

# **Influence of mesh decomposition methods on the simulation of the flow around a wing using Computational Fluid Dynamics**

**MECH3890 Individual Engineering Project**  
*Influence of mesh decomposition methods on the simulation of the flow around a wing using Computational Fluid Dynamics*

Author Name: Jonay Ramón Alamán  
Supervisors Name:

Dr. Marcos Carreres Talens (UPV)

Dr. Daniel Ruprecht (University of Leeds)

Examiner Name: Dr. Qingen Meng

Date of Submission: 26<sup>th</sup> April 2018

MECH3890 Individual Engineering Project

TITLE OF PROJECT

Influence of mesh decomposition methods on the simulation of the flow around a wing using Computational Fluid Dynamics

PRESENTED BY

Jonay Ramón Alamán

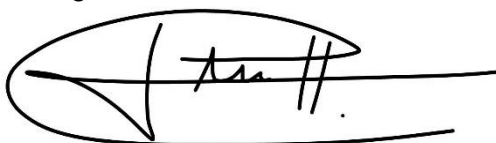
If The Project Is Industrially Linked Tick This Box  
And Provide Details Below

Company Name and Address:

This project report presents my own work and does not contain any unacknowledged work from any other sources.

Signed

Date: 24/04/2018



## Table of contents

Table of contents.....	iii
Acknowledgements .....	v
Abstract.....	vi
List of figures, tables and equations .....	vii
1. Introduction.....	1
1.1. Introduction.....	1
1.2. Aims .....	2
1.3. Objectives.....	2
1.4. Project report layout.....	2
2. Literature review .....	3
2.1. Parallel computing .....	3
2.1.1. High-Performance Computer.....	3
2.2. Mesh partitioning methods.....	4
2.2.1. Geometry-based partitioning.....	4
2.2.2. Graph-based partitioning .....	6
2.3. Influence of the partitioning .....	6
2.3.1. Parameters.....	7
2.3.2. Influence.....	8
2.4. Results quantification.....	10
3. Problem set up and meshing .....	12
3.1. Geometry.....	12
3.2. Boundary conditions .....	12
3.3. Mesh.....	13
3.3.1. Mesh independence .....	14
3.4. Case set up .....	15
4. Results and analysis .....	16
4.1. Partitions.....	16
4.1.1. Interface size .....	16
4.1.2. Load Balancing Factor.....	19
4.2. Results.....	19
4.2.1. Timing.....	19
4.2.2. Speedup .....	22
4.2.3. Comparison between nodes .....	23
5. Influence of the factors.....	24
5.1. Interface cut-edge size and Load Balancing Factor .....	24
5.2. Total CPU Time .....	26

6. Conclusion.....	27
6.1. Achievements .....	27
6.2. Discussions .....	27
6.3. Conclusions .....	28
6.4. Future work.....	28
7. References .....	29
8. Appendix I – Meeting Log .....	31

## Acknowledgements

I would like to thank Dr. Ruprecht for his continued support, interest and help during the development of this project.

I would also like to thank my parents, Enrique Ramón and Gisela Alamán, for their continuous support and their faith in me and my abilities especially in my hardest times. I would like to extend this gratitude to my siblings, David and Celia, and friends.

## Abstract

The increase in usage of Computational Fluid Dynamics software for different aeronautical applications has led to the necessity of continue reducing the time needed for running the simulations. Computational power has begun to reduce its rate of growth in the recent years thus, a different way of reducing those times need to be studied.

As the optimal mesh decomposition method has been found out to be problem dependent, using a 3D model of a wing with a span of 3m, this study uses different methods available in ANSYS Fluent to investigate its influence on the simulations run with the software. It aims to study the scalability of the problem and show a guide on the optimal method for the problem.

The results of this investigation showed that the mesh decomposition methods do not only influence the simulation in the time needed for performing the simulation but also in other ways such as the rate of convergence. METIS showed the best results when working with a low number of processors and faster convergence while specifically-chosen geometry-based methods showed to be able to give as good scalability as METIS for a larger number of processors.

## List of figures, tables and equations

Figure 1.1 Use of CFD at AIRBUS (AIRBUS, 2010).....	1
Figure 2.1 Deviation from the ideal behaviour when increasing the number of partitions (Haddadi et al. 2017).....	4
Figure 2.2 Partitioned domain calculation process (Ierotheou et al., 2000).....	4
Figure 2.3 Resultant partitions using a Cartesian RCB method (Magoules, 2007; ANSYS, 2018).....	5
Figure 2.4 Multilevel algorithm process (Mengoules, 2007).....	6
Figure 2.5 Partitions obtained with Scotch for (a) Haddadi et al. (2017) study with 11 partitions. (b) Zang (2014) study with 4 partitions.....	11
Figure 3.1 NACA 0012 Aerofoil (Airfoiltools, 2018).....	12
Figure 3.2 Resultant geometry and calculus domain with the correspondent coordinate system -Z, red arrow; X, green arrow; and Y, blue arrow-.....	12
Figure 3.3 Resultant mesh in the plane XY, around the aaerofoil.....	14
Figure 4.1 Partitions 0 to 6 obtained when decomposing the domain on 12 partitions. (a) Cartesian Strip, (b) Cartesian Z-Coordinate, (c) METIS, (d) Cartesian X-Coordinate, (E) Cartesian Y-Coordinate.....	17
Figure 4.2. Evolution of the rate of increase in the number of I-faces of the best and worst performing methods.....	18
Figure 4.3. Evolution of the Load Balancing Factors for each method with the number of partitions.....	19
Figure 4.4. Evolution of the partition time for each method.....	20
Figure 4.5. Evolution of the WCT for each method for 500 iterations.....	20
Figure 4.6. Evolution of the WCT for each method with the lower limit of the residuals set at $10^{-3}$ .....	20
Figure 4.7. Evolution of the residuals for each method (a) METIS (b) Cartesian Z-Coordinate (c) Cartesian X-Coordinate (d) Y-Coordinate.....	21
Figure 4.8. Speedup achieved by each method with respect to the base case.....	22
Figure 4.9. Wall clock time evolution between nodes METIS partitions.....	23
Figure 4.10. Wall clock time evolution between nodes Cartesian Strip partitions...	23
Figure 4.11. Wall clock time evolution between nodes Cartesian X-Coordinate partitions.....	23
Figure 5.1. Study on the influence of interface cut-edge size and LBF on WCT for each number of partitions studied (a) 4 partitions (b) 8 partitions (c) 12 partitions (d) 16 partitions (e) 20 partitions (.1) Interface cut-edge size (.2) LBF.....	25

Figure 5.2. Total CPU Time evolution.....	26
Figure 5.3. Total CPU Time evolution (Haddadi et al., 2017).....	26
Table 4.1. Number of Interface faces produced by each method.....	18
Equation 2.1 Total CPU Time equation.....	7
Equation 2.3 Energy consumption equation.....	7
Equation 2.4 Load Balancing Factor equation.....	7
Equation 2.4 Load Balancing Factor equation.....	8
Equation 2.5 Cost function proposed by Vanderstraeten et al. (1996).....	9



# 1. Introduction

## 1.1. Introduction

The development and increase in accuracy of CFD codes has “revolutionised” the development of projects in the aerospace industry becoming one of the “primary tools” for aircraft design (Johnson et al., 2005). As can be observed in Figure 1, in 2010 CFD was frequently or moderately used by Airbus in approximately 83% of the items listed. Items with growing use are mainly limited by the complexity of the geometries and the coupling of aerodynamics with other disciplines such as thermodynamics or acoustics (Abbas-Bayoumi and Becker, 2011). Even though, the program development of an aircraft still “take much too long” (Spalart and Venkatakrishnan, 2016). In fact, Abbas-Bayoumi and Becker (2011) describe time reduction as one main aim for Airbus. In the last thirty years speedup in CFD has been mainly due to the increase in the computing power –a rate of 3.8 every two year- but this is decreasing and it is expected a factor of 1.8 for the period 2013-2025 (Spalart and Venkatakrishnan, 2016). Despite this slowdown in the computing power growth, CFD requirements in order to increase its usage are becoming more demanding even for more complex situations (Spalart and Venkatakrishnan, 2016). Thus, it is necessary to study speedup methods other than trusting the increase in computational power.

In figures, according to Johnson et al. (2003) in 2002 more than 20,000 CFD cases were run in Boeing commercial airplanes. Six years later, according to Professor Jameson (2008), between 50,000 and 100,000 simulations were needed in Boeing for



Figure 1.1 Use of CFD at AIRBUS (AIRBUS, 2010)

developing an aircraft. Also at that time between 4,000 and 6,000 iteration on 288 processors during 12.9 hours were needed for Airbus to obtain only six points of the polar in landing configuration. More complex –but not less common- situations such as the study of ground effect –an increase in lift due to the compression of air under the wing when it is close to a surface (e.g. the runway)- need meshes with 48 million points working on 64 processors for 6 days (Jameson, 2008). Thus, a reduction to a fifth of the initial time, which is achieved in the results, would mean a huge reduction in the simulation time. These calculation-time reductions have a relevant economic

impact as the cost of 3 hours of CFD calculations was estimated by McDonnell Douglas on about \$1,000 (Jameson, 2008).

Even each stakeholder in the industry develops its own CFD codes (AIRBUS, 2018) the case here studied has been as close to real-world problems as possible to make the results on optimal decomposition strategies transferable. As an example, in the sake of fidelity to real-world problems the model used for the study of flow around a wing –k-epsilon model- belongs to the same group of models –Reynolds Averaged Navier-Stokes (RANS)- that has been qualified, from inside the industry, as the most common way to approach these problems (Abbas-Bayoumi and Becker, 2011).

## 1.2. Aims

The aim of this project is the optimization of the mesh partitioning for wing analysis in CFD. The purpose is to find the method that best fits the problem in order to divide it in the optimal number of partitions for which the time is reduced while the costs do not overcome the benefits of using a larger number of processors to run the case. It has been aimed to study the non-user-defined methods which, according to Vanderstraeten et al. (1996), constitute the first steps of a more complex thus, efficient, method.

## 1.3. Objectives

- Define a real-life based case manageable with the time and resources available.
- Select varied partition methods available in ANSYS Fluent.
- Analyse the parameters used in similar studies for evaluating the methods to find the optimal one.
- Study the influence of the methods other than the calculation time.
- Establish recommendation for the selection of a partitioning method.

## 1.4. Project report layout

This project has been produced with the following structure:

- Chapter 2: Analyses the study done on previous works.
- Chapter 3: Illustrates how the geometry was meshed and the case configured.
- Chapter 4: Shows the obtained results for the case with the different methods.
- Influence of the factors: Shows the study of the importance of each of the factors outlined as important in the Literature review.
- Conclusion: Shows the conclusions derived from the study.

## 2. Literature review

### 2.1. Parallel computing

Parallel computing is, according to the Spanish Dictionary of Engineering (RAI, 2018), “computation that uses simultaneously the different processors on a system for a faster resolution of the programmed algorithm”. This is achieved by dividing the problem in a number of subtasks assigning each one to a processor – CPU- to be performed all at the same time. The aim is to reduce the time needed for performing the calculations in traditional –serial- computing where the tasks are performed consecutively (Barney, 2018). When applied to CFD, performing a simulation in parallel means dividing the domain, that is, the region where the flow variables are being solved given some boundary conditions (Atkins, A.G. and Escudier, 2013), into different subdomains and assigning each subdomain to a processor.

Even the domain is divided, flow variables in each partition are not independent of those in the rest of subdomains and the flow field still needs to be solved for the whole domain thus, it is necessary to maintain communication between the different processors to maintain the process shown in Figure 2.2. At that point, a compromise arises because, as explained by Haddadi et al. (2017); even the time spent on calculating the solution for the flow field decreases with the number of processors, the time spent passing data between processors is increased. This, as shown in Figure 2.1, distances the speedup –rate of reduction of the simulation time with respect to a base case- from the ideal behaviour as the number of partitions increases. Eventually, it can lead to the increase in the total time needed with respect to a lower number of partitions. This capacity to reduce calculation time while increasing the number of processors is known as scalability (ANSYS, 2017).

#### 2.1.1. High-Performance Computer

The HPC facilities can be divided, according to the taxonomy proposed by Michael J. Flynn in 1966, in four groups depending on the capability of simultaneously handling instructions (Clevenger et al., 2015). The HPC facility used for this study works with a Multiple Instruction Multiple Data (MIMD) paradigm in which the mesh is split by the master processor among the nodes available (the slaves). Despite this architecture is suitable for the number of processors being used in this study –a maximum of 30- it may show scalability problems when the number of CPUs is increased to hundreds or thousands of processors (Manke, 2001).

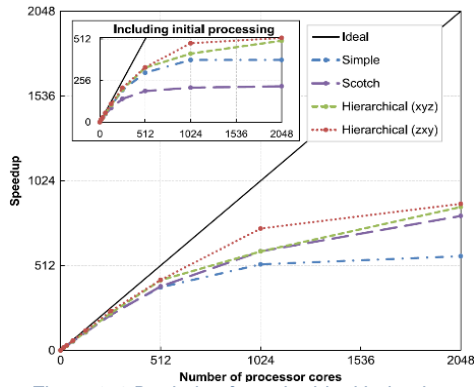


Figure 2.1 Deviation from the ideal behaviour when increasing the number of partitions (Haddadi et al. 2017)

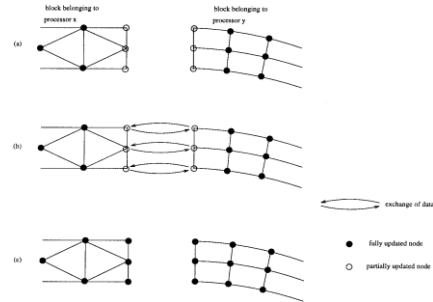


Figure 2.2 Partitioned domain calculation process (Ierotheou et al., 2000)

## 2.2. Mesh partitioning methods

In order to assign a task to each of the previously mentioned CPUs, the domain –the mesh that divides it into discrete elements- has to be partitioned. This can be done through several methods that will result in different partition structures with different communication necessities which will influence the time needed to perform the calculations (Haddadi et al., 2017). As done by Shang (2014) the methods here studied have been selected from two main groups: graph-based and geometry-based methods.

### 2.2.1. Geometry-based partitioning

These methods are the classical approach to mesh partitioning. The mesh is partitioned based on its coordinate system, on the position of the nodes –edge intersections- and centroids of the elements (Magoulès, 2007). They aim to minimize the distance between the elements that will become part of the interface –the face between two partitions- (Shang, 2014). This intention is to assign continuous vertices to the partitions (Magoulès, 2007) –instead of creating divided subdomains-. One of the main advantages is that they are fast partitioning the mesh but, as they do not take into account the “connectivity information” given by the graph they often end up giving large cut edges –high number of elements in the interface- (Magoulès, 2007). Magoulès (2007) recommends them for simulations in which the amount of work each CPU performs solely –or highly- depends on the number of elements of the partition especially in meshes with uniformly distributed edges –such as the one here studied-. The geometry-based methods can be further subdivided depending on the way the cut direction is selected.

#### *Recursive Coordinate Bisection (RCB)*

Is the group that contains the geometry-based methods experimentally analysed. It divides the mesh in two roughly equal halves until the number of

subdivisions equals the desired one (Magoulès, 2007). However, in order to achieve an odd number of partitions –E.g. three- the mesh is divided into two parts, one twice the size of the other, and then the former is divided again ending up with three roughly equal partitions (ANSYS, 2017).

These methods create cutting planes orthogonal to the selected direction thus, in order to optimize the method, the direction selected –if needed to- should be the longest direction of the domain. These methods offer “extremely” short partition times, with partitioned mesh easily parallelizable with the need of little memory. However, for irregular structures, they offer poor solutions. (Magoulès, 2007).

From among the geometry-based RBS methods available in Fluent four have been chosen to be studied.

#### Cartesian Strip

Autonomously finds the largest coordinate in the initial mesh and uses it to create the partitions (ANSYS, 2017).

#### Cartesian X-, Y-, Z-Coordinate

Uses the selected coordinate from the initial mesh for creating the partitions (ANSYS, 2017). For the case being studied the Z-Coordinate case will give the same results as the Cartesian Strip method as it is the longest coordinate of the parent domain. It has been used to find errors and measure uncertainty.

#### Other Geometry-based methods

Such as the Recursive Inertial Bisection (RIB), this uses the principal axes of the domain to perform the partition; the Space-Filling Curve (SFC), which uses curves in more than one dimension; or the Circle Bisection (CB), which uses circles instead of straight lines. They have not been used whether because they are not available in Fluent (SFC and CB) or because for this case their result is the same

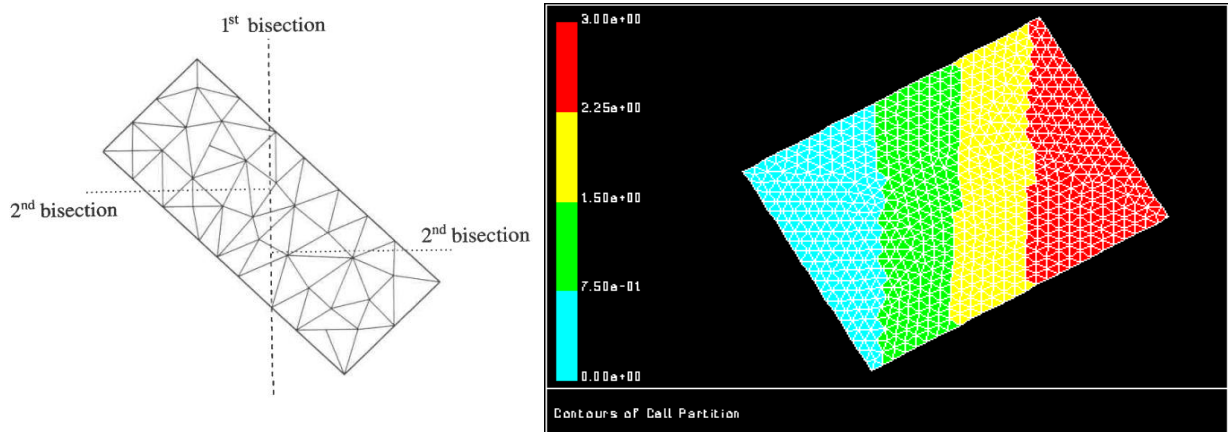


Figure 2.2 Resultant partitions using a Cartesian RCB method (Magoulès,2007; ANSYS, 2018)

than the obtained with RCB (RIB).

### 2.2.2. Graph-based partitioning

Also known as Coordinate-free or combinational methods they do not take into account the proximity of the other nodes but globally the mesh aiming to reduce the cutting edge size. These methods are slower partitioning the mesh but tend to reduce the size of the interface (Magoulès, 2007).

#### *METIS*

Designed by Karypis and Kumar (1999), it uses a “multilevel approach” in which, as shown in Figure 2.4, the elements of the graph are joint to obtain a coarser graph. The latter is divided “carefully” and last the divisions are extended back to the finest graph with the occasional need of a refinement algorithm (ANSYS, 2017; Karypis and Kumar, 1998).

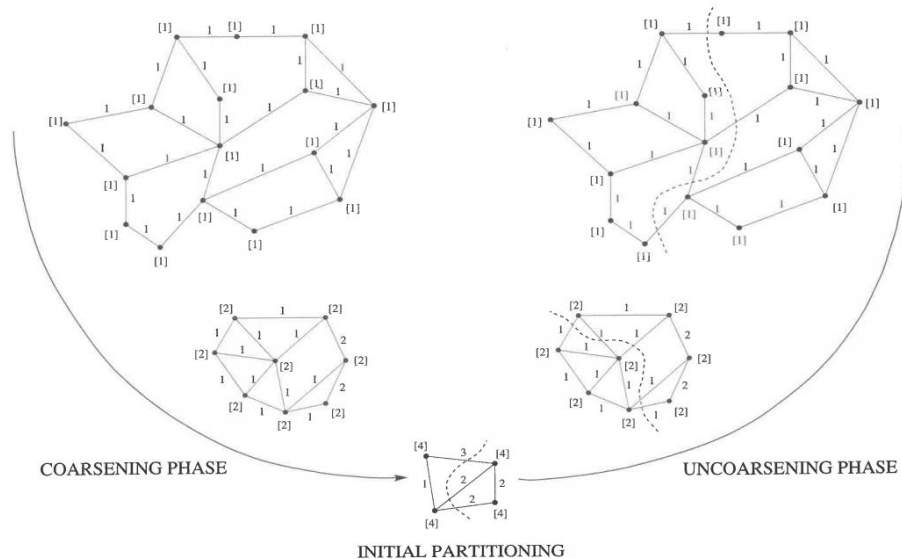


Figure 2.3 Multilevel algorithm process (Mengoules, 2007)

### 2.3. Influence of the partitioning

As mentioned before, the way the mesh is partitioned influences the time needed for calculating the solution of the system but, as shown by authors such as Haddadi et al. (2017) or Shang (2014), they also influence other important parameters such as the partition time or the energy consumption of the HPC –which is expenditure for the user of the method that should be considered-. It is equally important to analyse the causes of these differences in the results and performance of each method. Thus, some parameters have been defined to be able to rank the methods studied.

### 2.3.1. Parameters

**Wall Clock Time (WCT):** Time needed to perform the calculation (Haddadi et al., 2017) –from the moment the simulation is started in the CPU(s) until the last processor has finished-.

**Total CPU Time (TCT):** Aggregated time used by the processors to perform the calculation (Haddadi et al., 2017).

$$TCT = WCT \times n_p$$

*Equation 2.1 Total CPU Time equation*

Where  $n_p$  is the number of processors being used in the simulation.

This parameters quantifies the computational resources being used. In parallel simulation, even the simulation time –WCT- is being reduced with respect to the serial case it does not mean that the total time the resources are being used is reduced.

**Speedup:** Relation between the TCT needed for performing the case with a given number of processor and the time needed for performing that same case in serial –base case- (Haddadi et al., 2017). Other authors such as Jamshed (2015) define it as the ratio between WCT and the base case time but the former definition has been considered to be more illustrative especially when showing the results in graphs.

$$Speedup = \frac{TCT}{T_{base-case}}$$

*Equation 2.2 Speedup equation*

**Partition Time (PT):** Time needed to divide the domain in the different partitions.

**Energy consumption ( $E_c$ ):** Amount of energy required for performing a simulation. It measures the cost of operating the system (Haddadi et al., 2017). It is defined multiplied by an overhead factor to consider the power losses of the system but, for this study, the system is assumed to be ideal as this losses highly depend on the HPC facility.

$$E_c = \frac{CPU\ power}{n_p} * TCT$$

*Equation 2.3 Energy consumption equation*



**Load Balancing Factor (LBF):** Measures how the workload is distributed through the processors. Thus, it represents how much more work one processor will be carrying with respect to the others. That imbalance in the amount of work would mean also an imbalance in the time needed for performing the calculations in a processor and, as the problem is parallelised and the calculations in a processor depend on the development of those in the neighbouring one, imbalances produce delays in the simulation. For the case here studied, as there are not big imbalances between the workload of different elements of the mesh, the load carried by each processor is equivalent to the number of elements it carries. The LBF has been defined as the ratio of the maximum difference in workload between the processors being used and the minimum workload of a processor.

$$LBF = \frac{\max(N_e^k) - \min(N_e^k)}{\min(N_e^k)} * 100$$

*Equation 2.4 Load Balancing Factor equation*

Alternatively, the ratio between the maximum and the minimum number of elements in a partition is also studied.

These factors are based on the LBF proposed by Vanderstraeten et al. (1996) which is simply the ratio between the average and the maximum number of elements in the partitions. It was modified to fully represent the deviation from the ideal case.

### 2.3.2. Influence

There is still lack of uniformity on the parameters that should be studied and the importance that should be given to each of them. Even as long ago as in 1996, Vanderstraeten et al. already diagnosed the underestimation of the importance of the LBF when designing algorithms for partitioning the mesh. According to them, excessive importance was being given to the reduction of the interprocessor communication by reducing the number of interface elements. In fact, in studies such as the one performed by Haddadi et al. (2017) one of the main capabilities emphasized from the automatic method –Scotch- is the reduction of cut-edge sizes while the LBF is not mentioned. Even though, this omission may be due to the fact that the case studied in that paper belongs to one of the three cases for which Vanderstraeten et al. (1996) define as “reasonable” to “prioritize” the reduction in the interface size –when the main communication between nodes is between neighboring subdomains-. The other two cases listed by Vanderstraeten et al. are:



- When the “computational complexity” can be simplified to the nodes or elements that compose the domain. This assumption seems to be made in the ANSYS User Guide (2018) when it is claimed that “create partitions with equal number of cells” is one of the main objectives when partitioning a mesh.

- When the method used for solving the system is not sensitive to the differences in partitions.

As will be explained in the results section, the case here studied may fulfil the first and second conditions.

Due to the problem-dependency of the importance of the factors that should be considered Vanderstraeten et al. (1996) proposed a “cost function” in which each of the factors ( $C_i$ ) influencing the performance of the CFD code solving the partitioned system is given a specific weight ( $\alpha_i$ ). Thus, the cost function to be optimized would be:

$$C = \sum \alpha_i C_i$$

*Equation 2.5 Cost function proposed by Vanderstraeten et al. (1996)*

This function should be optimized after partitioning the mesh with “suboptimal but fast” algorithms which have been studied to suit the problem (Vanderstraeten et al., 1996). That is the step done in this work, with the analysis of which method best suits the problem it will be possible to establish a cost function and further optimize the chosen method.

As a sample three factors ( $C_i$ ) are given:

- **Interface size:** Considered always helpful but not necessarily governing.
- **Load imbalance:** It may get as simple as creating same-size partitions but for more computationally complex cases it may be necessary a more thorough study on the size of the partition assigned to each processor.
- **Subdomain aspect ratio:** Has been observed that this factor highly influences the convergence of domain decomposition.

This last factor, more thoroughly studied by Farhat et al. (1995) is, from the three factors highlighted by Vanderstraeten et al. (1996), the one less explicitly mentioned in the literature despite that Farhat et al. (1995) achieved times up to 1.54

lower than in non-optimized cases. This, as mentioned above, is probably due to the fact that the importance of each factor is problem-dependent.

## 2.4. Results quantification

Two main papers which studied the performance of different partitioning methods on different problems have been analysed aiming to extract some common trends and features to study. These are Shang (2014) that partitioned the mesh obtained for analysing a submarine and Haddadi et al. (2017) whose subject of analysis was a “simple” multiphase case included in OpenFOAM –an open source CFD software- that consisted on a cube.

While the former studies three geometry-based methods –Scotch, PT-Scotch and Zoltan- and two graph-based methods –METIS and ParMETIS-, the latter study analyses only geometry-based methods –Scotch, Hierarchical and Simple-.

Both studies agree that –as can be observed in Figure 2.5 (Haddadi et al., 2017)- the higher differences between methods appear as the number of processors increases –performance is similar for a low number of CPUs-.

Haddadi et al (2017) conclude that more manually-configured methods are preferable –among the geometry-based methods- and recommend the election of the principal axis of the geometry for the partition. Even though, one of the automatic methods analysed by Haddadi et al. (2017) –Scotch- is the second best performing for the case of Shang (2014). Looking at the available images of the obtained partitions in each study –Figure 2.5- it seems that the difference is due to the difference in the case analysed and that, when partitioning the cube, Scotch has generated a huge interface cut-edge and connecting many different processors. In fact, Shang observed that METIS –graph-based- was the best-performing and mentioned that geometry-based “induced an extremely non-uniform mesh distribution”. The poor performance of the most automatic method studied by Haddadi et al. –Simple- is probably due to its “simplicity” as it performs the partitions assigning parts with the same amount of elements to each CPU and the remaining cells –if the ratio between the number of cells and the number of CPUs is not a natural number- are assigned to the CPUs in order (Wang et al., 2012) instead of trying to reduce the interface cut-edge size or the LBF.

Regarding graph-based methods –METIS- Shang recommended it if memory is not a limitation. Even though, he still found some problems with the memory requirements of Scotch.

Haddadi et al. are able to outline some general trends and rules:

- The speedup depends on the solver used but there are similar trends for the different methods.
- The number of cells per core should be kept in the range 50,000-100,000.
- Total computational time grows with the number of partitions.
- The execution time ideal evolution is inversely proportional to the number of partitions. It deviates from this behaviour as the number of partitions increases due to communication overheads. Overhead higher than 25% of ET is not considered acceptable in their study.

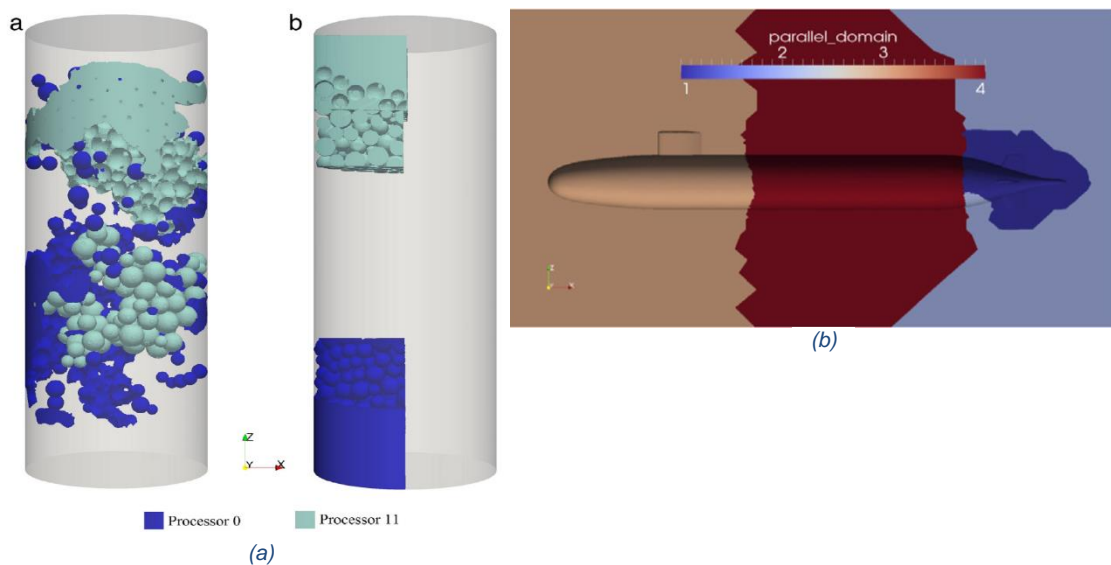


Figure 2.5. Partitions obtained with Scotch for (a) Haddadi et al. (2017) study with 11 partitions. (b) Zang (2014) study with 4 partitions

### 3. Problem set up and meshing

#### 3.1. Geometry

The geometry studied is a NACA 0012 aerofoil –as the one shown in Figure 3.1- extruded 3m in the Z direction

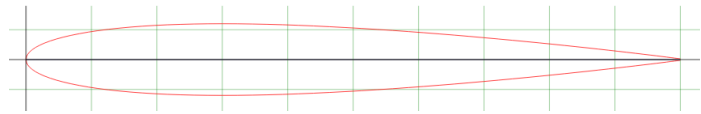


Figure 3.1 NACA 0012 Aerofoil (Airfoiltools, 2018)

–Figure 3.2-. It is a symmetric aerofoil thus, non-cambered –the line that crosses through the middle point of each section from the leading to the trailing edge is a straight line- with a maximum thickness of 12% of the chord –the straight line that joins the leading with the trailing edge- at 30% of it. It was defined with a chord of 1m. The aerofoil was placed at zero angle of attack –the chord line is parallel to the direction of the flow- with the trailing edge lying on the origin of coordinates and the chord on the x-axis.

#### 3.2. Boundary conditions

Based on the results obtained by Craft et al. (2006) the boundaries of the domain were placed at  $x=-1.738\text{m}$  upstream,  $x=0.678\text{m}$  downstream and  $y=\pm 0.5\text{m}$  as they show this was sufficient for the boundaries not to affect the flow around the aerofoil. As the boundaries were set at the same position as Craft et al. did, the Reynolds number –a non-dimensional number that characterizes the flow by relating its density, velocity, viscosity and a characteristic dimension and makes possible to establish the transition from laminar to turbulent flow (RAI, 2018)- was, as they had done,  $4.35 \times 10^6$  –thus, turbulent- with the chord as characteristic dimension. As shown in Equation 3.2, that is equivalent to a free flow velocity of  $63.4967\text{m/s}$  -  $228.5879\text{km/h}$ .

$$Re = \frac{\rho * U_{\infty} * c}{\mu}$$

Equation 3.1 Reynolds number definition

Where  $\rho$  is the density of the fluid;  $U_{\infty}$  is the free stream velocity;  $c$  is the characteristic dimension –the chord- and  $\mu$  the viscosity of the fluid. As the fluid is air at standard atmospheric conditions:

$$\rho = 1.225\text{kg}/\text{m}^3; \mu = 1.789 * 10^{-5}\text{kg}/(\text{ms})$$

(Haynes et al., 2016)

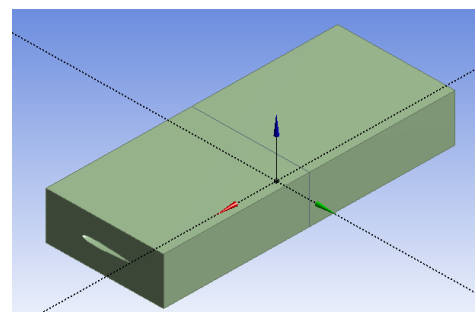


Figure 3.2 Resultant geometry and calculus domain with the correspondent coordinate system –Z, red arrow; X, green arrow; and Y, blue arrow-

$$U_{\infty} = \frac{Re * \mu}{\rho * c} = \frac{4.35 * 10^6 * 1.789 * 10^5}{1.225 * 1} = 63.4967m/s$$

*Equation 3.2 Free stream velocity from Reynolds number*

At the inlet –the domain limit ahead the leading edge- the turbulent intensity – the ratio between the “root-mean-square” of the variations of velocity and the mean velocity of the flow (ANSYS, 2017)- was set at 2% and the hydraulic diameter – diameter of a theoretical circular pipe that generates the equivalent flow conditions at same velocity and friction coefficient as the real, rectangular, diameter (RAI, 2018)- to 0.80023m (Craft et al., 2006).

At the outlet –the domain limit behind the trailing edge- the pressure was set at atmospheric conditions –gauge pressure of 0Pa- turbulent intensity and hydraulic diameter were kept at the inlet values.

For the upper, lower and side limits of the domain, in order to simulate the conditions of free flow around the aerofoil, the condition of symmetry was used. Thus, the model represents an infinite wing –equivalent to a 2D wing, an aerofoil-.

The surfaces of the wing were set as walls with a no-slip condition –the relative velocity of the flow with respect to the wall in the tangential direction is zero (Atkins, M.T. and Escudier, 2018)-.

### **3.3. Mesh**

The first mesh was created with 717,430 elements. The wall of the aerofoil was divided in 100 divisions clustering the elements on the parts of it where the flow varies the most –leading and trailing edges-. The bias factor –ratio between the size of the largest and the smallest elements- was set to 10.

The leading and trailing edges of the wing were divided, along the length of the wing –Z coordinate-, in 100 parts.

The edges of the side walls of the domains –rectangles- were divided into elements of  $5 \times 10^{-2}$ m.

The most important features of the resultant mesh are shown in Figure 3.3.

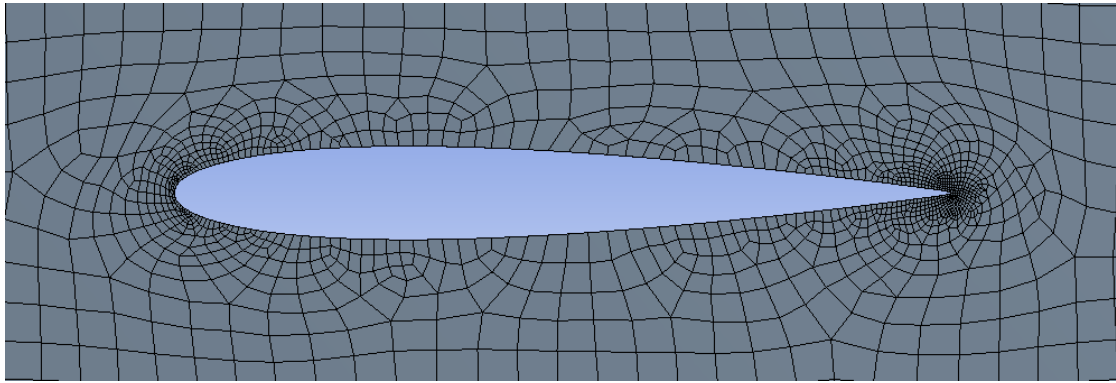


Figure 3.3 Resultant mesh in the plane XY, around the aerofoil

### 3.3.1. Mesh independence

Whenever a case is performed in a CFD code it is necessary to check that the results obtained are independent of the mesh used to obtain them (Gilkesson, 2018). It is necessary to measure the errors due to the size of the elements of the mesh.

For doing this it is necessary to create meshes with different number of elements –thus elements of different sizes as the domain is the same-, run the same simulation in all the meshes and compare the results to verify that the variation is inside a reasonable limit and quantify that variation to take it into account when giving the results.

In this case, as recommended by Gilkesson (2018) three meshes were created with an –almost- constant effective ratio –Equation 3.3- of the number of elements between them. These were of 1.1905 between the finest and medium mesh and of 1.1917 between the medium and coarsest mesh, inside the range recommended ( $1.1 < r < 1.3$ ). Thus, the number of elements was 1.2105 million elements for the finest mesh and 423,867 elements for the coarsest mesh.

$$r_{eff} = \left( \frac{N_1}{N_2} \right)^{\frac{1}{D}}$$

Equation 3.3 Effective ratio equation

Where  $N_1$  is the number of elements in the fine mesh;  $N_2$ , the number of elements in the coarse mesh and  $D$  is the dimension of the problem –three-.

After running the case in all three meshes a relevant variable –pressure at 50% of the chord at the centre of the wing- was extracted in all three cases and compared. Errors –calculated as shown in Equation 3.4- of -1.82% and 3.06% were

obtained between the fine and medium and the medium and coarse meshes respectively.

$$e = \frac{(f_2 - f_1)}{f_1}$$

*Equation 3.4 Error due to mesh*

Where  $f_1$  is the value in the fine mesh;  $f_2$ , the value in the coarse mesh

### **3.4. Case set up**

With the previously explained boundary conditions, the case was configured with the segregated pressure based solver because the flow was expected to be subsonic on the whole domain and stable enough to converge with it. Turbulence was modelled –as done by Craft et al. (2006)- with a k- $\epsilon$  model with wall functions to solve the flow in the boundary layer as there was no interest on studying it in detail.

## 4. Results and analysis

The previously explained case was run in ANSYS Fluent using two different convergence criteria:

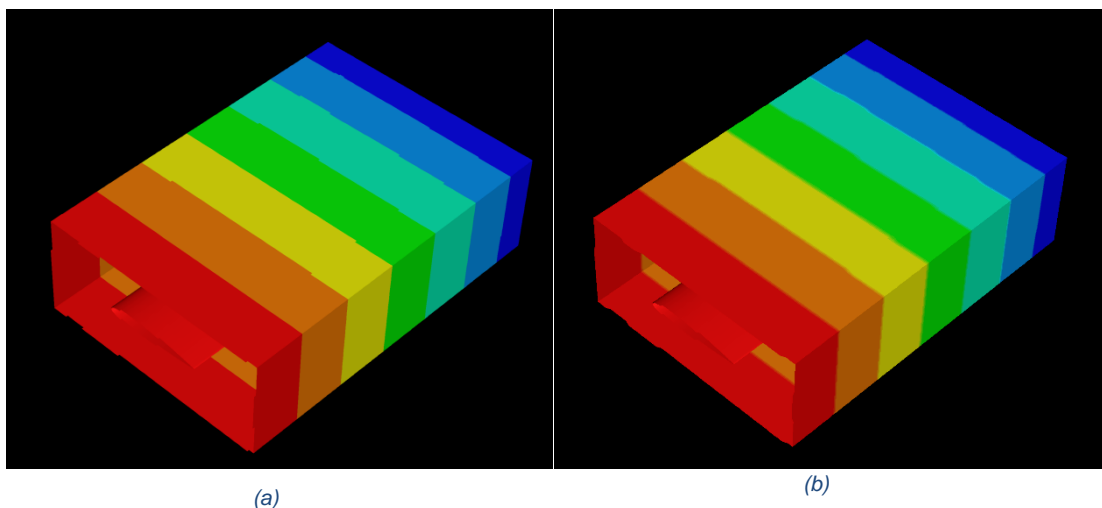
- **Setting a minimum level for the residuals.** That is, automatically ending the calculations when the variation of some selected variables with respect to the previous iteration is under a minimum value –i.e. 0.001-. ANSYS Fluent uses the variation in the continuity equation; in the velocity in the three directions and the variation in  $k$  and  $\epsilon$  –from the turbulence model-. This is the default convergence criteria in ANSYS Fluent.
- **Setting a number of iterations.** The case is run for a number of iterations such that the residuals have dropped to a low level and stay constant. For this case, it was run for 500 iterations.

The case was run in the nodes 2, 7 and 15 of the HPC facility of the University of Leeds. Node 2 is composed of 20 cores –CPUs- of the model E5-2630v4 with 64 Gb of memory. Node 7, of 12 cores of the model Xeon X5660 with 48 Gb of memory. Node 15 is composed of 32 old liquid cooled nodes with 48 Gb of memory. While node 2 was used to undertake the deepest study on the methods, nodes 7 and 15 were used, mainly, to study the influence of the hardware in the results.

### 4.1. Partitions

The geometry was decomposed in 4, 8, 12, 16 and 20 partitions which each method. Figure 4.1 shows, for each 12 partitions case, the first seven divisions – partitions 0 to 6-.

#### 4.1.1. Interface size





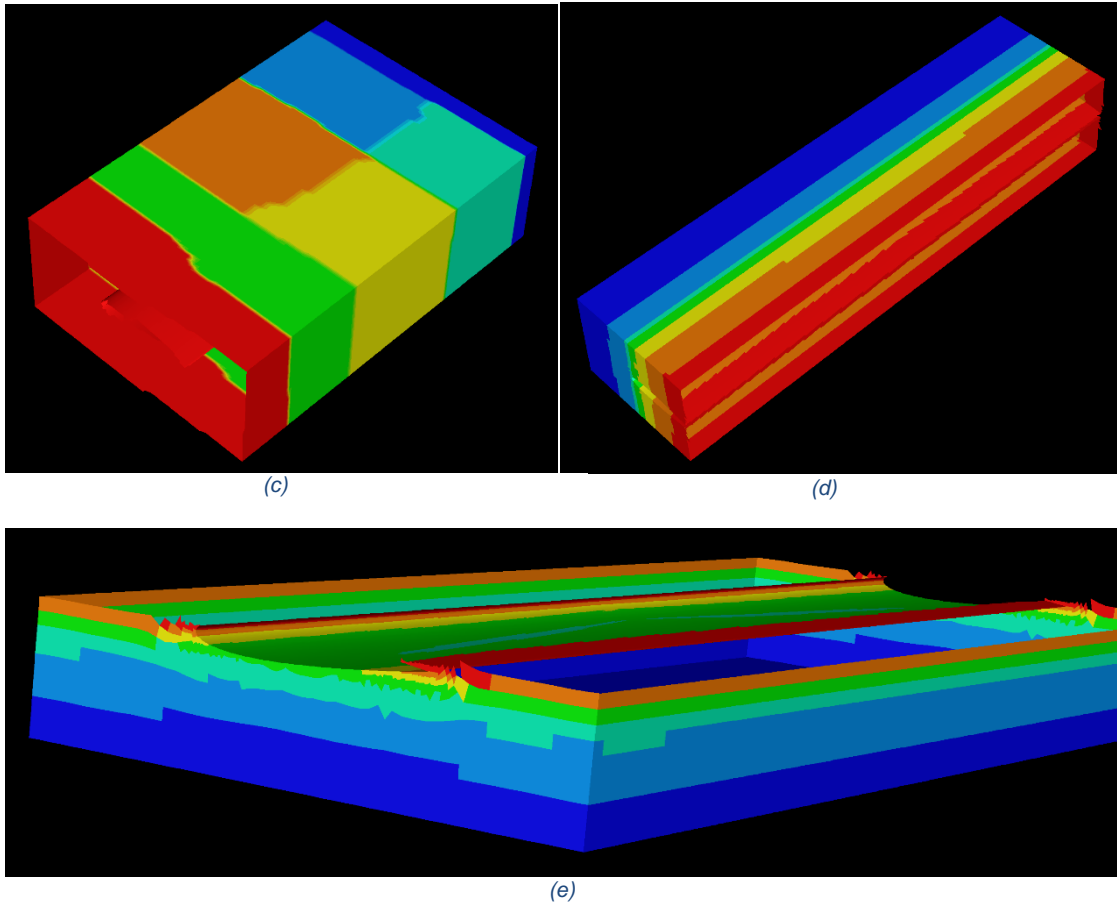


Figure 4.1 Partitions 0 to 6 obtained when decomposing the domain on 12 partitions. (a) Cartesian Strip, (b) Cartesian Z-Coordinate, (c) METIS, (d) Cartesian X-Coordinate, (E) Cartesian Y-Coordinate

As was explained with Figure 3.2, the Z coordinate is parallel to the longitudinal direction of the domain. As explained previously and can be observed in Figure 4.1 (a) and (b) Cartesian Strip method partitions the domain based on its original longest direction (Z coordinate) creating planes perpendicular to it. Cartesian Z-Coordinate produces, as expected, the same results but, in that case, the direction had been selected by the user. Even this methods produce, at the beginning, the smallest interface cut-edge size achievable by an RCB (geometry-based) method in the first partitions, when increasing the number of partitions it will eventually get to a point in which continuing partitioning in that direction would not produce the smallest cut-edge size. This is because the length of the partition in another coordinate –i.e. X coordinate- would be greater than the length of the partition in the firstly chosen coordinate –i.e. Z coordinate-. For this case, the length of the domain is 2.41m, 1m and 3m in X, Y and Z coordinate respectively thus, dividing the domain into 4 partitions would generate:

- Four  $2.41 \times 1 \text{m}^2$  ( $9.64 \text{m}^2$ ) interfaces when dividing the domain exclusively with the Cartesian Z-Coordinate method.

- Two  $1.208 \times 1 \text{m}^2$  and two  $1.5 \times 1 \text{m}^2$  ( $5.416 \text{m}^2$ ) interfaces when partitioning the domain firstly with the Cartesian Z-Coordinate method and later with the Cartesian X-Coordinate method.

This, obviously, is not that simple when performing partitions with the computer and it depends, for example, on the number and size of element faces that lay on each interface but it shows clearly how the geometry-based methods, despite being less memory expensive and needing less time to perform the partitions also have their drawbacks. Even though, it can be observed in Figure 4.1 (a), (c) and (d) and in Table 4.1 that using the longest dimension still is a logical election as their interface size is smaller than the interface size of the shorter ones. Even more, as the length of the direction decreases, the size of the interface increases.

Table 4.1. Number of Interface faces produced by each method

METHOD	NUMBER OF PARTITIONS				
	4	8	12	16	20
METIS	13759	33066	46756	53658	66162
CARTESIAN SRIP	18772	47422	74076	100366	126478
Z-COORDINATE	18772	47422	74076	100366	126478
X-COORDINATE	93864	187266	292436	386676	470750
Y-COORDINATE	189812	373936	536612	706474	861084

The differences between partitioning in the largest direction or in the shortest are significant. They represent up to 911.144% of the Cartesian Z-Coordinate interface faces (I-Faces), for the four partitions case. These differences are reduced to 580.817% of the Cartesian Z-Coordinate faces for 20 partitions. This reduction is due, mainly, to the fast increase in the number of I-Faces for the first partitions of Cartesian Z-Coordinate which starts using a non-optimum partition direction. As shown in Figure 4.2 the rate of increase in I-faces for both methods tends to converge.

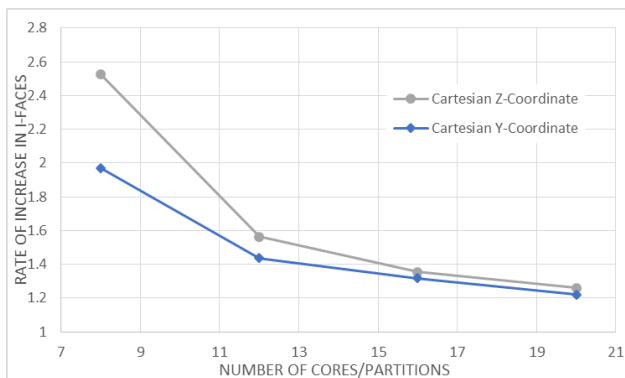


Figure 4.2. Evolution of the rate of increase in the number of I-faces of the best and worst performing methods

In Table 4.1 it can be also observed that METIS is the method which obtains the shortest interface cut-edge size. This is achieved by, as can be observed in Figure 4.1, partitioning the mesh in more than one direction.

### 4.1.2. Load Balancing Factor

As explained previously, it is considered that the LBF plays an important role in the influence of the partitioning methods. A well-balanced partitioned domain is key to reduce the amount of time needed to run the simulations. Even the differences between the best and the worst method on balancing loads are of approximately the same order of magnitude as the differences in the size of the cut-edge interface – between 4372.152% and 565.046%- it is needed a deeper study. By means of Equation 2.4. it was obtained the LBF for each method. Its evolution for each method can be observed in Figure 4.3.

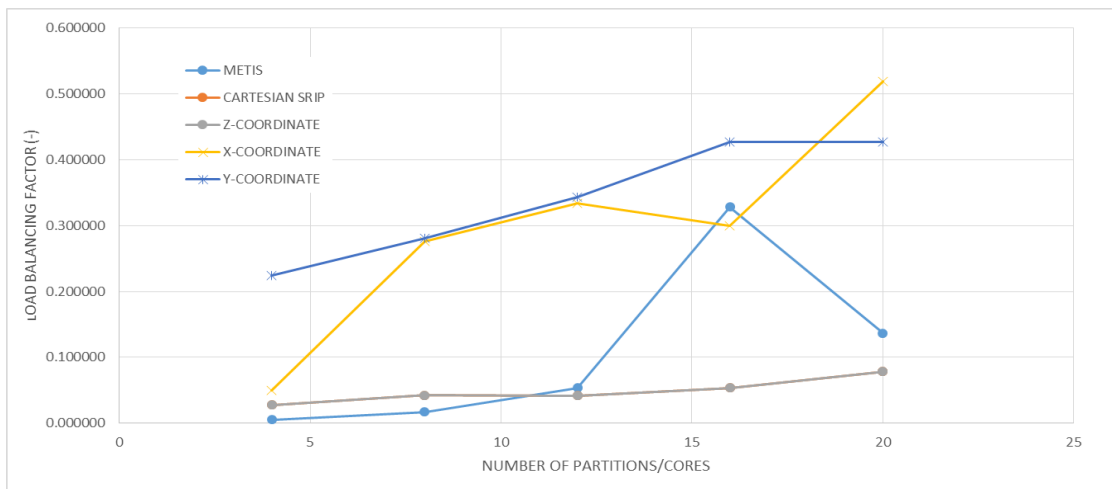


Figure 4.3. Evolution of the Load Balancing Factors for each method with the number of partitions

From Figure 4.3 it can be concluded that, even there are generally-better and generally-worse methods for balancing the loads it is not exclusively dependent on the method –may be on the importance given to it in the algorithm of the method- but also on the number of partitions.

## 4.2. Results

### 4.2.1. Timing

As explained in chapter 2 –Literature review- it is necessary to define a base case in order to study the influence of the methods on the wall-clock time and time-based parameters.

The selected base case was the METIS one partition case performed on one of the CPUs of the second node. Its Wall-Clock time was 1999s –33min and 19s-. As explained previously, it will be used as the reference case to calculate how the partitioning modifies the parameters.

For Node 2 the case was run with 4, 8, 12, 16 and 20 partitions. The following graphs –Figures 4.4 to 4.6- show the time needed for partitioning the mesh and for running the case.

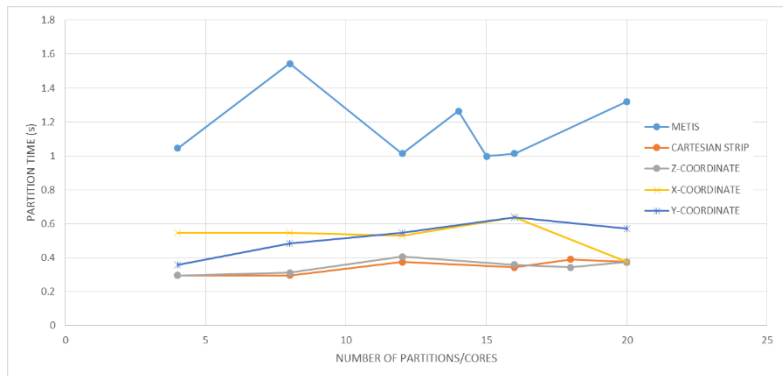


Figure 4.4. Evolution of the partition time for each method

It can be observed how the graph-based method – METIS- performs slower partitions than the studied geometry-based methods.

It is also shown that the partition time depends, not only on the method, but also on the number of partitions being produced.

In Figure 4.5 is shown the WCT needed for running each case on the Node 2 with both, the residuals limited to a value of  $10^{-3}$  and run for 500 iterations.

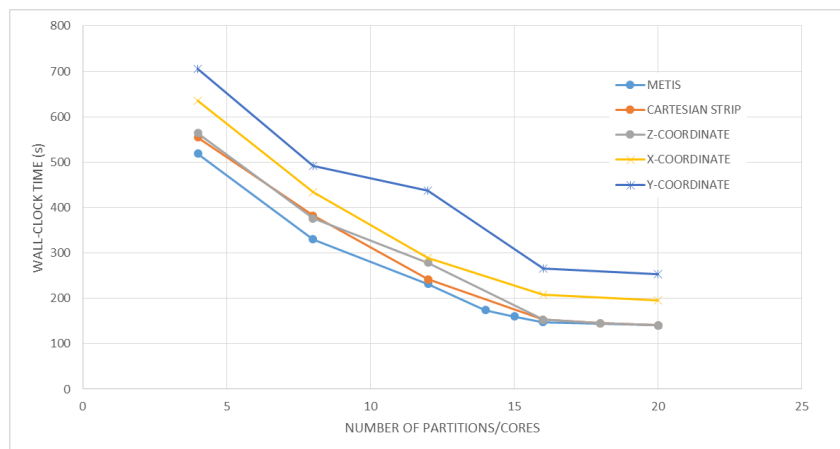


Figure 4.5. Evolution of the WCT for each method for 500 iterations

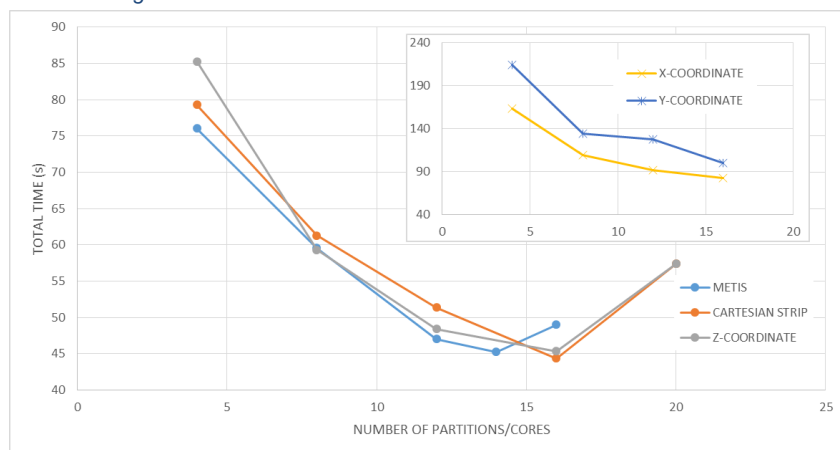


Figure 4.6. Evolution of the WCT for each method with the lower limit of the residuals set at  $10^{-3}$

Looking at the values for the times in Figures 4.4 to 4.6 it can be observed that the time spent on partitioning the mesh played a secondary role on the total time needed for simulating the case.

Both figures -4.5 and 4.6- show the reduction of the WCT needed for running the case –thus, the scalability- for all the methods when initially increasing the number of cores. The greatest differences in WCT were found for the initial number of partitions studied -4 and 8- in which METIS showed the smallest times of all the methods. Even though, all the three fastest methods –METIS, Cartesian Strip and Cartesian Z-Coordinate- converged when increasing the number of processors. METIS started reducing its rate of reduction of WCT earlier than the other two methods. This could be due to, as can be observed in Figure 4.6 bigger communication overheads. In that figure, for the simulations in which the value of the residuals for convergence was set at  $10^{-3}$ , the case was not scalable from a point. It can be observed how the WCT started increasing for METIS, Cartesian Strip and Cartesian Z-Coordinate methods showing that the communications overheads were starting to be higher than the time saved by splitting the case between more partitions. Interestingly, the number of partitions for which the WCT started to increase was different for each method showing the previously mentioned influence

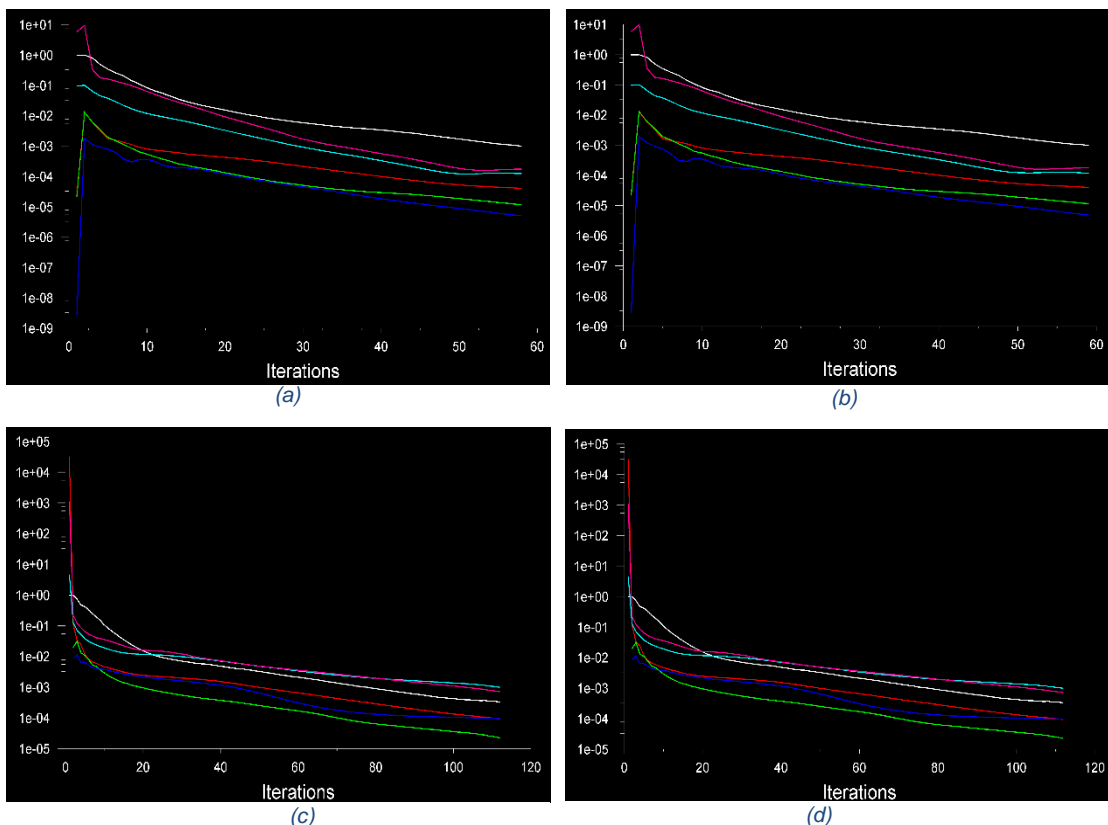


Figure 4.7. Evolution of the residuals for each method (a) METIS (b) Cartesian Z-Coordinate (c) Cartesian X-Coordinate (d) Y-Coordinate

of the method in the convergence of the case. This was analysed by checking the evolution of the residuals –Figure 4.7-.

The trends were observed to be equal for each method independently of the number of partitions –even the number of iterations depended on the number of partitions-. However, when comparing between methods it can be observed in Figure 4.7 how the shape of the graph was similar –with slight differences and with variations in the number of iterations needed- for METIS, Cartesian Strip and Cartesian Z-Coordinate but that the trends where completely different than those of Cartesian X-Coordinate and Cartesian Y-Coordinate –which were identical-.

Cartesian X-Coordinate and Cartesian Y-Coordinate showed larger WCT for every number of partitions tending to stabilize at a bigger value than the other three methods.

#### 4.2.2. Speedup

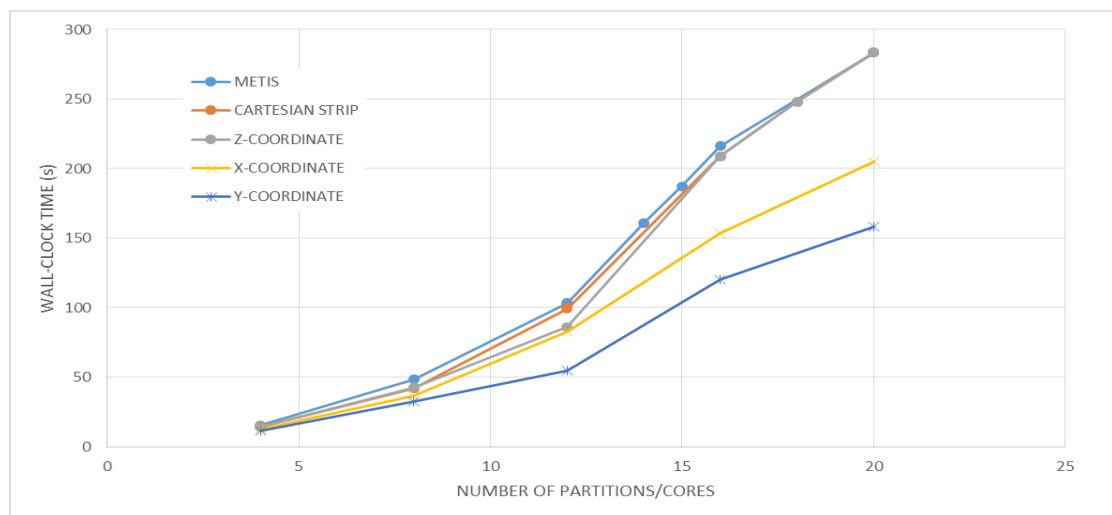


Figure 4.8. Speedup achieved by each method with respect to the base case

When the reduction in time is non-dimensionalised –with the speedup, Figure 4.8- it can be observed how the performance of the methods highly differed. Even for a low number of processors the differences on the speedup obtained with the three fastest methods –METIS, Cartesian Strip and Cartesian Z-Coordinate- and two slowest –Cartesian X and Y-Coordinate- were small, these differences highly increased with the number of processors. As observed in Figure 4.5, these three fastest methods converged to a number with the increase in the number of processors, however, for the speedup the previously described behaviour observed in Figure 4.5 in which the WCT tended to a constant value is not observed in the figure for the speedup.

### 4.2.3. Comparison between nodes

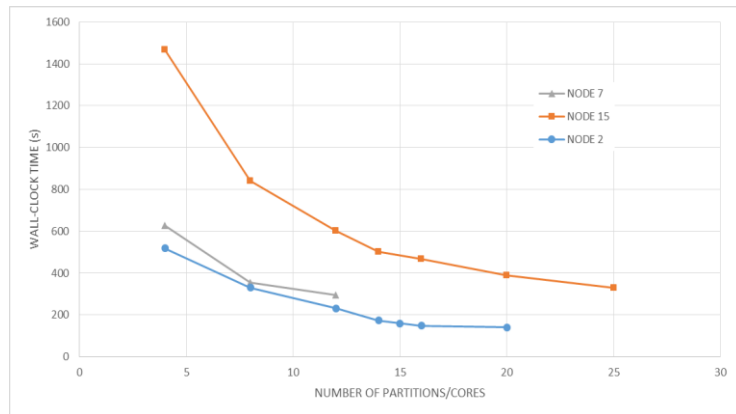


Figure 4.9. Wall clock time evolution between nodes METIS partitions

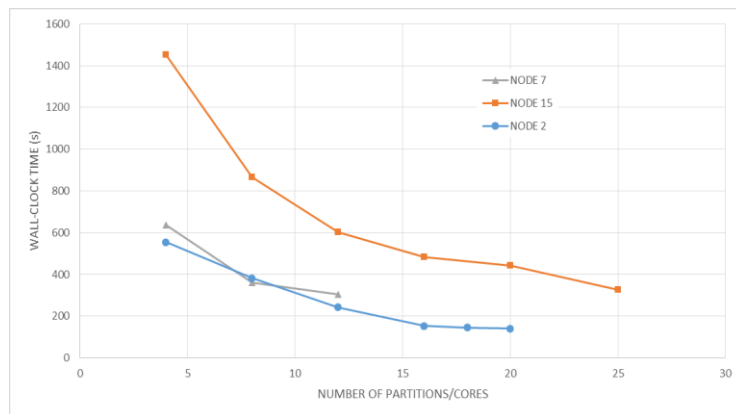


Figure 4.10. Wall clock time evolution between nodes Cartesian Strip partitions

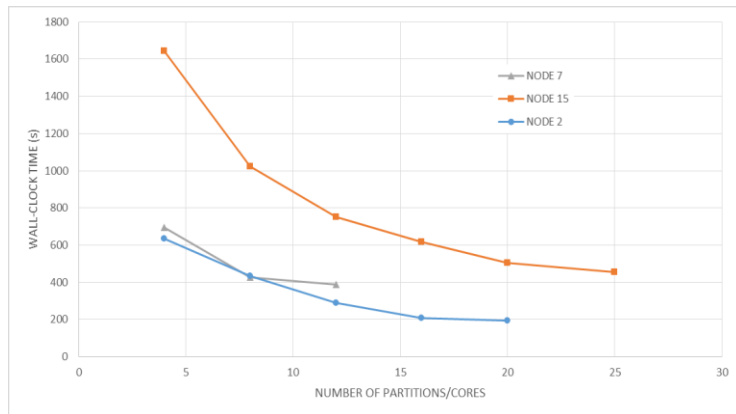


Figure 4.11. Wall clock time evolution between nodes Cartesian X-Coordinate partitions

It can not only be observed the differences in WCT due to the different capabilities of the processors –E.g. Node 15 is composed by old liquid refrigerated systems- but also that the processors slightly modify the trends. Even though, for all the processors tested, the conclusions such as the fastest method are the same. The shapes of the graphs are modified but the same conclusions, such as which is the fastest method, are derived from all the methods.

## 5. Influence of the factors

### 5.1. Interface cut-edge size and Load Balancing Factor

After analysing the trends followed by each variable it is necessary to study the causes of these trends.

As mentioned previously, in the review of literature, the main causes for the performance of a decomposition method are the capacity of balancing the load and the cut edge size of the interface. But it was unclear and problem-dependent the importance of each of the factors. For analysing the importance of each factor in the decompositions of the mesh of a wing Figure 5.1 shows the size of the interface or the LBF together with the WCT needed for performing the simulations.

It can be clearly observed in Figure 5.1 (a) and (b) that both of the factors are in accordance to what has been previously said, with the increase in the LBF –higher imbalances between partitions- and the number of I-Faces the WCT increased. However Figure 5.1 (b), (c) and (d) show how the importance of the size of the interface is much higher than that of the LBF. In Figure 5.1 (b) the LBF obtained with the Cartesian X-Coordinate method is 98.39% of that of Cartesian Y-Coordinate while the number of I-Faces is half of the Y-Coordinate method value. The time difference is reduced from the previous number of partitions while the I-Faces ratio is almost maintained thus, is it the fact of reducing the difference on the LBF seemed the most reasonable cause for the reduction in time difference –given that no that big communication overhead effects where expected for that low number of partitions-. This same conclusion can be obtained from the differences in WCT between METIS and Cartesian Strip observed in Figure 5.1 (c) to (d).

As mentioned in the review of literature, Cartesian Z-Coordinate and Cartesian Strip were used to analyse the possible errors. A mean value of 10.4s for the error was obtained.





Figure 5.1. Study on the influence of interface cut-edge size and LBF on WCT for each number of partitions studied (a) 4 partitions (b) 8 partitions (c) 12 partitions (d) 16 partitions (e) 20 partitions (.1) Interface cut-edge size (.2) LBF

## 5.2. Total CPU Time

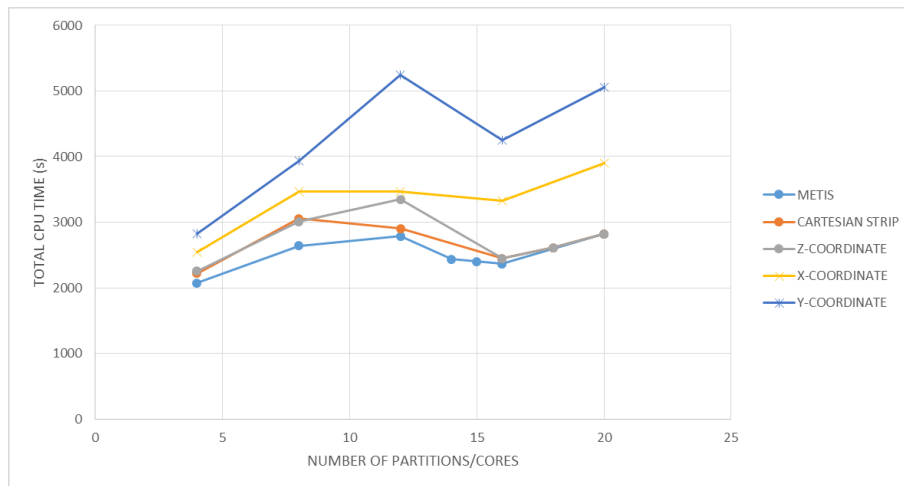


Figure 5.2. Total CPU Time evolution

It can be observed how the TCT followed a generally increasing trend –the local minimums at 16 partitions are higher than the TCT for 4 processors- even the reductions in time for some partitions were able to compensate the increase in the number of CPUs. This unstable behaviour for a low number of processors was also observed by Haddadi et al. (2017) as can be seen in Figure 5.3.

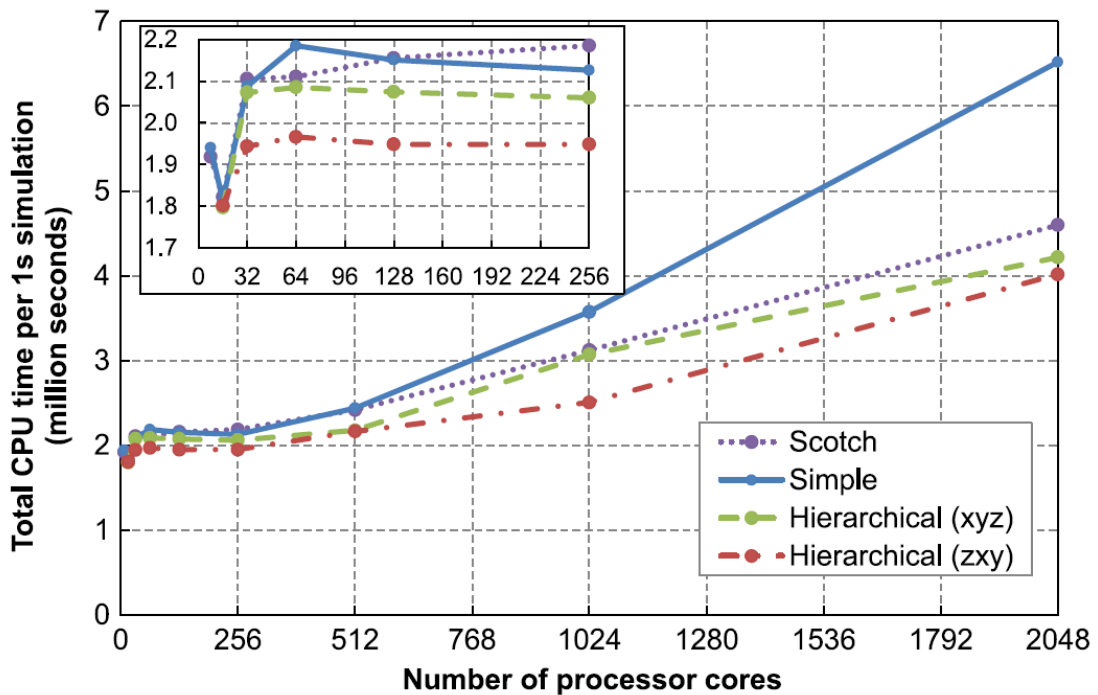


Figure 5.3. Total CPU Time evolution (Haddadi et al., 2017)

## 6. Conclusion

### 6.1. Achievements

With the development of this project, the following items have been achieved.

- The production of partitioned domains of a wing with good scalability for the number of cores studied.
- The comparison of different mesh partitioning methods available in the software.
- The measure of the characteristic parameters of a parallel computation in the wing.
- The identification and measurement of the factors influencing the performance of the models.

### 6.2. Discussions

This project has studied the case-dependent problem of partitioning a mesh designed to analyse the flow around a wing. The results obtained corroborate that mesh partitioning is a case-dependent problem and that there are several factors influencing it but the importance of each is different. It has shown how in a problem with a so much dominant coordinate as this (z-coordinate) performance of geometry-based models can get as good as graph-based. The study has shown the importance of adequately selecting the method for fitting the problem –time differences between the geometry-based methods studies-. The importance of studies like this can be observed in the results where reductions of even four times the initial WCT were obtained.

The study has set a good start for further study even it is acknowledged that differences may arise when varying some parameters –E.g. the dimensions of the domain or the shape of the wing- or the models employed for solving the turbulence. The available hardware should also be beard in mind as –as corroborated while developing this project- it is necessary a system with a single type of CPUs in the nodes for the results to be applicable. Despite those small variations trends in the results and conclusions are expected to be transferable as they have been shown to be –in a more generalised view- from previous studies on less similar problems.

### 6.3. Conclusions

- The partitioning methods here studied show good scalability for the decomposition of the domain.
- The Interface cut-edge size is the parameter most affecting the performance of the mesh partitioning methods.
- The methods do not only affect the time needed for running the simulation but also the convergence of the model and the behaviour of essential evaluation variables such as the residuals.
- The graph-based method studied has shown the best performance for a low number of partitions.
- When the number of partitions was increased, geometric-based methods specifically chosen for the problem showed as good performance as graph-based methods.
- The geometry-based methods studied delay the appearance of communication overheads.
- Mesh-partitioning time represents a truly small contribution to the total time needed when compared with the WCT.

### 6.4. Future work

The work that could be developed in the future from this study would be the introduction of new features to the geometry. Convert it into a 3-D wing for the study of the wingtip vortices and set the basis for the introduction of the fuselage of the aircraft in the model.

Another line of investigation which could be started from this work would be, as proposed by Vanderstraeten et al. (1996) the development of more complex user-defined partitioning methods.

## 7. References

- Abbas-Bayoumi, A. and Becker, K. 2011. An industrial view on numerical simulation for aircraft aerodynamic design. *Journal of Mathematics in Industry*. **1**(1), p10.
- AIRBUS. 2010. *2010 use of CFD at Airbus*. An industrial view on numerical simulation for aircraft aerodynamic design: Adel Abbas-Bayoumi.
- AIRBUS. 2018. *A partnership for next-generation computational fluid dynamics*. [Online]. [Accessed 07 April 2018]. Available from: <http://www.airbus.com/newsroom/news/en/2018/01/a-partnership-for-next-generation-computational-fluid-dynamics.html>
- ANSYS. 2017. *ANSYS DOCUMENTATION - ANSYS USER GUIDE*.
- Atkins, A.G. and Escudier, M.P. 2013. *A dictionary of mechanical engineering*. Oxford: Oxford University Press.
2018. s.v.
- Barney, B. 2018. *Introduction to Parallel Computing*. [Online]. [Accessed 10 April 2018]. Available from: [https://computing.llnl.gov/tutorials/parallel\\_comp/](https://computing.llnl.gov/tutorials/parallel_comp/)
- Clevenger, L.A., Eng, H., Khan, K., Maghsoudi, J. and Reid, M. 2015. Parallel Computing Hardware and Software Architectures for High Performance Computing. *Proceedings of Student-Faculty Research Day, CSIS, Pace University*. pp.D7.1-D7.7.
- Craft, T.J., Gerasimov, A.V., Launder, B.E. and Robinson, C.M.E. 2006. A computational study of the near-field generation and decay of wingtip vortices. *International Journal of Heat and Fluid Flow*. **27**(4), pp.684-695.
- Gilkeson, C. 2018. *MECH5770M: Computational Fluid Dynamics Analysis. Lecture 7: Verification & Validation*. Unpublished.
- Haddadi, B., Jordan, C. and Harasek, M. 2017. Cost efficient CFD simulations: Proper selection of domain partitioning strategies. *Computer Physics Communications*. **219**, pp.121-134.
- Haynes, W.M., Lide, D.R. and Bruno, T.J. 2016. *CRC handbook of chemistry and physics: a ready-reference book of chemical and physical data*. 97th /-in-chief W.M. Haynes ; associate, David R. Lide, Thomas J. Bruno. ed. Boca Raton: CRC Press.
- Jameson, A. 2008. Computational Fluid Dynamics and Airplane Design: Its Current and Future Impact. In: *22 February 2008, University of Cincinnati*.
- Johnson, F.T., Tinoco, E.N. and Yu, N.J. 2005. Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle. *Computers & Fluids*. **34**(10), pp.1115-1151.

- Karypis, G. and Kumar, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*. **20**(1), pp.359-392.
- Magoulès, F. 2007. *Mesh Partitioning Techniques and Domain Decomposition Methods*.
- Manke, J.W. 2001. Parallel computing in aerospace. *Parallel Computing*. **27**(4), pp.329-336.
- Diccionario Español de Ingeniería*. 2018. s.v.
- Shang, Z. 2014. Impact of mesh partitioning methods in CFD for large scale parallel computing. *Computers & Fluids*. **103**, pp.1-5.
- Spalart, P.R. and Venkatakrisnan, V. 2016. On the role and challenges of CFD in the aerospace industry. *The Aeronautical Journal*. **120**(1223), pp.209-232.
- Vanderstraeten, D., Farhat, C., Chen, P.S., Keunings, R. and Ozone, O. 1996. A retrofit based methodology for the fast generation and optimization of large-scale mesh partitions: beyond the minimum interface size criterion. *Computer Methods in Applied Mechanics and Engineering*. **133**(1), pp.25-45.
- Wang, M., Tang, Y., Guo, X. and Ren, X. 2012. Performance analysis of the graph-partitioning algorithms used in OpenFOAM. In: *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), 18-20 Oct. 2012*, pp.99-104.

## 8. Appendix I – Meeting Log

Date of meeting	Summary of discussion	Objectives for next meeting	Supervisor initials
13/10/2017	<ul style="list-style-type: none"> <li>• Presentation</li> <li>• Previous knowledge of any CFD software</li> <li>• Expectations on the project</li> <li>• Explanation of the project</li> <li>• Objectives of the project</li> </ul>	<ul style="list-style-type: none"> <li>• Discuss viability of the cases found by students</li> <li>• Discuss software selection</li> </ul>	
20/10/2017	<ul style="list-style-type: none"> <li>• Discussion about the project decision made</li> <li>• Request of sending papers to tutor for checking the viability of the cases chosen</li> <li>• Intertool focusing</li> </ul>	<ul style="list-style-type: none"> <li>• Decide softwares for being used</li> <li>• Talk about simple simulations performed in other softwares than fluent</li> <li>• Discuss the scoping document</li> </ul>	
13/11/2017	<ul style="list-style-type: none"> <li>• Decision of the case for simple simulations. Cylinder for Von Karman vortex street</li> </ul>	<ul style="list-style-type: none"> <li>• Start the simple cases</li> <li>• Study the capabilities of Fluent PostProcessing tool</li> </ul>	
20/11/2017	<ul style="list-style-type: none"> <li>• Comments on the evolution of the simple case</li> <li>• Proposed variations to the simple case: reduce the mesh size, look for the range at which k-omega is applicable</li> </ul>	<ul style="list-style-type: none"> <li>• Obtain converged cases for the Von Karman vortex street</li> </ul>	
27/11/2017	<ul style="list-style-type: none"> <li>• Comments on the simple cases: Increase velocity, try with the laminar model, try using water, familiarize with the interface for the HPC facility</li> </ul>	<ul style="list-style-type: none"> <li>• Comment the results obtained for the variations in the simple case</li> <li>• Comment any possible problem with the HPC facility</li> </ul>	

13/12/2018	<ul style="list-style-type: none"> <li>• Commentary on the results from the simple cases.</li> <li>• No problems found when using the HPC facility</li> <li>• Commentary on the Scoping document</li> <li>• Recommendation to use a Reference Manager –EndNote-</li> <li>• Problems found with the transient in Fluent</li> <li>• Recommendation of using a coarser mesh near the boundary condition</li> <li>• Recommendation of dedicating more time to literature review.</li> <li>• Comments on the work for Christmas holidays</li> </ul>		
22/1/2018	<ul style="list-style-type: none"> <li>• Definitive decision of how to do the project after the simple cases</li> </ul>	<ul style="list-style-type: none"> <li>• Comment the first steps taken on the project: methods, geometry, mesh</li> </ul>	
29/1/2018	<ul style="list-style-type: none"> <li>• Commentary on the overall evolution of the first steps of the project</li> </ul>	<ul style="list-style-type: none"> <li>• Comments on any problem with the geometry and/or its meshing</li> </ul>	
5/2/2018	<ul style="list-style-type: none"> <li>• Comments on the general ongoing of the project</li> <li>• Recommendation of trying to automatize the process</li> </ul>	<ul style="list-style-type: none"> <li>• Comments on the results</li> </ul>	
19/2/2018	<ul style="list-style-type: none"> <li>• Doubts about problems in the results</li> <li>• Problems with choosing the node for the calculation</li> </ul>	<ul style="list-style-type: none"> <li>• Comments on the performance of the solutions given to the problems</li> </ul>	
1/3/2018	<ul style="list-style-type: none"> <li>• Checking mesh independence</li> <li>• Recommendations on how to approach the project</li> <li>• Doubts</li> </ul>	<ul style="list-style-type: none"> <li>• Comments on the results</li> </ul>	
12/3/2018	<ul style="list-style-type: none"> <li>• Recommendation on validating the results</li> </ul>	<ul style="list-style-type: none"> <li>• Comments on the results</li> </ul>	



	<ul style="list-style-type: none"> <li>• Checking mesh independence</li> </ul>		
19/3/2018	<ul style="list-style-type: none"> <li>• Analysing the obtained results</li> </ul>	<ul style="list-style-type: none"> <li>• Comment on the general evolution of the project</li> </ul>	
6/4/2018	<ul style="list-style-type: none"> <li>• Analysing the obtained results</li> <li>• Comments on the report</li> </ul>	<ul style="list-style-type: none"> <li>• Comment on the general evolution of the project</li> </ul>	
9/4/2018	<ul style="list-style-type: none"> <li>• Analysing the obtained results</li> <li>• Comments on the report</li> </ul>	<ul style="list-style-type: none"> <li>• Comment on the general evolution of the project</li> </ul>	
17/4/2018	<ul style="list-style-type: none"> <li>• Last doubts and analysis comment</li> </ul>		