illustrated and in the second the outer boundary of the airfoil mesh along with the nodes of the minor mesh on the boundary, the intersection points, the common nodes and the crosspoints are illustrated.

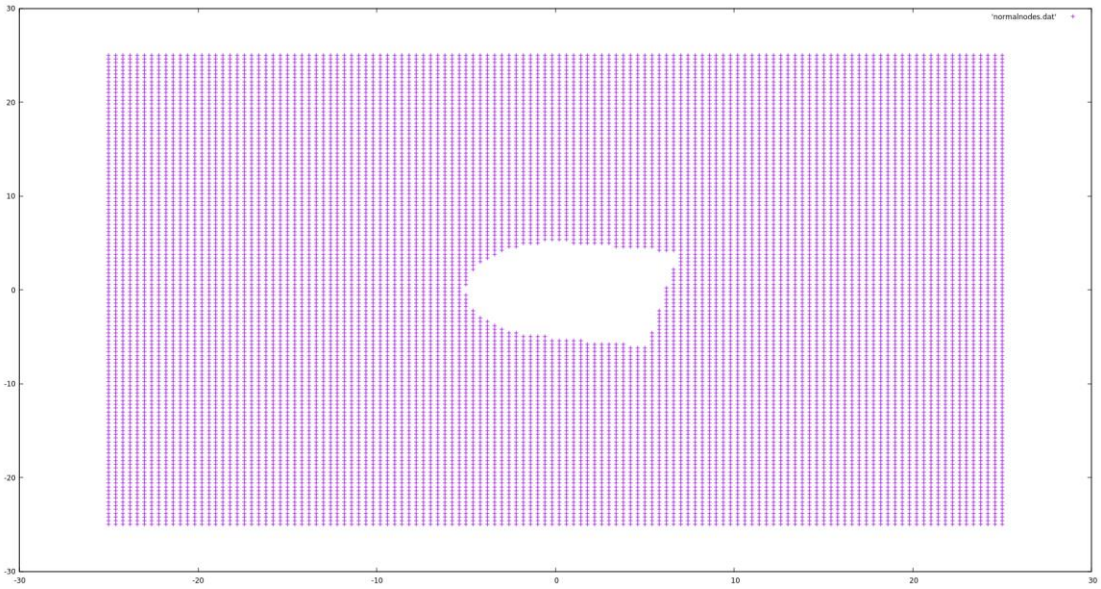


Figure 33. Hole creation in the major grid

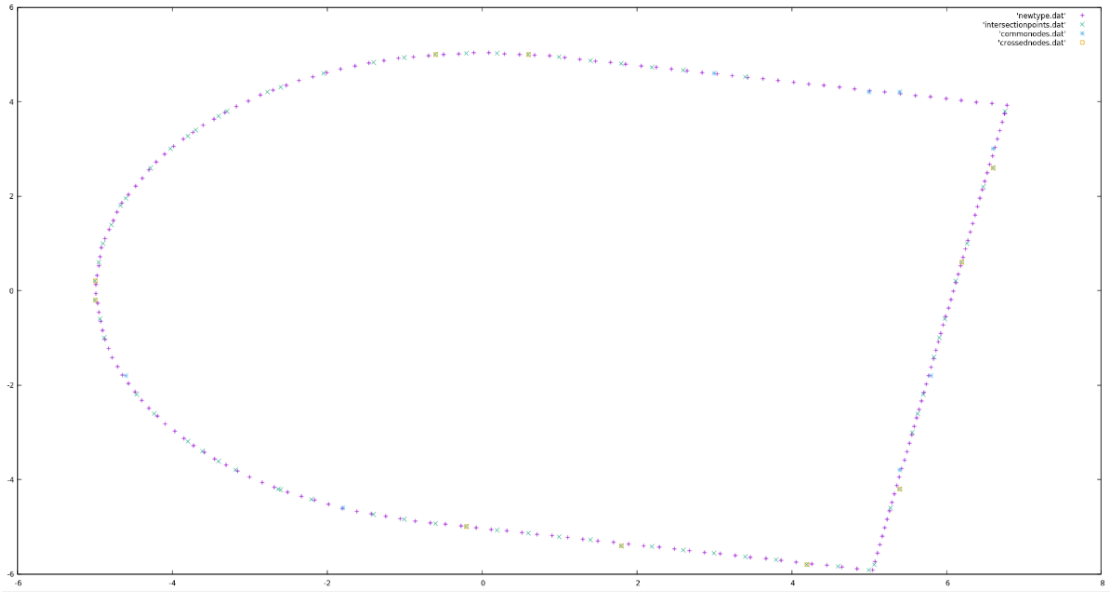


Figure 34. Outer boundary of the airfoil mesh along the new points which are created by the intersection

Next, the patched mesh is shown in figures 35 and 36.

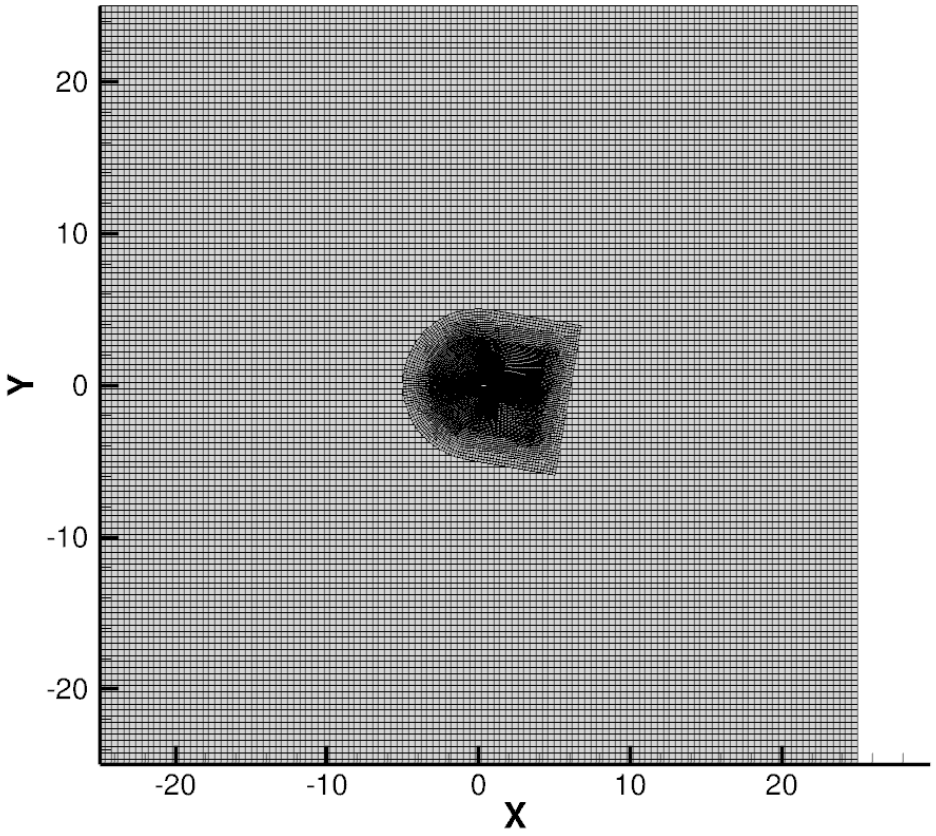


Figure 35The final mesh

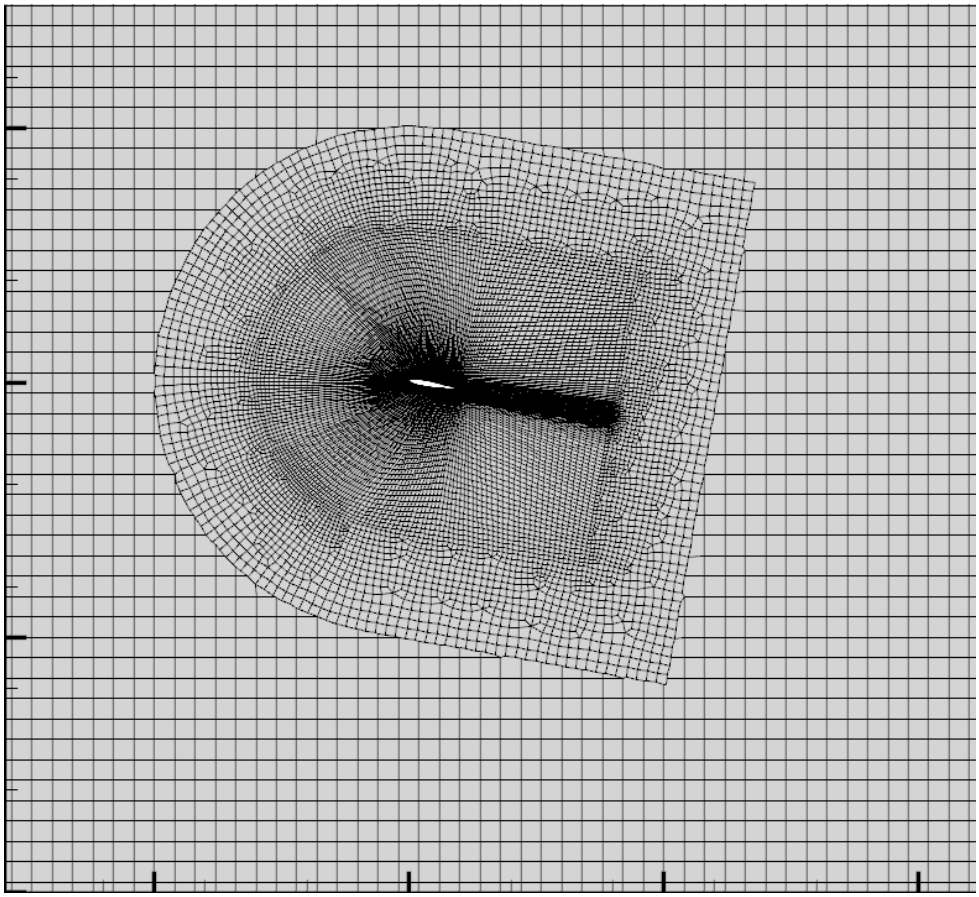


Figure 36. The final mesh on a closer view

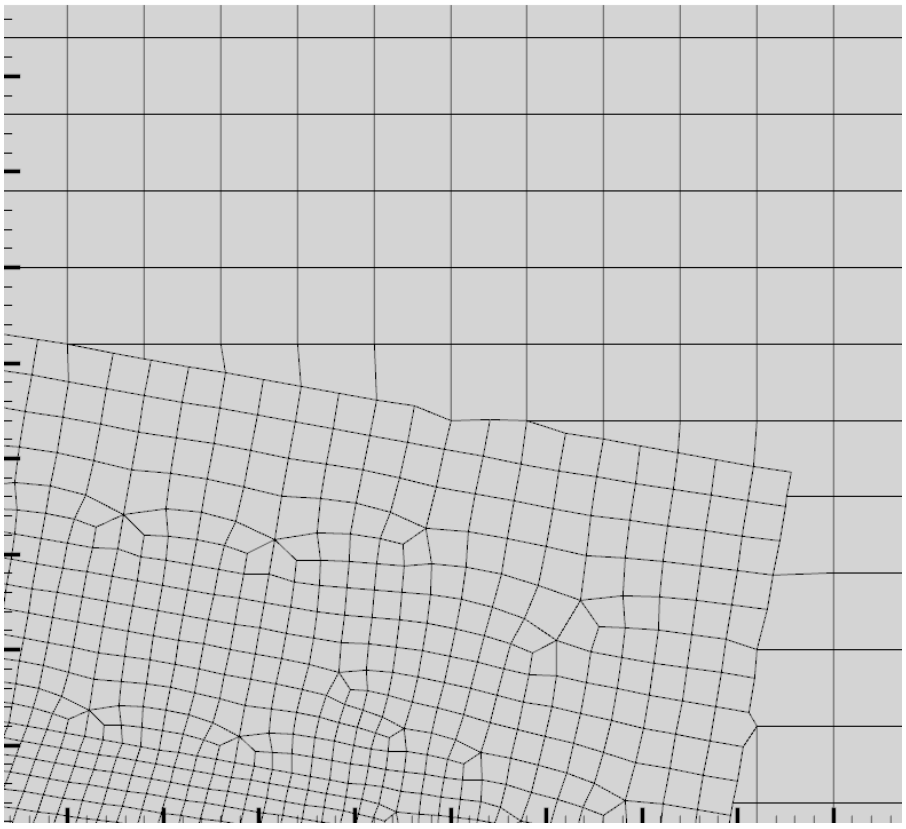


Figure 37.Cut cells illustrated

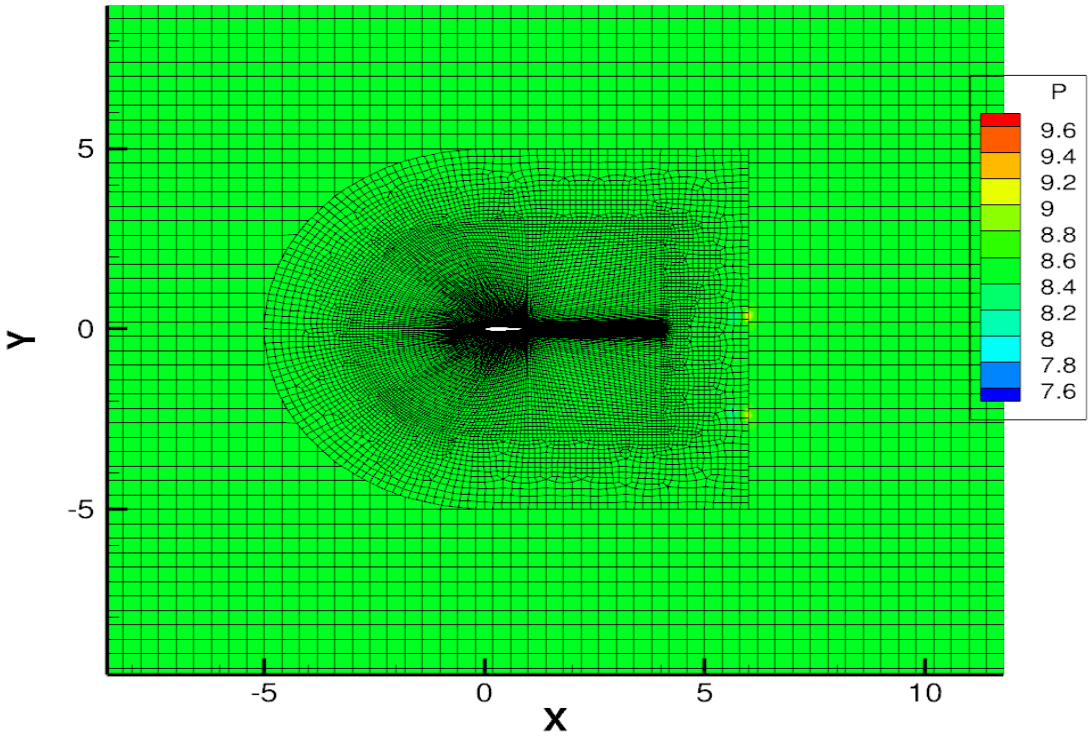


Figure 38. Pressure contour at aoa 0^0, MACH 0.20

Figure 39. Residuals

The previous figure show that the flow simulation with the unified Chimera grid method converges.

Chapter 6

6. Conclusions and next steps

As the first phase of this project is finished the following conclusions have been derived. First of all, the code for mesh composition, is successfully working for structured-structured meshes and structured-unstructured meshes as well. At this point, the mesh composition functions only for two meshes (one major and one embedded). The mesh-composition algorithm was successfully integrated into the eulerian solver MaPFlow.

The advantage of this Chimera grids technique is that global conservation is ensured. Furthermore, the advantages of the flow solver MaPFlow who runs in MPI are fully exploited. It was found that implementing the zonal interface generation algorithm in the case of moving grids is feasible. Regarding stationary Chimera grids, this method prevails over the classic Chimera grid approach.

However, when the problem comes to moving minor grid(s), then the complexity of this method might increase, as the algorithm for the mesh composition would be called in every timestep. Another disadvantage that arises from the above method is the quality of the cut cells which are created in the common boundary of the two meshes. There are cases in which those cut cells are triangles or normal polygons, but this is not ensured for all the cases.

Concerning the future expansion of this study, I would like to suggest the below.

- It was shown that with some expansion to the code which patches the two meshes on the outer boundary of the minor mesh, the moving Chimera grids problem can be treated. Therefore, the expansion is highly suggested.
- Development of quality mesh control for the cut cells. In the case of cut cells with a big aspect ratio or angles with a big deviation from 90 degrees, new cells can be created (e.g. triangular cells).
- Expand the above algorithm so that more than one minor meshes can be simulated which could also be moving.

References

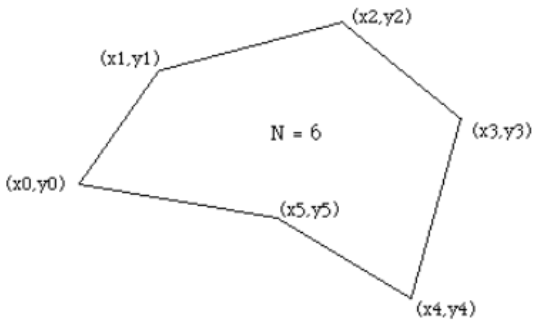
1. Atta E. (1981), *'Component -Adaptive Grid Interfacing'*, AIAA paper 81-0382,1981
2. Benek J.A., Steger J.L, Dougherty F.C, (1983), *'A flexible grid embedding technique with application to the Euler Equations'*, AIAA paper 83-1944,1983
3. Wang Z.J, Yang H.Q ,(1994),' *A unified conservative zonal interface treatment for arbitrarily patched and overlapped grids*', AIAA paper 94-0320
4. Wang Z.J, (1995), *'A fully conservative Structured/Unstructured Chimera Grid Scheme'*, AIAA 95-0671
5. Wang Z.J *(1998) 'A Conservative Interface Algorithm for Moving Chimera) Overlapped) Grids'*, International Journal of Computational Fluid Dynamics, 10:3, 255-265
6. Papadakis Giorgos, (2014),' *Development of a hybrid compressible vortex particle method and application to external problems including helicopter flows',*
7. Bayyuk, S. A., Powell, K. G. and van Leer, B. (1993). 'A Simulation Technique for 2~D Unsteady Inviscid Flows around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry', AIAA Paper No. 93-3391

APPENDIX A

A1. Calculating the area and centroid of a polygon

Area

The problem of determining the area of a polygon seems at best messy but the final formula is particularly simple. Consider a polygon made up of line segments between N vertices (x_i,y_i), i=0 to N-1. The last vertex (x_N,y_N) is assumed to be the same as the first, ie: the polygon is closed.



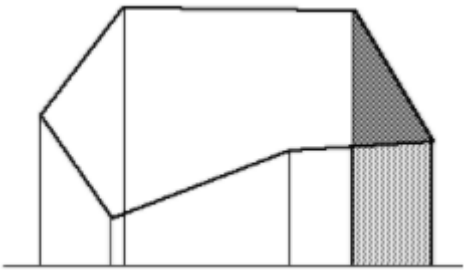
The area is given by

$$A = \frac{1}{2}\sum_{i=0}^{N-1}(x_i y_{i+1} - x_{i+1} y_i)$$

Note for polygons with holes. The holes are usually defined by ordering the vertices of the enclosing polygon in the opposite direction to those of the holes. This algorithm still works except that the absolute value should be taken after adding the polygon area to the area of all the holes. That is, the holes areas will be of opposite sign to the bounding polygon area.

The sign of the area expression above (without the absolute value) can be used to determine the ordering of the vertices of the polygon. If the sign is positive then the polygon vertices are ordered counter clockwise about the normal, otherwise clockwise.

To derive this solution, project lines from each vertex to some horizontal line below the lowest part of the polygon. The enclosed region from each line segment is made up of a triangle and rectangle. Sum these areas together noting that the areas outside the polygon eventually cancel as the polygon loops around to the beginning



The only restriction that will be placed on the polygon for this technique to work is that the polygon must not be self intersecting, for example the solution will fail in the following cases.

Centroid

The centroid is also known as the "centre of gravity" or the "center of mass". The position of the centroid assuming the polygon to be made of a material of uniform density is given below. As in the calculation of the area above, x_N is assumed to be x_0, in other words the polygon is closed.

$$Cx = \frac{1}{6A}\sum_{i=0}^{N-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$Cy = \frac{1}{6A}\sum_{i=0}^{N-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$