TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF PRODUCTION ENGINEERING & MANAGEMENT

On the simulation of Steady and Unsteady Incompressible Flows using the Finite Volume approach and Artificial Compressibility concept on hybrid unstructured grids

By

Sotirios S. Sarakinos

A dissertation submitted in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy (PhD)

Supervisor: Dr. Ioannis K. Nikolos, Associate Professor

Chania, Greece, April 2016

To my family and Kleoniki

Abstract

In this study the development and evaluation of a Computational Fluid Dynamics (CFD) code for the simulation of incompressible flows is reported. The code, named *Galatea-I* after the sea-nymph of ancient Greek mythology, utilizes the Navier-Stokes equations, augmented with the artificial compressibility method – which is considered superior to pressure-based methods such as SIMPLE, especially in case of steady state flows – for the simulation of inviscid, laminar and turbulent viscous incompressible flows, of steady or unsteady nature. For the simulation of turbulence the Reynolds Averaged form of the Navier-Stokes (RANS) is used, where the stress tensor in the viscous fluxes vector is analyzed with the Boussinesq assumption in a laminar and a turbulent part. For the evaluation of the turbulent kinematic energy and the turbulent kinematic viscosity the SST turbulence model has been incorporated in the *Galatea-I* solver. The flow model, as well as the turbulence model equations are discretized in space over three dimensional hybrid unstructured grids with a node-centered, Finite Volume (FV) scheme. For the evaluation of inviscid fluxes Roe's approximate Riemann solver is used, while for the calculation of the velocity gradients, which are required for the evaluation of the viscous fluxes, either an element based approach, or a nodal averaging method is used. Free-slip or no-slip conditions are imposed on solid boundaries, while at the inlet or outlet boundaries a characteristics based boundary conditions scheme has been incorporated. Time integration in pseudo-time is performed with an explicit four-stage Runge-Kutta (RK(4)) scheme, while for the time-accurate evaluation of unsteady flows a dual time-stepping scheme is adopted. Two acceleration techniques have been applied in the Galatea-I solver. Firstly, via parallel processing with the domain decomposition approach, where the initial computational grid is divided into smaller sub-domains, each attributed to a single computer core and treated as an autonomous grid with inner boundaries, where information from adjacent grids is passed via the Message Passing Interface (MPI). Secondly, with an agglomeration multigrid method, where a number of consecutively coarser meshes are generated by fusing adjacent control volumes of the finer meshes and evaluation of the governing equations is performed successively on all generated and initial meshes, thus enhancing the convergence rate of the iterative procedures. The performance of the *Galatea-I* solver was assessed with a number of steady and unsteady test cases, demonstrating the capabilities of the proposed methodologies in accuracy and efficiency. While many of the utilized test cases can be characterized as standard for the evaluation of incompressible flow solvers, the proposed code was used against more complex ones, such as the DARPA SUBOFF model and the DLR-F11 aircraft model in high lift configuration. As far as the latter test case is concerned, although it constitutes a test case where traditionally compressible flow solvers with preconditioning matrices are evaluated, the *Galatea-I* solver has generated excellent results.

Keywords: Incompressible flows, RANS equations, Artificial Compressibility, SST, three-dimensional unstructured grids, parallel processing, domain decomposition, MPI, agglomeration multigrid, DARPA SUBOFF, DLR-F11 aircraft model.

Πρόλογος

Σε αυτή τη διατριβή παρουσιάζεται η ανάπτυξη και αξιολόγηση ενός κώδικα Υπολογιστικής Ρευστοδυναμικής (CFD) για την προσομοίωση ροών ασυμπίεστου ρευστού. Ο κώδικας, που ονομάστηκε Galatea-I από τη νύμφη της αρχαίας ελληνικής μυθολογίας, χρησιμοποιεί τις εξισώσεις Navier-Stokes για ασυμπίεστα ρευστά, τροποποιημένες με τη μεθοδολογία της Τεχνητής Συμπιεστότητας (artificial compressibility) – που θεωρείται ανώτερη από τις μεθοδολογίες που χρησιμοποιούν διόρθωση της πίεσης για την συντήρηση της συνθήκης μη συμπίεσης του ρευστού, όπως η SIMPLE, ειδικά στην περίπτωση μόνιμων ροών – για την προσομοίωση ατριβών, στρωτών συνεκτικών και τυρβωδών ροών ασυμπίεστου ρευστού, μόνιμης αλλά και μη μόνιμης κατάστασης. Για την προσομοίωση της τύρβης γίνεται χρήση των Σταθμισμένων κατά Reynolds εξισώσεων Navier-Stokes (RANS), ενώ ο τανιστής των τάσεων που βρίσκεται στο συνεκτικό διάνυσμα ροής αναλύεται με βάση την υπόθεση Boussinesq σε δύο μέρη – ένα στρωτό και ένα τυρβώδες. Για τον υπολογισμό της τυρβώδους κινητικής ενέργειας και της τυρβώδους κινηματικής συνεκτικότητας γίνεται χρήση του μοντέλου τύρβης SST. Για τη χωρική διακριτοποίηση του μοντέλου ροής αλλά και του μοντέλου τύρβης εφαρμόστηκε ένα κεντροκομβικό σχήμα πεπερασμένων διαφορών σε τρισδιάστατα υβριδικά μη δομημένα πλέγματα. Ο υπολογισμός των ατριβών διανυσμάτων ροής γίνεται με τον προσεγγιστικό επιλύτη του Roe για προβλήματα Riemann, ενώ οι μερικές παράγωγοι της ταχύτητας που είναι απαραίτητες για την εκτίμηση των συνεκτικών διανυσμάτων ροής, υπολογίζονται είτε με τη χρήση μιας στοιχειοκεντρικής μεθόδου, είτε με μεθοδολογία ακμών. Οι οριακές συνθήκες που εφαρμόζονται στα στερεά όρια είναι είτε ολίσθησης για ατριβείς ροές, είτε μη ολίσθησης για συνεκτικές, ενώ στα όρια εισόδου και εξόδου του υπολογιστικού χωρίου εφαρμόζονται οριακές συνθήκες που βασίζονται στη μέθοδο των χαρακτηριστικών μεταβλητών. Η ολοκλήρωση των εξισώσεων στον ψευδό-χρόνο γίνεται με μια ρητή μέθοδο Runge-Kutta τεσσάρων βημάτων (RK(4)), ενώ για την προσομοίωση μη μόνιμων ροών έχει ενσωματωθεί στον κώδικα μια διαδικασία δυϊκού χρονικού βήματος. Δύο μέθοδοι επιτάχυνσης έχουν ενσωματωθεί στον επιλύτη *Galatea-I*. Αρχικά, μια μέθοδος παράλληλης επεξεργασίας που βασίζεται στη μέθοδο διαμέρισης πεδίου (domain decomposition), όπου το αρχικό υπολογιστικό πλέγμα χωρίζεται σε μικρότερα υποπεδία, καθένα εκ των οποίων ανατίθεται σε ένα πυρήνα του επεξεργαστή και διαχειρίζεται ως αυτόνομο πλέγμα με εσωτερικά όρια όπου κατάλληλη πληροφορία αποστέλλεται από τα γειτονικά υποπεδία με το πρωτόκολλο MPI. Η δεύτερη μέθοδος επιτάχυνσης βασίζεται στη μεθοδολογία πολυπλέγματος με συσσωμάτωση, κατά την οποία ένας αριθμός από διαδοχικά αραιότερα πλέγματα κατασκευάζονται συγχωνεύοντας γειτονικούς όγκους ελέγχου των πυκνότερων πλεγμάτων και η επίλυση των εξισώσεων ροής και τύρβης γίνεται διαδοχικά σε όλα τα διαθέσιμα πλέγματα διαφορετικής πύκνωσης, βελτιώνοντας κατ' αυτό τον τρόπο το ρυθμό σύγκλισης των επαναληπτικών διαδικασιών. Οι επιδόσεις του επιλύτη *Galatea-I* αξιολογήθηκαν με τη εφαρμογή του σε ένα αριθμό υποθέσεων δοκιμής μόνιμης και μη μόνιμης ροής, παρουσιάζοντας έτσι τις δυνατότητες της προτεινόμενης μεθοδολογίας σε ακρίβεια και αποδοτικότητα. Αν και πολλές από τις υποθέσεις δοκιμής που χρησιμοποιήθηκαν χαρακτηρίζονται ως πρότυπες για την αξιολόγηση επιλυτών ασυμπίεστης ροής, ο προτεινόμενος κώδικας χρησιμοποιήθηκε για την προσομοίωση πιο περίπλοκων προβλημάτων, όπως το μοντέλο

DARPA SUBOFF, και το μοντέλο αεροσκάφους DLR-F11 σε διάταξη υψηλής άντωσης. Όσον αφορά την τελευταία περίπτωση, αν και αποτελεί πρόβλημα για την προσομοίωση του οποίου χρησιμοποιούνται παραδοσιακά επιλύτες συμπιεστής ροής με πίνακες προπαρασκευής για την αντιμετώπιση των χαμηλών αριθμών Mach, ο επιλύτης *Galatea-I* παρουσίασε εξαιρετικά αποτελέσματα.

Λέξεις κλειδιά: Ασυμπίεστο ρευστό, εξισώσεις RANS, Τεχνητή Συμπιεστότητα, SST, τρισδιάστατα μη δομημένα πλέγματα, παράλληλη επεξεργασία, διαμέριση πεδίου, MPI, πολυπλέγμα συσσωμάτωσης, DARPA SUBOFF, μοντέλο αεροσκάφους DLR-F11.

Acknowledgments

With the completion of this study I would like to express my gratitude to those who have supported me throughout this process.

First of all I would like to thank my supervisor Associate Professor Ioannis K. Nikolos for giving me the opportunity to do this PhD and inspired my scientific interest in Computational Fluid Dynamics. Throughout our long collaboration he has been tirelessly supportive, extremely patient and positively motivational.

Secondly, I would like to thank Associate Professor Anargyros Delis and Assistant Professor Dimitrios Rovas for their help as co-advisors, as well as all the members of the approving committee for honoring me by participating at the defense of my dissertation.

Additionally, I would like to thank Dr. Georgios Lygidakis as a co-worker and as a friend for the stimulating discussions, for the long hours of working with me and especially for providing me the methodology for the agglomeration multigrid method and the SST turbulence model.

Finally, I would like to thank my family for their endless support throughout this thesis and my life in general, for believing in me and for being there for me whenever I needed it. Last but not least I would like to thank Kleoniki Nastouli for being with me for all those years, for pushing me to accomplish my dreams and for helping me conquer my fears.

Contents

CHAPTER 1

INTRODUCTION

In this chapter the literature review behind Computational Fluid Dynamics (CFD) topics, especially on incompressible fluid flows is presented, along with the main features presented in this work.

1.1 Setup of a CFD simulation

Computational Fluid Dynamics (CFD) is the field of Fluid Mechanics that deals with the analysis of systems involving fluid motion, heat transfer and associated phenomena, by means of simulations on a computer system [Ver07]. The fundamental basis of most CFD problems is the Navier-Stokes equations that describe the behavior of a single-phase fluid in space and time. The typical procedure for a CFD application contains the definition of the geometry of the problem, the spatial discretization of the geometry and the integration of the equations in time, for obtaining a steady-state solution, or part of a transient simulation.

For the spatial discretization, initially a computational mesh or grid must be defined. The grid is actually a specifically ordered set of points that define the geometry of the problem. There exist basically two types of grids, *Structured* and *Unstructured* grids [Bla01]. With the first type of grids each point can be identified by three indexes *i, j, k*, while the grid cells are quadrilaterals in 2D and hexahedrons in 3D. In unstructured grids the cells have no particular order and the nodes cannot be identified by their indexes. Grid cells can be triangles in 2D, or tetrahedrons in 3D, while more complex grids can contain a mix of different types of elements, such as quadrilaterals in 2D and prisms and pyramids in 3D.

The main advantage of structured grids remains with the fact that the index of all nodes represents a linear address space, allowing the access to a node's neighbors simply by adding or subtracting an integer value to the corresponding index. This property allows faster and easier evaluation of fluxes and gradients, as no complex data structures should be defined to hold topological data of the mesh. The disadvantage for this type of meshes is the difficulty of generating them, especially in case of complex geometries [Bla01]. To remedy this drawback the geometry can be divided into smaller and simpler parts, called blocks that can be meshed more easily. This type of mesh is called *multiblock* [Lee81] [Ros92] [Kue93]. In this case, however, the exchange of flow parameters between blocks has to be considered, as the complexity of the algorithm can be increased. While they present a good alternative to the restricting "one-block" structured grids, the main disadvantage of this type of meshes is the time needed for their generation; an experienced user may require days, or even weeks for very complex geometries.

A special case of structured grids are the so-called *Chimera* grids [Bun85] [Che90], where each geometrical entity is generated separately and then is combined with the others in a way that they overlap each other. The challenge for this method is the accurate transfer of quantities between overlapping meshes where they meet. The main advantage of this technique, over the multiblock approach is that no specific action has to be taken for the grids on each block to meet at their interfaces. However, the conservation of the properties of the governing equations can be difficult to maintain between the different overlapping grids.

The second basic type of grid is the unstructured one [Tho93]. These grids can be generated automatically with sufficient ease on any type of geometry, however complex. While it is still necessary to provide some appropriate parameters for a good grid quality, the required time for its generation is by no means comparable to that needed for a structured one. The main disadvantage of such grids is the lack of the structured topology, leading to the necessity of appropriate data structures inside the flow solver that will aid to the correct computation of fluxes and gradients. As a result, the memory requirements for such meshes can be high, compared to the structured grids. However, despite all the problems that such grids can induce, their usage is increasing and many researchers today apply their CFD codes on such grid topologies.

While initially the first unstructured grids consisted exclusively of triangles or tetrahedrons, these types of grids were proved useful only in the case of inviscid type of flows. In case of viscous flows, where the boundary layer may be (for large values of the Reynolds number) close to the solid wall boundary, the size of the grid has to be extremely fine at those areas. In order to reduce the grid size due to that restriction, prisms with small height to base ratio are used at those areas instead. At the interface of a quadrilateral surface of a prism and a triangular one of a tetrahedron, pyramidal elements are used. The grids that combine more than one type of elements are called mix-grids or hybrid meshes. The use of hybrid meshes was first reported by Nakahashi and Obayashi [Nak87], and Weatherhill [Wea88].

Another decision that has to be made by the researcher is the discretization scheme for the governing equations. The developed methodologies throughout the years are the finite-difference method, the finite volume method, and the finite element one. The first method involves the utilization of the Taylor series expansion for the discretization of the derivatives of the flow variables [Mcd71]. This method was one of the first used for the numerical solution of differential equations. In the field of incompressible flows, the first works presented used the finite difference method for the evaluation of the flow field [Har65] [Cho67a] [Kim85] [Rog89] [Ros91]. The type of grids used in these works is the so-called staggered grids; Cartesian type (structured) grids where the scalar parameters such as pressure, density etc. are stored at the center of a cell, while the velocity and momentum variables are stored at the cell faces.

With the finite volume scheme, the computational grid is discretized into a number of polyhedral control volumes. The spatial derivatives at the right hand side of the governing equations are calculated as a sum of all the fluxes that cross through the faces of the control volume. The accuracy of the method depends on the specific scheme used for the storage of flow variables and the calculation of fluxes. The decision can be made between the cell-centered scheme, where variables are calculated at the center of each grid element and fluxes cross through the elements'

faces, and the node-centered one where the variables are calculated on each node and the fluxes cross through the interface at the middle of the edge connecting this node and any neighboring one. The finite volume scheme is very popular among researchers [Zha00] [Tai03] [Kal05] [Tai05].

The finite element method was originally used for structural analysis. This method uses unstructured grids, exclusively. Within the method, the governing equations have to be transformed from a differential form to an equivalent integral one. This is performed either with the *variational principle* where a physical solution is sought, for which a functional possesses an extremum, or with the *method of weighted residuals*, or *weak formulation*. With the latter the errors of the approximation of the solution are required to have a weighted average of zero over the physical domain. Incompressible flow simulations with the finite element method have been reported in [Tez92] [Tez92] [Gre98].

1.2 Incompressible flow solving techniques

The formulation of incompressible flows is most commonly achieved with the use of the incompressible Navier-Stokes equations [Kwak11], however the "incompressibility" assumption was always difficult to impose, due to the lack of the pressure term in the continuity equation, strengthening in this way the elliptic nature of the equations. The first approach to address this problem was proposed by Harlow and Welch [Har65]. Their method involved the solution of the Poisson equation for pressure to satisfy the continuity equation and called it the marker-and-cell (MAC) method, as they also used marked elements to track the fluid location in case of a free surface. The MAC method constituted the basis for all pressure-based methods for the simulation of incompressible flows, while it can be categorized as a special case of the pressure projection method.

The pressure projection method, first formulated by Chorin in 1967 [Cho67b] involved the evaluation of the flow field in two steps. In the first step an auxiliary velocity field is calculated from the momentum equations with reference to the values of the previous step, while omitting the pressure term, or calculating it with regard to values from the previous times step [Dwy86] [Dwy89]. This calculation may involve several intermediate fields of velocity. In the second step, the pressure is calculated along with the true value of velocity, in an iterative procedure that converges to a divergence free velocity field. Thus, the correction of pressure assures the satisfaction of the continuity equation. The main disadvantage of these methods can be the large amount of time needed for the convergence of the pressure correction procedure, or the solving of the Poisson equation for pressure [Kwa11].

As Chorin's procedure calculates the flow in a number of steps, the method is also characterized as fractional-step method. One of the problems encountered with this method is the boundary conditions that have to be applied at the intermediate steps [Ors86]. A generalized scheme, where physical boundary conditions can be applied was developed by Rosenfeld et al. [Ros91] and Kiris and Kwak [Kir01].

With the major drawback of the MAC method being the requirement for many iterations to solve the Poisson equation for pressure, the following idea grew, that for a steady-state solution the correct pressure is needed only when the procedure has converged; the resulted SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) method simplifies the equation for pressure requiring only a few iterations at each time step [Car73] [Pat72]. Initially, a guessed pressure value is used to evaluate an intermediate velocity value with the momentum equation. The pressure is corrected using values obtained by a simplified momentum equation, involving a function depending on the discretization scheme that was chosen. The method has been used with success for many computations [Doo84] [Van84] [Jan86] [Ach89] [Dav96].

On a different premise, Chorin proposed his "artificial compressibility" method in 1967 [Cho67a], where the incompressible Navier-Stokes equations would be formulated as the compressible ones, as the temporal derivative of an artificial density variable, directly connected to pressure with an "artificial compressibility" parameter, could be added to the continuity equation, inducing a hyperbolic nature to the governing equations. The time parameter of this extra derivative is actually an artificial pseudo-time parameter. The idea behind this approach is that when the equations reach a steady-state the temporal derivative would be equal to zero, and the incompressible equations would be restored. A similar method was proposed by Steger and Kutler [Ste77], who adopted the approximate factorization method by Beam and Warming [Bea76]. While Chorin's method was initially developed for steady-state problems in two dimensions, it was fully extended to three dimensions by Kwak et al. [Kwa86], using generalized curvilinear coordinates. For the simulation of time-accurate flow problems, Rogers and Kwak [Rog89] [Rog90] applied a second derivative of pressure in time to the momentum equations. The new time variable was considered the physical time. For the evaluation of the flow state in a physical time step, the governing equations have to converge to a divergence-free velocity field in pseudo-time. Pan et al. [Pan89] and Taylor et al. [Tay91] developed finite-volume methods for the evaluation of incompressible flows using structured grids in a generalized coordinate formulation. Tai and Zhao [Zha00] proposed a methodology of artificial compressibility, using characteristics for the discretization of the inviscid terms in 3D. Kallinderis and Ahn [Kal05] reported a finite volume method for the artificial compressibility approach where viscous fluxes were calculated using an edge-dual volume method.

Finally, other methods for the evaluation of incompressible flows involve the use of derived variables, such as the stream function and the vorticity. The first approach is used to simulate simple two-dimensional flows, as in order to extend it in three dimensions the velocity vector potential has to be added to the equation, increasing in this way the complexity of the algorithm [Qua13]. For the second method, instead of the momentum equation the vorticity transport equation is used. The vorticity and velocity are calculated by the new equations, while pressure still has to be evaluated with the Poisson equation. While some approaches have been reported in 2D, extension in the three dimensions is limited to simple geometries [Den79] [Haf89].

1.3 Parallel computing

With the advances in computer technology, the means for faster and more efficient simulations were always presented to researchers, as well as industrial users. The idea of a computing machine was present since the ancient times, with devices such as the abacus, or the more complex "*Antikythera Mechanism*", resembling the need of man for hardware assistance in complex calculations. However, the first analytical machine, as the term is realized today was first presented by the English mechanical engineer and polymath Charles Babbage in the early 19th century [Hal70]. While Babbage was considered the "father of the computer", his revolutionary machine was in comparison nowhere near the modern computers; it consisted of a vast network of cogs and wheels, while input was imported in the form of punched cards. The principle of the first modern computer was first described by computer scientist Alan Turing in his 1936 paper [Tur36]. Turing proposed the creation of a device that would manipulate symbols written on a strip of tape according to a set of laws. This hypothetical device, later named a *Turing machine* had set the basis for the algorithmic way in which modern computers operate, characterizing them as Turing-complete.

In the following years, and with the advances in electronic hardware, more complex devices were produced and mainly used for cryptology calculations in the Second World War. The mathematical basis of digital computing was developed by the British mathematician George Boole already in 1854 [Boo54], who introduced his *Boolean algebra* that is used even today in modern computers. However, the hardware architecture of modern computers was first described by John Von Neumann in 1945 [Neu45]. The *von Neumann architecture* describes the existence and function of a Central Processing Unit (CPU), which performs basic arithmetic and logical actions, along with a Memory Unit that is capable of storing data and interacting with the CPU. User interaction is performed with appropriate input/output procedures. The *von Neumann architecture* was applied to the *ENIAC* (Electronic Numerical Integrator And Computer), the first electronic programmable computer, built in the United States of America [McC99].

While the first computers with electronic components used vacuum tubes in their design, the rise to the "second generation" in computer era was given with the invention of the bipolar transistor. The transistors replaced vacuum tubes in the computer design from 1955 [Fey65], as they were smaller, required less power and emitted less heat, while their life expectancy was significantly larger than that of the vacuum tubes. From then on, computer power increased year by year. In 1962 one of the first supercomputers, named the Atlas was built in the United Kingdom by the Manchester University and the Ferranti Ltd. It was the first machine to use virtual memory and paging, with pipelined instruction execution and was capable to produce approximately 500 kFLOPS [Lav75]. In the United States a series of computers designed by Seymour Cray used innovative architecture and parallelism to achieve superior, for that time, performance. His CDC6600 is considered the first supercomputer, with performance of about 1 MFLOPS [Imp04]. Computer power has since increased dramatically.

In his paper in 1965 Gordon Moore [Moo65] stated the observation that the number of transistors in a dense integrated circuit would double once every year; in 1975 he revised his forecast by doubling the expected time to two years. While *Moore's Law* is nothing but an observation or projection of the continuing trends in computing and not a physical law, it has continued to apply

for over half a century. Today's computer systems can combine many different types of processing units and memory units to achieve the best result in computer power. In the last years "cramming" many cores in a single CPU has become common practice of the computer companies, while parallelism can be achieved even between Graphic Processing Units (GPUs), to increase that desired number in terms of floating-point operations per second (FLOPS). With the first computer systems thriving to produce several MFLOPS, comparison to today's biggest systems is impossible; the current biggest known computer system incorporates 3,120,000 processor cores, achieving the awe-inspiring amount of 54,902.4 TFLOPS [Top500].

With the advancements in the hardware arena being ceaseless, the bar is then passed to the algorithmic research, for the better utilization of all the available computer architectures. During the past years, an effort has been made to provide automation in parallelism, by introducing compiler programs that can parallelize an algorithm so it can be executed simultaneously by several processor cores. While this approach can work in several cases, best performance is achieved when the programmer himself provides the parallel algorithm [Gro99]. In 1967 Gene Amdhal published his argument about the limits of parallelism and the maximum speed-up that can be achieved with the use of multiple processors [Amd67]. His argument is now known as Amdhal's Law and is still used to predict the theoretical maximum acceleration that can be achieved by an algorithm that utilizes several processors in parallel.

In 1966 Michael J. Flynn proposed a classification system for computer architectures based on the level of parallelism that can be achieved with them [Fly66] [Fly72]. The classification that is today known as *Flynn's Taxonomy* categorizes computer architectures based on the data that each processor can be provided, and the differentiation on the instruction that can be executed by each CPU. Therefore, the initial categorization introduced four types of computer architecture:

- **SISD** (Single Information, Single Data): A sequential computer that can exploit no parallelism either in instruction or data.
- **SIMD** (Single Instruction, Multiple Data): A computer system that can exploit multiple data with a single instruction stream to perform operations that can be naturally parallelized.
- **MISD** (Multiple Information, Single Data): Multiple instruction streams operate on a single data stream.
- **MIMD** (Multiple Information, Multiple Data): Multiple instruction streams that can operate on multiple data streams. Today's distributed systems are recognized as MIMD architectures.

To expand the above classification to more modern needs, Frederica Darema proposed the term SPMD (Single Process, Multiple Data), where multiple autonomous processors can execute simultaneously the same program on different data [Dar88]. SPMD is nowadays the most common style of parallel programming.

While Flynn defined the level of parallelism at a hardware level, his taxonomy can be extended to classify parallel programming techniques. Data parallelism exists since the introduction of vector processors in the 1970s; CPUs that could implement instruction sets containing instructions that operated on one-dimensional arrays called vectors. However, since the SIMD machines have

become obsolete, data parallelism has become a programming style; e.g. application programming interfaces (API) such as the OpenMP provide the programmer with tools to hint the compiler when to apply parallelism on sequentially coded loops [Gro99].

Programs that share the MIMD architecture can operate either on a single shared, or a distributed memory space. The shared-memory model allows each processor to have access to all of a single shared address space. While shared-memory machines are hard to come by, this model can be achieved in software level when processes have their local memory address space and also share a portion of that memory.

A different approach to the data distribution and usage is the message-passing model, where a number of processes, each with their own memory address space, communicate by sending and receiving messages. The Message-Passing-Interface (MPI), developed in 1992 [MPI93] provided the means for that kind of parallelism. With the message-passing model a set of data is copied from the memory address space of a processor to that of another processor. The defining feature of the MPI is that both processors must execute the message-passing procedure; the first one sends and the latter receives.

With the advance of technology and the computer architectures, as well as the advancements in software parallelization and parallel algorithms, more complex problems could be solved in a logical amount of time. CFD was also benefited by parallel processing. Although it is difficult to pinpoint the first attempts to parallelize a CFD algorithm, examples of such use exist even in the late 1980s. In 1989 Roy D. Williams [Wil89] proposed an algorithm for the parallelization of the Euler equations for compressible fluid. In his work, he used a parallelization strategy similar to the domain decomposition with overlapping elements; the initial mesh was divided to the number of used processors and communication at the inner-boundaries (where the grid was cut) was performed by preserving copies of the inner-boundary nodes on neighboring processors. Data of the whole grid was either stored on the address space of each processor, or copies of each physical node shared data.

A more definitive approach was reported by Venkatakrishnan et al. in 1992 [Ven92], who parallelized a compressible Euler solver on unstructured grids. In their approach the domain decomposition method is fully defined, with different types of grid division, while a set of data structures for the communication between processors is proposed. Exchange of data is performed with a message-passing model. The same work has been reported in length in 1995 [Ven95], where further information on the decomposition approach was given, as well as the communication methodology between processors. In 1996 Lanteri [Lan96] compared the domain decomposition approach using overlapping and non-overlapping techniques. Despite the larger communication load, the latter proved to be more efficient; communication was performed with blocking send and receive routines.

In 2003 Tai and Zhao proposed a methodology for the parallelization of an unsteady incompressible flow solver. In their work they used the domain decomposition with overlapping elements approach along with a geometric multigrid method to accelerate the simulation procedure. Communication between processors was performed with the MPI, while the domain decomposition was performed on each level of the non-nested multigrids, with respect to the

finest grid. A similar approach was reported in 2005 by Kallinderis and Ahn [Kal05] employing the 3D incompressible Navier-Stokes with artificial compressibility, following the SPMD paradigm.

Finally, a relatively recent trend in parallel processing is the usage of the Graphics Processing Units (GPUs). A GPU is a processor specialized in graphics rendering. The demand for faster, higher quality graphics in computer games has led to the development of better GPUs with more than 1000 cores per processor unit. Use of the GPU with non-graphical problems was made possible by translating them to graphical programming tasks. However, programming with a GPU is different than conventional programming with a CPU. Specific programming languages have been developed for this task, such as CUDA and BrookGPU, while a standardized effort was made with OpenCL (Open Computing Language). Several researchers have tried to solve CFD problems on GPUs in the past years, such as Göddeke et al. [God09], Thibault and Senocak [Thi09] and Jespersen [Jes10]. While GPUs seem a better alternative than the conventional CPUs, special consideration has to be made for the transfer of data to the GPU memory, which is usually smaller than virtual memory, from the CPU memory and vice-versa. Bad decisions in that area may produce deterioration of an algorithm's speedup.

1.4 Domain Decomposition

As it was hinted in paragraph 1.3, the most common parallelization procedure in CFD is the domain decomposition with, or without overlapping elements. The main concept of the method is to divide the initial computational mesh into smaller ones, using a decomposition algorithm, and solve any flow equations on each sub-domain separately, using data that can be imported through the newly generated inner boundaries from neighboring sub-domains. The first domain decomposition method, known as *alternating Schwarz method*, was formulated by H. A. Schwarz in 1870 [Sch70], although it initially served as a theoretical tool; its use for solving elliptic boundary value problems was proved more than half a century later [Smi04]. The use of domain decomposition has been applied to a wide variety of fields, such as spectral methods, adaptive methods, mixed finite element methods, boundary element methods and others.

While the part of the solution procedure involves the specific PDEs to be evaluated, the initial part of the domain decomposition method concerns the division of the computational domain into smaller ones. As the computational domain in most cases takes the form of a structured or unstructured grid, the initial division of that grid proves to be essential, mainly for reasons involving code synchronicity and acceleration. A grid is essentially a number of nodes connected by edges, while computation of the PDEs is performed either on the grid nodes, or the grid elements. Therefore, code synchronicity would be achieved with a division of the initial mesh into smaller ones that contain (as much as possible) the same amount of nodes, or elements. The second criterion, which is the obtained acceleration, involves also the amount of data that would be transferred between neighboring sub-domains, which means minimization of the connecting edges, or faces between neighboring partitions. For those reasons several algorithms have been developed.

In 1992 Venkatakrishnan et al. [Ven92] proposed three partitioning algorithms that they used in their parallelization method. All methods were used for the partitioning of triangular grids and are based on the centroidal-dual grid that is composed by lines joining triangle centroids. The first method, called *coordinate bisection*, uses the coordinate information of the centroidal-dual mesh; coordinates are sorted in a particular direction and then divided into two sets. The second method, called *graph bisection*, involves an iterative method of defining level sets, which represents the neighbor lists starting with a root; when half of the mesh has been defined at one set a partition is complete. The third method, called *spectral bisection*, is based on the spectral partitioning algorithm. Partition division is then repeated as many times needed to produce the required number of divisions. They reported also that the *spectral bisection* method provides better partitions, in terms of inner boundary lengths.

Farhat and Lesoinne [Far93] reported a family of cost-effective algorithms for the automatic partitioning of arbitrary 2D and 3D meshes. In particular, they examined the effect of different partitioning methodologies on different solution techniques (explicit or implicit schemes). Moreover, they assessed the effect of particular multiprocessor architectures (MIMD, SIMD) on the design of a mesh partitioning algorithm, as well as the impact of the partitioning strategy on load balancing, operation count, operator conditioning and processor mapping.

Diniz et al. [Din95] proposed three algorithms for the partitioning of unstructured grids in parallel, capable of generating load balanced partitions, where communication between sub-domains is minimized and computational load is balanced. The algorithms combine two commonly used serial partitioners; the *Inertial* method [Nou87] and a parallel variant of the local greedy heuristic method by Fiduccia and Mattheyses [Fid82].

Multi-level algorithms prove to provide very good results and have moderate complexity [Bui93] [Hen95] [Kar95]; the mesh, considered originally as a graph, is coarsened until it contains only a few hundred nodes. At this point bisection of the coarsened graph is performed and then the partition is projected to the original graph by iterative refining. The algorithm assumes that since the finer mesh has more degrees of freedom, the edge-cut produced by the refinement is minimal.

A specific case of a multilevel partitioning algorithm is the METIS algorithm by Karypis [Kar98]. METIS is a software package for partitioning large irregular graphs, large meshes and reducing orderings of sparse matrices. The improvement to the multi-level partitioning algorithms that is introduced with this software is a set of algorithms for the better coarsening procedure, as well as the refinement steps. The resulting partitions are well load balanced with the minimum edge-cut between neighboring partitions. Various researchers have used this software for the acceleration via parallel processing of their methodologies [Kal05] [Tai03].

1.5 Present study

In this work a code named *Galatea-I* was developed, for the simulation of steady and unsteady incompressible inviscid, viscous laminar and turbulent flows, with the use of the incompressible Navier-Stokes equations, augmented by the artificial compressibility method. For the simulation of turbulence the appropriate Reynolds Averaged Navier-Stokes equations [RANS] are evaluated

along with the Shear Stress Transport model [Men03]. Spatial discretization is performed with a node-centered finite-volume scheme, on three-dimensional hybrid grids that are consisted of tetrahedral, prismatic and pyramidal elements [Bla01]. Calculation of the inviscid fluxes is performed with Roe's approximate Riemann solver [Roe81], while for the viscous fluxes two schemes have been incorporated, to the user's discretion; a method employing the edge-dual control volume paradigm [Kal05] and a simpler but significantly faster nodal-averaged one [Bla01]. Boundary conditions are imposed typically for solid walls and surfaces with free-slip conditions; however for the inlet and outlet regions a characteristics based scheme was developed, based on a similar two-dimensional scheme reported by Anderson et al. [And96]. Temporal discretization of the governing equations is performed with an explicit four-stage Runge-Kutta (RK(4)) scheme [Lal88], while unsteady solutions can be simulated with a dual time-stepping scheme [Kal05].

Acceleration of the simulation procedure is performed in two ways. Firstly, a domain decomposition with overlapping elements method has been incorporated to the solver. Division of the initial hybrid mesh is performed with the METIS [Kar95] software, while an efficient algorithm has been developed for the formulation of the overlapping elements layer. Communication between adjacent meshes is performed with the MPI, using mainly collective operation routines that distribute the required quantities to all eligible recipients. Appropriate data structures and algorithms have been developed for accurate exchange of data between adjacent partitions. Secondly, an agglomeration multigrid scheme has been incorporated, in which several successively coarser grids are generated with the fusion of adjacent control volumes, by following a set of rules. Fusion can be performed in isotropic, semi- or full-coarsening directional mode, depending of the type of flow to be simulated (inviscid, viscous laminar, or turbulent) and consequently on the type of grid (tetrahedral or hybrid) [Mav99] [Car00] [Nish13] [Lyg14a] [Lyg14b] [Lyg14c]. For the implementation of the parallel procedure on all multigrid levels special care has to be taken for the *ghost* nodes at the overlapping region, as they have to be merged or not, according to the behavior of their corresponding *core* nodes [Lyg14b] [Lyg14c]. A FAS [Bla01] or a combined FMG-FAS [Lam04] [Lyg14b] [Lyg14c] procedure can be followed for the iterative solution with the multigrid method.

The proposed methodology has been evaluated against many different three-dimensional and quasi-3D inviscid, viscous laminar and turbulent benchmark test cases, while the obtained numerical results were successfully compared to reference experimental and numerical results. The agreement between the obtained results with the proposed code and the reference ones indicates the capability of the aforementioned solver to predict accurately complex incompressible flow phenomena in an efficient manner, with the help of the proposed parallel processing methodology and the multigrid scheme.

The structure of this dissertation is as follows: in Chapter 2 the governing equations for incompressible flow, along with the formulation of the artificial compressibility method is presented, as well as the RANS equations with the SST turbulence model. In Chapter 3 the numerical methodology is described, namely the spatial discretization scheme, flux computation, with specific mention to the inflow and outflow boundary conditions, and the temporal discretization scheme, with different formulation in case of a time-marching solution. In Chapter

4 the parallelization strategy is presented with the mention of the algorithms used for the generation of the overlapping layer, as well as the exchange of data. Finally, in Chapter 5 several test cases are presented for the evaluation of the proposed methodology.

CHAPTER 2

MATHEMATICAL MODELING

In this chapter the mathematical modeling for the incompressible flow solver is presented. The governing equations for the flow model are the Navier-Stokes equations for incompressible fluid, coupled with the artificial compressibility method [Cho67a] [Rog91] [Kal05]. Turbulence is calculated via the Menter's Shear Stress Transport (SST) model [Men03].

2.1 Governing Equations

2.1.1Flow Model

Incompressible flow is depicted by the continuity equation, depicting conservation of mass and the momentum equations:

$$\begin{aligned}
&\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0\\
&\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=-\frac{\partial p}{\partial x}+\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \tau_{xz}}{\partial z}\\
&\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}=-\frac{\partial p}{\partial x}+\frac{\partial \tau_{yx}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y}+\frac{\partial \tau_{yz}}{\partial z}\\
&\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}=-\frac{\partial p}{\partial x}+\frac{\partial \tau_{zx}}{\partial x}+\frac{\partial \tau_{zy}}{\partial y}+\frac{\partial \tau_{zz}}{\partial z}
\end{aligned} \tag{2.1}$$

As the numerical solution of the above equations presents difficulties, especially due to the special role of pressure [Cho67a] and the challenge to maintain incompressibility in an iterative process [Kir01] [Kwa11] and thus satisfy the continuity equation, the artificial compressibility method has been applied to the original equations. With this method a pseudo-time derivative of pressure is added to the continuity equation, rendering the equations applicable to an iterative solution procedure.

The augmented with the artificial compressibility method Navier-Stokes equations are presented in vector form with dimensionless parameters as:

$$\frac{\partial \vec{Q}}{\partial \tau}+\nabla\left(\vec{f_i}-\vec{f_v}\right)=\vec{S} \tag{2.2}$$

where $\vec{S}$ denotes the source term vector, that is equal to zero in this work [Lyg14a] [Sar14], while τ represents the pseudo-time over which the variables are integrated so that the equations will converge in a steady state solution, while $\vec{Q}$, $\overrightarrow{f_i}$ and $\overrightarrow{f_v}$ denote the flow variables, inviscid fluxes and viscous fluxes vectors, respectively

$$\vec{Q} = [p\ u\ v\ w]^T \tag{2.3}$$

$$\overrightarrow{f_i} = \begin{pmatrix} \beta u \\ u \cdot u + p \\ u \cdot v \\ u \cdot w \end{pmatrix} \hat{\imath} + \begin{pmatrix} \beta v \\ v \cdot u \\ v \cdot v + p \\ v \cdot w \end{pmatrix} \hat{\jmath} + \begin{pmatrix} \beta w \\ w \cdot u \\ w \cdot v \\ w \cdot w + p \end{pmatrix} \hat{k} \tag{2.4}$$

$$\overrightarrow{f_v} = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \end{pmatrix} \hat{\imath} + \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \end{pmatrix} \hat{\jmath} + \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \end{pmatrix} \hat{k} \tag{2.5}$$

where p represents pressure, u, v, w are the Cartesian components of velocity and β the artificial compressibility parameter. The shear stresses τ_{ij} are formulated as:

$$\begin{aligned} \tau_{xx} &= 2\frac{\nu}{Re}\frac{\partial u}{\partial x} - \overline{u'u'} \\ \tau_{yy} &= 2\frac{\nu}{Re}\frac{\partial v}{\partial y} - \overline{v'v'} \\ \tau_{zz} &= 2\frac{\nu}{Re}\frac{\partial w}{\partial z} - \overline{w'w'} \\ \tau_{xy} = \tau_{yx} &= \frac{\nu}{Re}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) - \overline{u'v'} \\ \tau_{yz} = \tau_{zy} &= \frac{\nu}{Re}\left(\frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) - \overline{v'w'} \\ \tau_{xz} = \tau_{zx} &= \frac{\nu}{Re}\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right) - \overline{u'w'} \end{aligned} \tag{2.6}$$

Here *Re* denotes the Reynolds number and ν represents the kinematic viscosity of the fluid, while the averaged quantities $\overline{u_i u_j}$ are the Reynolds stress tensors [Bla01], formulating essentially the Reynolds Averaged form of the Navier-Stokes equations (RANS) [Bla01].

The normalization of the variables used in the above equations is performed using a reference length L^*, the free-stream pressure $p_\infty = \rho V^2_{ref}$, where ρ is the fluid density, the far-field velocity V_∞ and the far-field kinetic viscosity ν_{ref}, as:

$$x_i = \frac{x_i^*}{L^*},\ \ u_i = \frac{u_i^*}{V_\infty},\ \ \nu = \frac{\nu^*}{\nu_{ref}},\ \ \ p = \frac{p^*}{p_\infty},\ \ \ t = \frac{t^*}{(L^*/V_\infty)}\ and\ \ Re = \frac{(V_\infty \cdot L^*)}{\nu_{ref}} \tag{2.7}$$

2.1.2 Turbulence Modeling

In order to finalize the set of equations the Reynolds stress tensors should be defined. In this work the Reynolds averaging scheme along with the Boussinesq assumption is adopted [Bla01], so the formulation of the shear stresses is defined as [Lyg14a] [Sar14]

$$\tau_{ij}^{t} = -\overline{u_i u_j} = \nu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{2.8}$$

where k is the turbulence kinetic energy per unit mass and ν_t is the turbulent kinematic viscosity, therefore Equation 2.6 is written as:

$$\tau_{ij} = \tau_{ij}^{l} + \tau_{ij}^{t} = (\nu + \nu_t) \cdot \frac{1}{Re} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{2.9}$$

For the evaluation of the turbulent kinetic energy, the SST (Shear Stress Transport) two-equation turbulence model [Men03] was incorporated in the incompressible flow solver. The SST model is a combination of k-ε and k-ω turbulence models, with the transition between the two realized with the appropriate blending functions. The PDE formulation is similar to that of the flow model, with the exception of the source term being not equal to zero:

$$\frac{\partial \vec{Q}_{SST}}{\partial \tau} + \nabla \left(\vec{f}_{i_{SST}} - \vec{f}_{v_{SST}} \right) = \vec{S}_{SST} \tag{2.10}$$

The variables vector $\vec{Q}_{SST} = [k, \omega]^T$ of the turbulence model contains the turbulent kinetic energy k and the specific dissipation rate ω, while the inviscid and viscous terms, as well as the source term are defined as:

$$\vec{f}_{i_{SST}} = \begin{pmatrix} uk \\ u\omega \end{pmatrix} \hat{\imath} + \begin{pmatrix} vk \\ v\omega \end{pmatrix} \hat{\jmath} + \begin{pmatrix} wk \\ w\omega \end{pmatrix} \hat{k} \tag{2.11}$$

$$\overrightarrow{f_{v_{SST}}} = \begin{bmatrix} \left(\frac{\nu}{Re} + \nu_t \sigma_k \right) \cdot \frac{\partial k}{\partial x} \\ \left(\frac{\nu}{Re} + \nu_t \sigma_\omega \right) \cdot \frac{\partial \omega}{\partial x} \end{bmatrix} \hat{\imath} + \begin{bmatrix} \left(\frac{\nu}{Re} + \nu_t \sigma_k \right) \cdot \frac{\partial k}{\partial y} \\ \left(\frac{\nu}{Re} + \nu_t \sigma_\omega \right) \cdot \frac{\partial \omega}{\partial y} \end{bmatrix} \hat{\jmath} + \begin{bmatrix} \left(\frac{\nu}{Re} + \nu_t \sigma_k \right) \cdot \frac{\partial k}{\partial z} \\ \left(\frac{\nu}{Re} + \nu_t \sigma_\omega \right) \cdot \frac{\partial \omega}{\partial z} \end{bmatrix} \hat{k} \tag{2.12}$$

$$\vec{S}_{SST} = \vec{S}_{k\omega} + \vec{D}_{k\omega} = \begin{pmatrix} P_k - \beta^* \omega k \\ \frac{\gamma^* \rho}{\mu_t} P_k - \bar{\beta} \omega^2 \end{pmatrix} + \begin{pmatrix} 0 \\ 2(1 - F_1) \frac{\sigma_{\omega_2}}{\omega} \nabla k \nabla \omega \end{pmatrix} \tag{2.13}$$

The parameters σ_k, σ_ω, $\bar{\beta}$, γ^* are calculated with regard to the equivalent constants of the k-ω and k-ε models, using the blending function F_1 in a way that the parameter from the k-ω model is multiplied by F_1 and the one from *k-ε* with *(1-*F_1*)* and then added together. Therefore, if φ represents all coefficients of the SST model, φ_1 the constants from k-ω and φ_2 the ones from k-ε, then the first would be calculated as:

$$\varphi = F_1 \varphi_1 + (1 - F_1) \varphi_2 \tag{2.14}$$

The utilized constants of the k-ω and k-ε models have values σ_{k1}=0.85, σ_{k2}=1, $\sigma_{\omega 1}$=0.5, $\sigma_{\omega 2}$=0.856, β_1=0.075, β_2=0.0828, γ_1^*=0.555 and γ_2^*=0.44, while β^* has a constant value of 0.09 [Men03] [NASA]. The blending function F_1 is equal to zero far away from the solid wall surface, where the k-ε model is activated and switches to unity near the boundary, where the k-ω model takes over. The turbulent energy production term is calculated as:

$$P_k = \sum_{i=1}^{3}\sum_{j=1}^{3}\left(\tau_{ij}^{t}\frac{\partial u_i}{\partial x_j}\right) =$$

$$\nu_t \sum_{i=1}^{3}\sum_{j=1}^{3}\left[\left(\frac{\partial u_i}{\partial x_j}\right)^2 + \frac{\partial u_i}{\partial x_j}\frac{\partial u_j}{\partial x_i}\right] = \tag{2.15}$$

$$\nu_t\left[2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + 2\left(\frac{\partial w}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial y}+\frac{\partial y}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)^2\right]$$

The blending function F_1 is evaluated as:

$$\begin{aligned} F_1 &= tanh(arg_1^4\ ) \\ arg_1 &= min\left[max\left(\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega Re}\right), \frac{4\rho\sigma_{\omega_2}k}{CD_{k\omega}d^2}\right] \\ CD_{k\omega} &= max\left(2\sigma_{\omega_2}\frac{1}{\omega}\nabla k\nabla\omega, 10^{-10}\right) \end{aligned} \tag{2.16}$$

where d is the distance of the point in space where the SST model is applied, from the nearest solid wall surface.

The turbulent kinematic viscosity is also calculated with the help of a blending function as [Men03] [NASA]

$$\nu_t = \frac{\alpha_1 k}{\max(a_1\omega, SF_2)} \tag{2.17}$$

$$S = \sqrt{\sum_{i=1}^{3}\sum_{j=1}^{3}(S_{ij}S_{ij})} = \sqrt{2\sum_{i=1}^{3}\sum_{j=1}^{3}\left(\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right)} = \tag{2.18}$$

$$\sqrt{2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + 2\left(\frac{\partial w}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial y}+\frac{\partial y}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)^2} =$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) \tag{2.19}$$

$$F_2 = \tanh(\arg_2^2\ ) \tag{2.20}$$

$$\arg_2^2 = \max\left(\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega Re}\right) \tag{2.21}$$

The SST model selection over other alternative two-equation models, such as the *k*-*ε* and *k*-*ω*, was made considering its superiority, as it combines the advantages of the two aforementioned models, while diminishing their drawbacks [Men03]. In particular, while the *k*-*ε* model is very accurate at predicting turbulence outside the boundary layer, it is unstable near the wall [Rod86]. In contrast, the *k*-*ω* model has great advantage near the wall surfaces, while its solution proves to be sensitive to the free stream values of *ω* outside the boundary layer [Men92]. The blending function utilized by the SST model designates essentially that near the wall boundaries the k-ω model will be more prevalent, while away from the wall boundaries the k-ε model will be sovereign.

2.2 Time-Accurate Formulation

The original form of the governing equations, described by Chorin [Cho67] was in a steady-state formulation, where time was advanced through the pseudo-time parameter τ. For the simulation of time-accurate flows the governing equations have to be modified to include a real time parameter t. In this work, the dual-time stepping scheme, developed by Belov [Bel95] is adopted. In this method, a true-time derivative of the velocity components is added to the original equation (2.2) as:

$$K\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{Q}}{\partial \tau} + \nabla\left(\vec{f_i} - \vec{f_v}\right) = \vec{S} \tag{2.22}$$

$$\text{K=diag(0,1,1,1)} \tag{2.23}$$

The same dual-time stepping scheme has to be applied to the turbulence model equations, in order to couple them with the flow ones:

$$\frac{\partial \vec{Q}_{SST}}{\partial t} + \frac{\partial \vec{Q}_{SST}}{\partial \tau} + \nabla\left(\vec{f}_{i_{SST}} - \vec{f}_{v_{SST}}\right) = \vec{S}_{SST} \tag{2.10}$$

CHAPTER 3

NUMERICAL MODELING

In this chapter the numerical scheme for the evaluation of the governing equations described in the previous chapter is presented. A node-centered finite volume scheme was adopted for the spatial discretization of the flow and turbulence model equations, while temporal discretization is performed with an explicit four-stage Runge-Kutta scheme. The evaluation of inviscid fluxes is carried out with Roe's approximate Riemann solver, while two methods were adopted for the calculation of the viscous ones; an element based approach, based on the edge-dual volume scheme, and a nodal-averaging one.

3.1 Spatial discretization

The computational meshes over which the flow and turbulence equations are discretized, are hybrid unstructured grids, consisting mainly of triangular elements in two dimensions, or tetrahedral elements in three dimensions. Layers of quadrilateral (in 2D) or prismatic (in 3D) elements are added to the areas of the geometry where no-slip conditions must be employed, so that the viscous boundary layer can be adequately computed. A three-dimensional grid may contain pyramidal elements to connect the prismatic elements with the tetrahedral ones. Although a structured grid would provide more accurate and efficient evaluation of fluxes and gradients [Bla01], with the minimum requirements in computer memory, the excessive time required for the generation of a structured grid on a complex geometry was the decisive factor for the use of unstructured ones in this study. A number of simulations on different test cases, on simple and more complex geometries, were rendered possible with the use of unstructured grids, without the tedious work of structured grid generation.

Discretization over the computational grids is performed with a node-centered finite volume scheme. With this method, the computational geometry is divided into small arbitrary control volumes. Inside each control volume the quantities that are associated to the flow, such as flow variables and gradients, are considered constant, while the surface integrals of Equation (2.2) is evaluated by the summation of the fluxes passing through the control volume boundaries [Bla01]. With the node-centered scheme, a control volume, also called median-dual volume in that case [Mav96] [Koo00] [Kim03] [Lyg12] [Sar14], is constructed around a grid node, by connecting with lines the edge midpoints, face barycenters and barycenters of grid elements sharing this node (only edge midpoints and element centers of mass in case of two-dimensional grids [Kal96] [Kal05]). Hence, each element's volume is shared equally to the number of its nodes. The 2D equivalent of the median-dual volume of a node, contributed by three triangular elements and two quadrilateral ones, is presented in Figure 3.1.

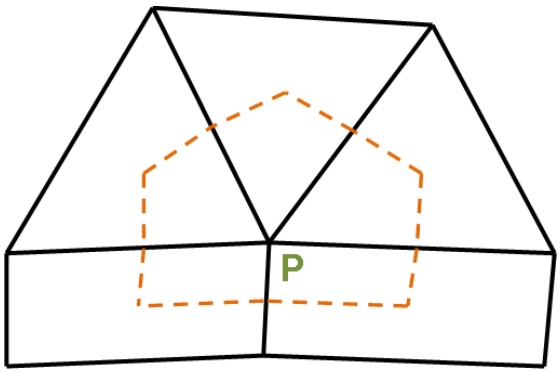


Figure 3.1 – 2D Median-dual volume of a node P, contributed by three triangular and two quadrilateral elements.

The control volume is bounded by dashed lines, which also constitute the control surface (boundary), over which inviscid and viscous fluxes will be eventually calculated. The contribution of a tetrahedral, a prismatic and a pyramidal element to the 3D median-dual volume of a node P is presented in Figure 3.2. For the sake of better presentation, the rear faces of the grid elements were painted in blue, while the surface of the control volume was painted in purple. Obviously, the control volume of the node P is not complete, as several other elements would also share this node and contribute to its control volume.

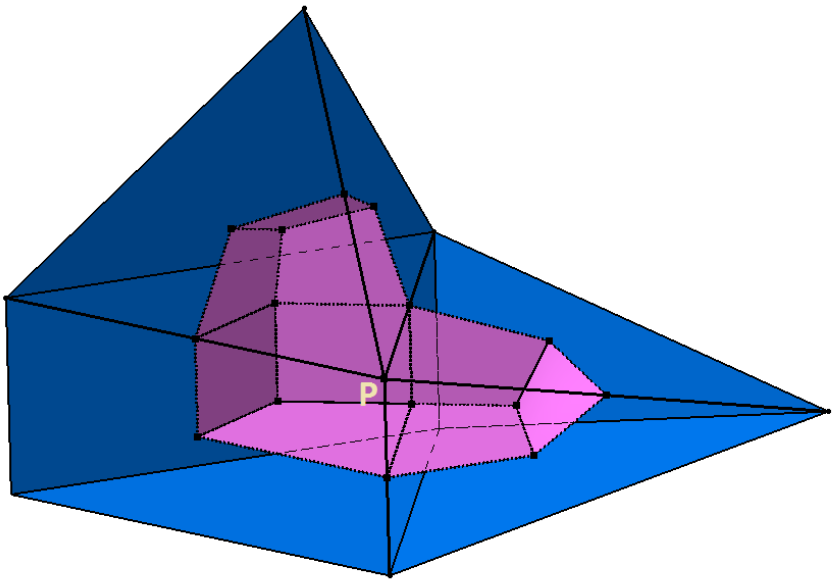


Figure 3.2 – 3D Median-dual volume of node P, contributed by a tetrahedral, a prismatic and a pyramidal element.

For the calculation of the grid elements' volumes, the tetrahedral elements are primarily evaluated as [Weiss]

$$\Omega = \frac{1}{3!}\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \tag{3.1}$$

where (x_i, y_i, z_i) represent the Cartesian coordinates of the tetrahedron's vertices, and i = 1, ..,4. The volumes of prisms and pyramids are evaluated with the use of the tetrahedron volume formula, after their appropriate division into tetrahedrons, as presented in Figure 3.3.

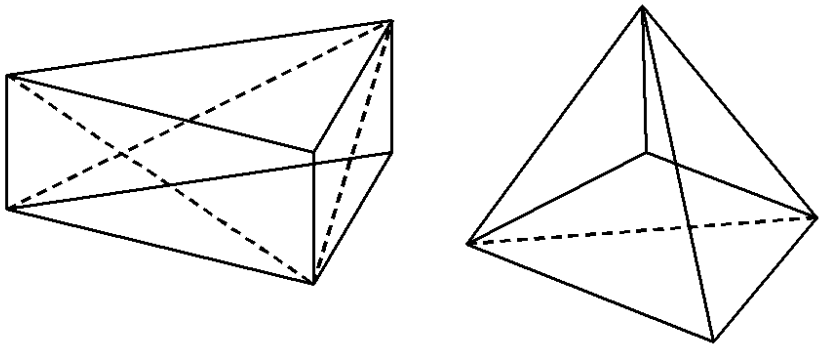


Figure 3.3 – 3D Division of prismatic and pyramidal elements into tetrahedrons for the calculation of their volumes.

With the definition of the discretization method, Equation 2.2 is integrated over the control volume Ω of a node P as

$$\frac{\partial}{\partial \tau}\iiint_{\Omega} \vec{Q}\,dV + \iiint_{\Omega} (\overrightarrow{f_i} - \overrightarrow{f_v})dV = \iiint_{\Omega} \vec{S}\,dV \tag{3.2}$$

which is then treated with the Green-Gauss divergence theorem [Bla01] into

$$\frac{\partial}{\partial \tau}\iiint_{\Omega} \vec{Q}\,dV + \iint_{\partial\Omega} (\overrightarrow{f_i} - \overrightarrow{f_v}) \cdot \hat{n}\,ds = \iiint_{\Omega} \vec{S}\,dV \tag{3.3}$$

where $\partial\Omega$ indicates the boundaries of the control volume Ω, defined by the sum of the faces that surround each edge PQ, with Q representing all the neighboring nodes of P. The unit vector $\hat{n}$ represents the outward pointing normal vector of the boundary $\partial\Omega$, thus transforming the inviscid and viscous flux vectors $\overrightarrow{f_i}$, $\overrightarrow{f_v}$ as [Shi01]

$$\overrightarrow{f_i} \cdot \hat{n} = \begin{bmatrix} \beta(u \cdot n_x + v \cdot n_y + w \cdot n_z) \\ u(u \cdot n_x + v \cdot n_y + w \cdot n_z) + n_x p \\ v(u \cdot n_x + v \cdot n_y + w \cdot n_z) + n_y p \\ w(u \cdot n_x + v \cdot n_y + w \cdot n_z) + n_z p \end{bmatrix} = \begin{bmatrix} \beta\Theta \\ u\Theta + n_x p \\ v\Theta + n_y p \\ w\Theta + n_z p \end{bmatrix} \tag{3.4}$$

$$\overrightarrow{f_v} \cdot \hat{n} = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_x\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_x\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_x\tau_{zz} \end{bmatrix} \tag{3.5}$$

where the normal to the control surface velocity Θ is illustrated as:

$$\Theta = un_x + vn_y + wn_z \tag{3.6}$$

The unit vector $\hat{n}$ is defined as

$$\hat{n} = \frac{\vec{n}}{|\vec{n}|} = (n_x, n_y, n_z) \tag{3.7}$$

In order to minimize memory usage, an edge-based data structure was employed in the programming of the above quantities [Bla01] [Kal05]. While most of the appropriate data, such as flow variables, derivatives and gradients, are stored on the nodes of the grid, calculation of fluxes is performed in an edge-wise manner to reduce the computational complexity. Therefore, the total computational flux amounted to a single node *P* is computed as the sum of fluxes passing the interface between the control volume of *P* and every neighboring node *Q*. The required normal to the control surface vector $\vec{n}_{PQ}$ would then be calculated as the sum of all the normal vectors to the facets that constitute the interface between nodes *P* and *Q* [Bla01], as it is presented in Figure 3.4.

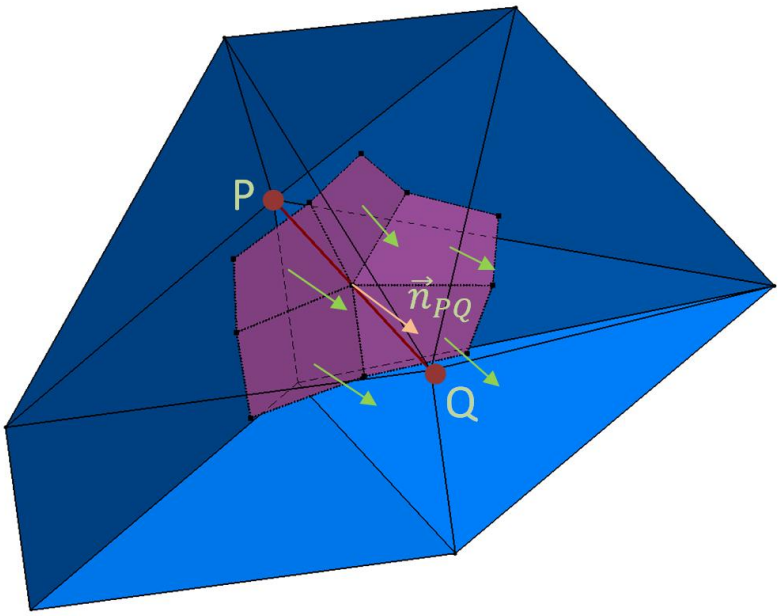


Figure 3.4 – The normal to the control surface vector $\vec{n}_{PQ}$ is calculated as the sum of the normal vectors to the facets that constitute the interface between nodes *P* and *Q*.

With the definition of the necessary components, Equation 3.3 is then formulated as:

$$\left(\frac{\partial \vec{Q}}{\partial \tau}\right)_P \Omega_P + \iint_{\partial\Omega} \vec{f_i} \cdot \hat{n}\, ds - \iint_{\partial\Omega} \vec{f_v} \cdot \hat{n}\, ds = \iiint_{\Omega} \vec{S}\, dV \tag{3.8}$$

The surface integrals for the inviscid and viscous terms of equation 3.8 are evaluated as sums of all the fluxes through the faces composing the control volume of node *P*, on the direction $\vec{n}_{PQ}$, where *Q* is each node connected to *P* with an edge *PQ*

$$\iint_{\partial\Omega} \vec{f_i} \cdot \hat{n}\, ds = \sum_{Q \epsilon K_N(P)} \vec{f_i}(\vec{Q}_L, \vec{Q}_R; \hat{n}) \tag{3.9}$$

$$\iint_{\partial\Omega} \vec{f_v} \cdot \hat{n}\, ds = \sum_{Q \epsilon K_N(P)} \vec{f_v}(\vec{Q}_L, \vec{Q}_R; \hat{n}) \tag{3.10}$$

where $\vec{Q}_L$ and $\vec{Q}_R$ are the values of the flow variables on the left and right side of an edge *PQ*, respectively.

3.2 Calculation of fluxes

3.2.1 Inviscid fluxes

For the calculation of the inviscid fluxes of the flow equations, a one-dimensional Riemann problem is considered along the direction of the normal vector of each face of the control volume of a node *P*. As the exact solution of such a problem requires an extensive amount of calculations, an upwind scheme with Roe's approximate Riemann solver [Roe81] is employed in this work. With this premise, the inviscid fluxes at the midpoint of an edge *PQ* can be evaluated as:

$$\vec{f_i}(\vec{Q}_L, \vec{Q}_R; \hat{n}) = \frac{1}{2}\left[\left(\vec{f_i}(\vec{Q}_L; \hat{n}) + \vec{f_i}(\vec{Q}_R; \hat{n})\right] - \frac{1}{2}\left|\tilde{A}(\vec{Q}_L, \vec{Q}_R; \hat{n})\right|(\vec{Q}_R - \vec{Q}_L)\right. \tag{3.11}$$

$$|\tilde{A}| = T\left|\tilde{\Lambda}\right|T^{-1} \tag{3.12}$$

$$\Lambda = diag(\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n) \tag{3.13}$$

Here $\tilde{A}$ represents the Jacobian matrix of the inviscid flux vector $\vec{f_i}$, approximated with Roe's hypothesis, while the quantities marked by the *L* and *R* subscripts denote values of the variables taken at the left and right side of the boundary between nodes *P* and *Q*, respectively. The dependent variables used for the construction of $\tilde{A}$ are simple algebraic averages of the *L* and *R* variable vectors, which Taylor and Witfield [Tay91] proved that they satisfy Roe's conditions for the construction of the flux Jacobian of the Euler equation with artificial compressibility. Hence, the flux Jacobian [Shi01]

$$\frac{\partial F}{\partial Q}=\begin{bmatrix} 0 & \beta n_x & \beta n_y & \beta n_z \\ n_x & \Theta+un_x & un_y & un_z \\ n_y & vn_x & \Theta+vn_y & vn_z \\ n_z & wn_x & wn_y & \Theta+wn_z \end{bmatrix} \tag{3.14}$$

satisfies:

$$\tilde{A}=\left.\frac{\partial F}{\partial Q}\right|_{Q=\bar{Q}} \tag{3.15}$$

$$\bar{Q}=\frac{1}{2}(Q_L+Q_R) \tag{3.16}$$

A non-singular eigensystem in three dimensions, proving that $\tilde{A}$ is diagonizable was reported by Shin [Shi01] and it is presented as follows:

$$\Lambda=diag(\Theta,\Theta,\Theta+c,\Theta-c) \tag{3.17}$$

$$c=\sqrt{\Theta^2+\beta} \tag{3.18}$$

$$T=\begin{bmatrix} 0 & 0 & c & -c \\ x_1 & x_2 & n_x+\dfrac{u(\Theta+c)}{\beta} & n_x+\dfrac{u(\Theta-c)}{\beta} \\ y_1 & y_2 & n_y+\dfrac{v(\Theta+c)}{\beta} & n_y+\dfrac{v(\Theta-c)}{\beta} \\ z_1 & z_2 & n_z+\dfrac{w(\Theta+c)}{\beta} & n_Z+\dfrac{w(\Theta-c)}{\beta} \end{bmatrix} \tag{3.19}$$

$$\mathrm{T}^{-1}=\begin{bmatrix} \dfrac{1}{c^2}[n_x(wy_2-vz_2) & \dfrac{1}{c^2}[\beta(n_zy_2-n_yz_2) & \dfrac{1}{c^2}[\beta(n_xz_2-n_zx_2) & \dfrac{1}{c^2}[\beta(n_yx_2-n_xy_2) \\ +n_y(uz_2-wx_2) & +\Theta(wy_2-vz_2)] & +\Theta(uz_2-wx_2)] & +\Theta(vx_2-uy_2)] \\ +n_z(vx_2-uy_2)] & & & \\ \dfrac{1}{c^2}[n_x(wy_1-vz_1) & \dfrac{1}{c^2}[\beta(n_yz_1-n_zy_1) & \dfrac{1}{c^2}[\beta(n_zx_1-n_xz_1) & \dfrac{1}{c^2}[\beta(n_xy_1-n_yx_1) \\ +n_y(uz_1-wx_1) & +\Theta(vz_1-wy_1)] & +\Theta(wx_1-uz_1)] & +\Theta(uy_1-vx_1)] \\ +n_z(vx_1-uy_1)] & & & \\ \dfrac{1}{2c^2}(c-\Theta) & \dfrac{1}{2c^2}\beta n_x & \dfrac{1}{2c^2}\beta n_y & \dfrac{1}{2c^2}\beta n_z \\ -\dfrac{1}{2c^2}(c+\Theta) & \dfrac{1}{2c^2}\beta n_x & \dfrac{1}{2c^2}\beta n_y & \dfrac{1}{2c^2}\beta n_z \end{bmatrix} \tag{3.20}$$

The quantity denoted by *c* represents the artificial speed of sound at a specified point in space and depends on the artificial compressibility parameter. The vectors $\hat{a}=(x_1,y_1,z_1)$ and $\hat{b}=(x_2,y_2,z_2)$ are unit vectors, with their cross product producing $\hat{n}$:

$$\hat{a}\times\hat{b}=\hat{n} \tag{3.21}$$

For the calculation of vectors $\hat{a}$ and $\hat{b}$ the component of $\hat{a}$ which is in rotational order after the one with the maximum value is set to zero. Then the other two components are evaluated by a set of equations; the length of the vector must be equal to unity and the inner product between $\hat{a}$ and $\hat{n}$ must be equal to zero. Afterwards it is easy to calculate $\hat{b}$ as the cross product between $\hat{n}$ and $\hat{a}$. It is obvious then that there are three cases that must be considered for calculating $\hat{a}$ initially [Vra12]:

if $\max(x_1, y_1, z_1) = x_1$ then

$$y_1 = 0$$
$$\begin{cases} |\hat{a}| = 1 \\ \hat{a} \cdot \hat{n} = 0 \end{cases} \Rightarrow \begin{cases} \sqrt{x_1^2 + z_1^2} = 1 \\ x_1 n_x + z_1 n_z = 0 \end{cases}$$

if $\max(x_1, y_1, z_1) = y_1$ then

$$z_1 = 0$$
$$\begin{cases} |\hat{a}| = 1 \\ \hat{a} \cdot \hat{n} = 0 \end{cases} \Rightarrow \begin{cases} \sqrt{x_1^2 + y_1^2} = 1 \\ x_1 n_x + y_1 n_y = 0 \end{cases}$$

if $\max(x_1, y_1, z_1) = z_1$ then

$$x_1 = 0$$
$$\begin{cases} |\hat{a}| = 1 \\ \hat{a} \cdot \hat{n} = 0 \end{cases} \Rightarrow \begin{cases} \sqrt{y_1^2 + z_1^2} = 1 \\ y_1 n_y + z_1 n_z = 0 \end{cases}$$

Finally, using the secant plane approximation [Roe81] [Lan98] to the inviscid flux vector the following Equation (3.22) is used to transform Equation (3.11)

$$\vec{f}_i(\vec{Q}_R; \hat{n}) = |\tilde{A}|(\vec{Q}_R - \vec{Q}_L) + \vec{f}_i(\vec{Q}_L; \hat{n}) \tag{3.22}$$

thus obtaining the following formula that is actually implemented in this work:

$$\vec{f}_i(\vec{Q}_L, \vec{Q}_R; \hat{n}) = \vec{f}_i(\vec{Q}_L; \hat{n}) + |\tilde{A}|(\vec{Q}_R - \vec{Q}_L) \tag{3.23}$$

Algorithmically, a single edge-loop is required for the calculation of inviscid fluxes for all nodes of the computational grid. The nodes *P* and *Q* of an edge *PQ* of the grid are arbitrarily assigned the *L*, or *R* notification, so that the flux that passes through the boundary between the two nodes are added to the flux balance of *L* and subtracted from that of *R*. In this way the positive direction of $\hat{n}$ is preserved on all edges. The required values of variables at *L* and *R* side of edge *PQ* are equal to the variable values of the *L* and *R* nodes, respectively, for a first order accurate scheme. In a higher order scheme the appropriate *L* and *R* values are reconstructed with the Taylor series expansion, with a formulation based on the MUSCL (Monotonic Upstream Scheme for Conservation Laws) approach [Bla01]. Since no strong discontinuities are expected in an incompressible flow field, no limiter function is applied to the higher order scheme. Hence, the left and right state of a flow variable *U* at the midpoint of *PQ* can be evaluated as [Bar92] [And94] [Bla01] [ANSYS06] [Lyg13] [Sar14]

$$U_L^{PQ} = U_P + \frac{1}{2} \cdot (\nabla U)_L \cdot \vec{r}_{PQ} \tag{3.24}$$

$$U_R^{PQ} = U_Q + \frac{1}{2} \cdot (\nabla U)_R \cdot \vec{r}_{PQ} \tag{3.25}$$

where the gradients $(\nabla U)_L$ and $(\nabla U)_R$ represent in case of a second order scheme the gradients at $(\nabla U)_P$ and $(\nabla U)_Q$, respectively, while the vector $\vec{r}_{PQ}$ is the vector from node *P* to node *Q*. For the evaluation of the gradients of variables an approach based on the Green-Gauss theorem is used [Bar92] [Bla01]:

$$(\nabla U)_P = \frac{1}{\Omega_p} \sum_{Q \epsilon K_N(P)} \frac{1}{2}\left(U_P + U_Q\right) \cdot \vec{n}_{PQ} \tag{3.26}$$

3.2.2 Viscous fluxes

For the calculation of viscous fluxes, the evaluation of the velocity components' gradients at the middle of each edge of the computational grid is essential. For this purpose, two methodologies were employed in this work; a more accurate, but tedious element based method [Kal96] [Kal05] [Bla01] [Kal05] [Lyg14c] [Sar14] and a less accurate but significantly faster nodal-averaging one [Bla01] [Lyg14c] [Sar14].

For the first methodology a new type of control volume, called the edge-dual volume, must be constructed initially. This new type includes the elements of the grid that share a common edge [Kal96] [Kal05] [Sar14], as is obvious in Figure 3.5, where the equivalent edge-dual volume of an edge *e* shared by three tetrahedral elements, a prismatic and a pyramidal one is presented from two different viewpoints.

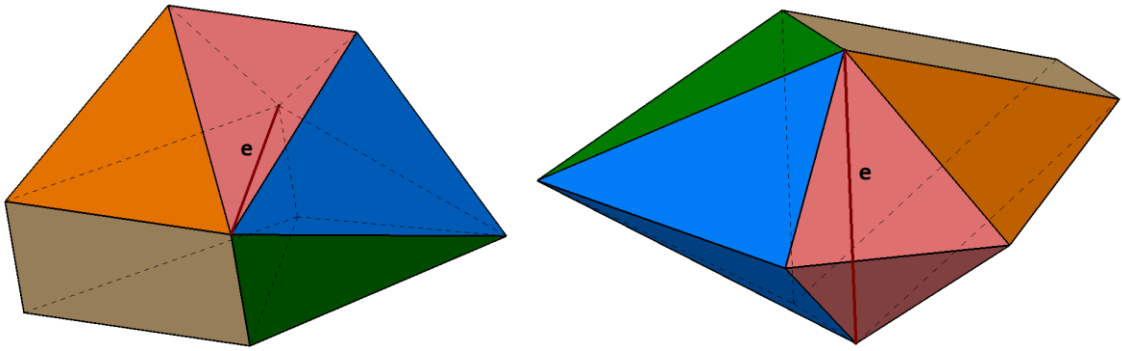


Figure 3.5 – Edge-dual control volume of an edge *e* that is shared by three tetrahedral, one prismatic and one pyramidal element.

Considering the aforementioned control volume configuration, the derivative of a velocity component (u) with respect to a Cartesian direction (x) at the midpoint of edge *e* can be evaluated using the divergence theorem as [Kal96] [Kal05] [Ahn06] [Lyg14c] [Sar14]

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{1}{V_e} \oint_e u n_x dS = \sum_{k=1}^{m} n_{x,k} \sum_{l=1}^{L_k} \frac{u_l}{L_k} \tag{3.27}$$

where *m* represents the number of boundary faces of the edge-dual volume, and L_k is the number of nodes that constitute each boundary face; the average of the velocity component on each face is required for this computation. While this method is proved [Kal05] to yield more accurate approximations of the velocity derivatives, it is also very demanding computationally, as a face-loop is required, which is not consistent with the edge-structure of the current flux balance evaluation.

To remedy the aforementioned case, a second methodology of node-averaging was made available, as a faster but less accurate alternative to the edge-dual method. In this case the gradients at the endpoints *P* and *Q* of an edge are utilized, as well as the directional derivative along the edge *PQ*. The mathematical formulation is presented as follows [Bla01] [Lyg14c] [Sar14]

$$(\nabla U)_{PQ} = \overline{(\nabla U)}_{PQ} - \left[\overline{(\nabla U)}_{PQ} \cdot \hat{r}_{PQ} - \left(\frac{\partial U}{\partial l}\right)_{PQ}\right]\hat{r}_{PQ} \tag{3.28}$$

$$\overline{(\nabla U)}_{PQ} = \frac{1}{2}\left((\nabla U)_P + (\nabla U)_Q\right) \tag{3.29}$$

$$\left(\frac{\partial U}{\partial l}\right)_{PQ} \approx \frac{U_Q - U_P}{|\vec{r}_{PQ}|} \tag{3.30}$$

$$\hat{r}_{PQ} = \frac{\vec{r}_{PQ}}{|\vec{r}_{PQ}|} \tag{3.31}$$

where $\vec{r}_{PQ}$ represents the vector connecting the nodes *P* and *Q*, while the gradients $(\nabla U)_P$ and $(\nabla U)_Q$ are the velocity components' gradients at the endpoints of edge *PQ*. This scheme is particularly attractive since it can be completed with a single edge-loop; the same loop that the inviscid fluxes are evaluated with and the utilized velocity gradients are already evaluated for the higher order inviscid fluxes, anyway. Moreover, although it provides less accurate results than the element-based model, these results have been proven adequate in all cases.

3.2.3 Turbulent fluxes

Fluxes for the turbulence model are calculated in the same node-centered finite-volume discretization scheme as with the flow model. The convective fluxes are evaluated with a simple first-order upwind scheme, at the middle of each edge *PQ* as [Lar91] [And94] [Koo00] [ANSYS06] [Lyg14a] [Sar14]:

$$\begin{gathered}\vec{f}_{i_{PQ_{SST}}} = \Theta^+ \vec{Q}_{P_{SST}} + \Theta^- \vec{Q}_{Q_{SST}} \\ \Theta^+ = \max(\Theta, 0), \quad \Theta^- = \min(\Theta, 0)\end{gathered} \tag{3.32}$$

Generally, a higher order scheme is not necessary for the above term, as the viscous terms are more dominant in the turbulence PDE's, so the numerical diffusion imported through the first order-scheme is easily neglected.

For the computation of the more important viscous fluxes at the middle of each edge *PQ*, the gradients of the corresponding turbulence variables (*k*, *ω*) must be primarily calculated in the

same way that the velocity gradients are calculated for the flow equations. Then, the viscous fluxes are calculated for an edge *PQ* as [Kou03]:

$$\vec{f}_{v_{SST}} = \begin{bmatrix} \left(\frac{\mu}{Re} + \frac{\mu_t}{\sigma_k}\right)\left(\frac{\partial k}{\partial x}n_x + \frac{\partial k}{\partial y}n_y + \frac{\partial k}{\partial z}n_z\right) \\ \left(\frac{\mu}{Re} + \frac{\mu_t}{\sigma_\omega}\right)\left(\frac{\partial \omega}{\partial x}n_x + \frac{\partial \omega}{\partial y}n_y + \frac{\partial \omega}{\partial z}n_z\right) \end{bmatrix} \tag{3.33}$$

3.2.4 Boundary conditions

In order to complete the flux balance on all no+des of the computational mesh, additional fluxes must be calculated for nodes that reside at the boundary of the domain from the direction of the boundary surfaces. There are four different types of boundary conditions encountered in this study: for solid wall, symmetry, inlet and outlet types of boundaries. The evaluation of all types of fluxes is carried out at the barycenter of the boundary surfaces, while subsequently the flux is equally distributed among the number of nodes of the boundary cell. The values of the variables used for the calculation of fluxes are derived either from the far-field, or from inside of the computational domain. The far field values are defined prior to the beginning of the simulation, while values from the inside of the domain are calculated as the average of the corresponding variables from all the nodes of the boundary face.

3.2.4.a Inlet and Outlet boundaries

At inlet and outlet boundaries a locally one-dimensional characteristic type of boundary conditions is used. For this type of boundary conditions the characteristic variables are calculated at the boundary surfaces of the inlet or outlet regions and subsequently are used for the calculation of fluxes on the boundary nodes. The derivation of this type of boundary condition for 3D grids is partially based on the work of Anderson et al. [And96] for 2D grids. For the identification of the characteristic variables the linearized one-dimensional Euler equation are taken into consideration [Lan98]

$$\frac{\partial \vec{q}}{\partial t} + A\frac{\partial \vec{q}}{\partial x_n} = 0 \tag{3.34}$$

where *A* is the flux Jacobian and x_n represents the normal to the boundary coordinate. The system of equations is considered hyperbolic if and only if matrix *A* is diagonizable, so it can be written in the following form [Lan98]

$$\mathrm{A} = \frac{\partial \vec{f_\iota}}{\partial \vec{q}} = T\Lambda T^{-1} \tag{3.35}$$

where T and T^{-1} are the left and right eigenvector matrices of the Jacobian matrix, while Λ is its eigenvalue diagonal matrix. Equation (3.35) entails the following [Lan98]:

$$\mathrm{A} = T\Lambda T^{-1} \Rightarrow\ T^{-1}AT = T^{-1}T\Lambda T^{-1}T \Longrightarrow \Lambda = T^{-1}AT \tag{3.36}$$

Thus, by multiplying Equation (3.34) with T^{-1} from the left side, and generating a product of matrices $I = T \cdot T^{-1}$, the following is implied:

$$T^{-1}\frac{\partial \vec{q}}{\partial t} + T^{-1}A\,\frac{\partial \vec{q}}{\partial x_n} = 0 \Rightarrow$$
$$T^{-1}\frac{\partial \vec{q}}{\partial t} + T^{-1}ATT^{-1}\,\frac{\partial \vec{q}}{\partial x_n} = 0 \overset{(3.36)}{\Longrightarrow}$$
$$T^{-1}\frac{\partial \vec{q}}{\partial t} + \Lambda T^{-1}\frac{\partial \vec{q}}{\partial x_n} = 0 \tag{3.37}$$

So, if the characteristic variables vector $\vec{w}$ is defined as $\partial\vec{w} = T^{-1}\partial\vec{q}$, the characteristics form of Equation (3.34) can be written as

$$\frac{\partial \vec{w}}{\partial t} + \Lambda\frac{\partial \vec{w}}{\partial x_n} = 0 \tag{3.38}$$

To identify the characteristic variables the characteristics vector is analyzed by its definition as follows:

$$\partial\vec{w} = T^{-1}\partial\vec{q} \;\Rightarrow \begin{bmatrix}\partial w_1\\ \partial w_2\\ \partial w_3\\ \partial w_4\end{bmatrix} = T^{-1}\begin{bmatrix}\partial q_1\\ \partial q_2\\ \partial q_3\\ \partial q_4\end{bmatrix} \tag{3.39}$$

If the matrix T^{-1} is equal to the left eigenvector of the Roe's averaged flux Jacobian defined in Section 3.2.1, then the above set of equations can be expanded as:

$$\partial w_1 = T_o^{-1}(1,1)\partial p + T_o^{-1}(1,2)\partial u + T_o^{-1}(1,3)\partial v + T_o^{-1}(1,4)\partial w \qquad \text{for } \frac{dx_n}{dt} = \lambda_1 = \Theta \tag{3.40}$$

$$\partial w_2 = T_o^{-1}(2,1)\partial p + T_o^{-1}(2,2)\partial u + T_o^{-1}(2,3)\partial v + T_o^{-1}(2,4)\partial w \qquad \text{for } \frac{dx_n}{dt} = \lambda_2 = \Theta \tag{3.41}$$

$$\partial w_3 = T_o^{-1}(3,1)\partial p + T_o^{-1}(3,1)\partial u + T_o^{-1}(3,1)\partial v + T_o^{-1}(3,1)\partial w \quad \text{for } \frac{dx_n}{dt} = \lambda_3 = \Theta + c \tag{3.42}$$

$$\partial w_4 = T_o^{-1}(4,1)\partial p + T_o^{-1}(4,1)\partial u + T_o^{-1}(4,1)\partial v + T_o^{-1}(4,1)\partial w \quad \text{for } \frac{dx_n}{dt} = \lambda_4 = \Theta - c \tag{3.43}$$

where $T_o^{-1}(i,j)$ represents the element of matrix T^{-1} at row *i* and column *j* and λ_k ($k=1,\ldots,4$) represents the respective eigenvalue of A. The elements of T^{-1} are considered constant in Equations (3.40) – (3.43), in order to maintain the linearization of the fluxes, so the subscript "*o*" is added to them to separate them from the variables, that have no subscript. The curves $dx = \lambda_i dt$ are called wave-fronts or characteristics [Lan98], while the characteristic variables $\vec{w}$ are the signals or information carried by the waves, with wave speeds equal to the eigenvalues λ_i. The direction of each wave is defined by the corresponding eigenvalue sign; a positive eigenvalue suggests the propagation of the respective wave towards the positive direction of $\vec{n}_{out}$. The characteristic variables can then be extracted by integrating Equations (3.40) – (3.43), implying the following set of equations:

$$\int \partial w_i = \int T_o^{-1}(i,1)\partial p + \int T_o^{-1}(i,2)\partial u + \int T_o^{-1}(i,3)\partial v + \int T_o^{-1}(i,4)\partial w \Rightarrow$$
$$w_i = T_0^{-1}(i,1)p + T_0^{-1}(i,2)u + T_0^{-1}(i,3)v + T_0^{-1}(i,4)w \quad (3.44)$$
$$i = 1,\dots,4$$

Considering the integration of the equation for the first characteristic variable, $T_0^{-1}(1,1)$ can be written as:

$$T_0^{-1}(1,1) = \frac{1}{c_0^2}\left[n_{x\,out}(w_0 y_2 - v_0 z_2) + n_{y_{out}}(u_0 z_2 - w_0 x_2) + n_{z\,out}(v_0 x_2 - u_0 y_2)\right] =$$
$$\frac{1}{c_0^2}\begin{vmatrix} n_{x_{out}} & n_{y_{out}} & n_{y_{out}} \\ x_2 & y_2 & z_2 \\ u_0 & v_0 & w_0 \end{vmatrix} = \frac{1}{c_0^2}\cdot \hat{n}_{out}\cdot\left(\hat{b}\times\vec{V}_o\right) = \frac{1}{c_0^2}\cdot\vec{V}_o\cdot\left(\hat{n}_{out}\times\hat{b}\right) = -\frac{1}{c_0^2}\cdot\vec{V}_o\cdot\hat{a} \quad (3.45)$$

where $\vec{n}_{out}$ is the outward-pointing normal to the boundary surface vector, as presented in Figure 3.6. The unit normal vector $\hat{n}_{out}$ can be calculated with the following formula:

$$\hat{n}_{out} = \frac{\vec{n}_{out}}{|\vec{n}_{out}|} \quad (3.46)$$

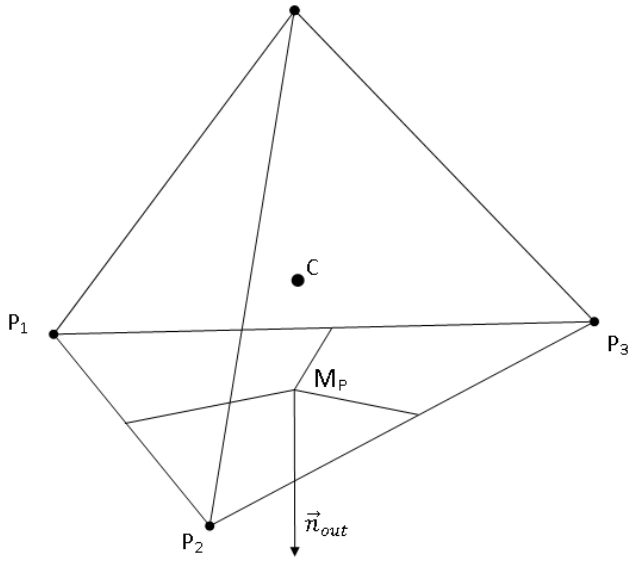


Figure 3.6 – Normal to the boundary $P_1P_2P_3$ normal vector $\vec{n}_{out}$.

Vector $\vec{V} = (u, v, w)$ represents the velocity vector, while $\hat{a}$ and $\hat{b}$ are unit vectors, normal to $\hat{n}_{out}$, as described in Equation (3.21). The internal product $\vec{V}\cdot\hat{a}$ is similar to the normal to the boundary surface velocity $\Theta = \vec{V}\cdot\hat{n}_{out}$, and from now on will be represented by Φ_1. Hence

$$T_0^{-1}(1,1) = -\frac{1}{c_0^2}\cdot\vec{V}_o\cdot\hat{a} = -\frac{1}{c_o^2}\Phi_{1_o} \tag{3.47}$$

The three right terms of Equation (3.40) can then be formulated as follows:

$$T_0^{-1}(1,2)u + T_0^{-1}(1,3)v + T_0^{-1}(1,4)w =$$

$$\frac{1}{c_0^2}\left\{\begin{bmatrix}\beta\left(n_{z_{out}}y_2 - n_{y_{out}}z_2\right)\\ +\Theta_0(w_0y_2 - v_0z_2)\end{bmatrix}u + \begin{bmatrix}\beta\left(n_{x_{out}}z_2 - n_{z_{out}}x_2\right)\\ +\Theta_0(u_0z_2 - w_0x_2)\end{bmatrix}v + \begin{bmatrix}\beta\left(n_{y_{out}}x_2 - n_{x_{out}}y_2\right)\\ +\Theta_0(v_0x_2 - u_0y_2)\end{bmatrix}w\right\}$$

$$=$$

$$\frac{1}{c_0^2}\begin{bmatrix}\beta\left[\left(n_{z_{out}}y_2 - n_{y_{out}}z_2\right)u + \left(n_{x_{out}}z_2 - n_{z_{out}}x_2\right)v + \left(n_{y_{out}}x_2 - n_{x_{out}}y_2\right)w\right]\\ +\Theta_0[(w_0y_2 - v_0z_2)u + (u_0z_2 - w_0x_2)v + (v_0x_2 - u_0y_2)w]\end{bmatrix} =$$

$$\frac{1}{c_0^2}\left[\beta\begin{vmatrix}u & v & w\\ x_2 & y_2 & z_2\\ n_{x_{out}} & n_{y_{out}} & n_{z_{out}}\end{vmatrix} + \Theta_0\begin{vmatrix}u & v & w\\ x_2 & y_2 & z_2\\ u_0 & v_0 & w_0\end{vmatrix}\right] =$$

$$\frac{1}{c_0^2}\left[\beta\cdot\vec{V}\cdot\left(\hat{b}\times\hat{n}_{out}\right) + \left(\vec{V}_o\cdot\hat{n}_{out}\right)\cdot\vec{V}\cdot\left(\hat{b}\times\vec{V}_o\right)\right] =$$

$$\frac{1}{c_0^2}\left[\beta\cdot\vec{V}\cdot\hat{a} + (\vec{V}_o\cdot\hat{n}_{out})\cdot\hat{b}\cdot\left(\vec{V}_o\times\vec{c}\right)\right] \overset{\hat{n}_{out}\cdot\hat{b}=0}{\Longrightarrow}$$

$$T_0^{-1}(1,2)u + T_0^{-1}(1,3)v + T_0^{-1}(1,4)w = \frac{1}{c_0^2}\beta\cdot\Phi_1 \tag{3.48}$$

Eventually, Equation (3.40) is written as:

$$w_1 = -\frac{1}{c_0^2}\Phi_{1_0}p + \frac{1}{c_0^2}\beta\Phi_1 \tag{3.49}$$

Similarly, it can be easily shown that Equation (3.41) yields the following formulation

$$w_2 = -\frac{1}{c_0^2}\Phi_{2_0}p + \frac{1}{c_0^2}\beta\Phi_2 \tag{3.50}$$

where Φ_2, analogously to Φ_1, is equal to $\vec{V}\cdot\hat{b}$. Finally, the integration of Equations (3.42) and (3.43) yields the following:

$$\int\partial w_3 = \int\frac{1}{2c_o^2}(c_0 - \Theta_0)\,\partial p + \int\frac{1}{2c_o^2}\beta\partial\Theta \Rightarrow w_3 = \frac{1}{2c_0^2}(c_0 - \Theta_0)p + \frac{1}{2c_0^2}\beta\Theta \tag{3.51}$$

$$\int\partial w_4 = -\int\frac{1}{2c_o^2}(c_0 + \Theta_0)\,\partial p + \int\frac{1}{2c_o^2}\beta\partial\Theta \Rightarrow w_4 = -\frac{1}{2c_0^2}(c_0 + \Theta_0)p + \frac{1}{2c_0^2}\beta\Theta \tag{3.52}$$

The complete set of equations is presented below:

$$\begin{cases} w_1 = -\frac{1}{c_0^2}\Phi_{1_0}p + \frac{1}{c_0^2}\beta\Phi_1 & \text{for } \frac{dx}{dt} = \lambda_1 = \Theta \quad (3.53) \\ w_2 = -\frac{1}{c_0^2}\Phi_{2_0}p + \frac{1}{c_0^2}\beta\Phi_2 & \text{for } \frac{dx}{dt} = \lambda_2 = \Theta \quad (3.54) \\ w_3 = \frac{1}{2c_0^2}(c_0 - \Theta_0)p + \frac{1}{2c_0^2}\beta\Theta & \text{for } \frac{dx}{dt} = \lambda_3 = \Theta + c \quad (3.55) \\ w_4 = -\frac{1}{2c_0^2}(c_0 + \Theta_0)p + \frac{1}{2c_0^2}\beta\Theta & \text{for } \frac{dx}{dt} = \lambda_4 = \Theta - c \quad (3.56) \end{cases}$$

Considering the eigenvalue signs, the third eigenvalue (λ_3) will be always positive, since $c \geq \Theta$. Hence, the corresponding wave will propagate the third characteristic variable towards the positive direction of $\vec{n}_{out}$, which is outwards of the computational domain. That means that at the barycenter of the boundary cell w_3 will have a value derived from the inside of the computational domain. In the same manner, the fourth eigenvalue will always be negative, so the corresponding wave will propagate the signal w_4 towards the negative direction of $\hat{n}_{out}$, which means that at the middle of the boundary cell w_4 will have a value derived from the free-stream. The first and second eigenvalues' signs depend on the location of the boundary cell; if it resides at the inflow, Θ will have a negative value, because $\vec{V}$ will have opposite direction from $\hat{n}_{out}$, so the characteristic variable will have a value that is derived from the far-field, while at the outflow the characteristic variable value will be derived from inside the computational domain for the exact opposite reason. The above relations provide four equations, with unknowns the pressure *p*, the normal velocity Θ and the tangential velocities at the boundary Φ_1 and Φ_2.

$$\begin{cases} w_{1_b} = w_{1_r} \\ w_{2_b} = w_{2_r} \\ w_{3_b} = w_{3_i} \\ w_{4_b} = w_{4_\infty} \end{cases} \Rightarrow \begin{cases} -\frac{1}{c_0^2}\Phi_{1_0}p_b + \frac{1}{c_0^2}\beta\Phi_{1_b} = -\frac{1}{c_0^2}\Phi_{1_0}p_r + \frac{1}{c_0^2}\beta\Phi_{1_r} & (3.57) \\ -\frac{1}{c_0^2}\Phi_{2_0}p_b + \frac{1}{c_0^2}\beta\Phi_{2_b} = -\frac{1}{c_0^2}\Phi_{2_0}p_r + \frac{1}{c_0^2}\beta\Phi_{2_r} & (3.58) \\ \frac{1}{2c_0^2}(c_0 - \Theta_0)p_b + \frac{1}{2c_0^2}\beta\Theta_b = \frac{1}{2c_0^2}(c_0 - \Theta_0)p_i + \frac{1}{2c_0^2}\beta\Theta_i & (3.59) \\ -\frac{1}{2c_0^2}(c_0 + \Theta_0)p_b + \frac{1}{2c_0^2}\beta\Theta_b = -\frac{1}{2c_0^2}(c_0 + \Theta_0)p_\infty + \frac{1}{2c_0^2}\beta\Theta_\infty & (3.60) \end{cases}$$

The above equations can also be expressed in matrix form (with the appropriate simplifications) below:

$$\begin{bmatrix} -\Phi_{1_0} & 0 & \beta & 0 \\ -\Phi_{2_0} & 0 & 0 & \beta \\ (c_0 - \Theta_0) & \beta & 0 & 0 \\ -(c_0 + \Theta_0) & \beta & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_b \\ \Theta_b \\ \Phi_{1_b} \\ \Phi_{2_b} \end{bmatrix} = \begin{bmatrix} -\Phi_{1_0}p_r + \beta\Phi_{1_r} \\ -\Phi_{2_0}p_r + \beta\Phi_{2_r} \\ (c_0 - \Theta_0)p_i + \beta\Theta_i \\ -(c_0 + \Theta_0)p_\infty + \beta\Theta_\infty \end{bmatrix} \quad (3.61)$$

In the above equations the subscript *b* denotes the variables that are evaluated at the barycenter of the boundary cell, while variables with subscript *o* are considered constant and in this work are

derived from the far-field. Also, the subscript ∞ denotes reference data taken from the free-stream and *i* denotes data taken from the inside of the computational domain; these are calculated as the average of the corresponding variables at the nodes of the boundary cell. Finally, special consideration is taken for data marked with the *r* subscript; data is taken from the free-stream in case of inflow, while for outflow data from inside the computational domain is used. The unknowns of the above set of equations are evaluated directly with the use of the following formulas:

$$\begin{cases} p_b = \frac{1}{2}(p_i + p_\infty) + \frac{\beta}{2c_0}(\Theta_i - \Theta_\infty) + \frac{1}{2c_0}(p_\infty - p_i) & (3.62) \\ \Theta_b = \frac{\Theta_i}{2c_0}(c_0 + \Theta_0) + \frac{\Theta_\infty}{2c_0}(c_0 - \Theta_0) + \frac{1}{2c_0}(p_i - p_\infty) & (3.63) \\ \Phi_{1_b} = \Phi_{1_r} + \frac{\Phi_{1_0}}{2c_0}(\Theta_i - \Theta_\infty) + \frac{\Phi_{1_0}}{2\beta}(p_i + p_\infty - 2p_r) + \frac{\Phi_{1_0}\Theta_0}{2\beta c_0}(p_\infty - p_i) & (3.64) \\ \Phi_{2_b} = \Phi_{2_r} + \frac{\Phi_{2_0}}{2c_0}(\Theta_i - \Theta_\infty) + \frac{\Phi_{2_0}}{2\beta}(p_i + p_\infty - 2p_r) + \frac{\Phi_{2_0}\Theta_o}{2\beta c_0}(p_\infty - p_i) & (3.65) \end{cases}$$

Finally, the enforcement of the inflow and outflow boundary conditions is performed through the evaluation of the appropriate inviscid fluxes at the boundary cells, while the viscous fluxes are neglected, as their role is not so important so far away from solid walls. The required flow data (p_b, u_b, v_b, w_b) can be easily extracted from the calculated variables (p_b, Θ_b, Φ_{1b}, Φ_{2b}), as they are directly linked. Specifically, while pressure at the boundary is directly calculated, the velocity components can be calculated by the normal and shear velocities as shown below:

$$\begin{cases} \Theta_b = u_b n_{x_{out}} + v_b n_{y_{out}} + w_b n_{z_{out}} & (3.66) \\ \Phi_{1_b} = u_b x_1 + v_b y_1 + w_b z_1 \quad \Rightarrow & (3.67) \\ \Phi_{2_b} = u_b x_2 + v_b y_2 + w_b z_2 & (3.68) \end{cases}$$

$$\begin{cases} u_b = \Phi_{1_b}\left(n_{z_{out}} y_2 - n_{y_{out}} z_2\right) + \Phi_{2_b}\left(n_{y_{out}} z_1 - n_{z_{out}} y_1\right) + \Theta_b(y_1 z_2 - y_2 z_1) & (3.69) \\ v_b = -\left[\Phi_{1_b}\left(n_{z_{out}} x_2 - n_{x_{out}} z_2\right) + \Phi_{2_b}\left(n_{x_{out}} z_1 - n_{z_{out}} x_1\right) + \Theta_b(x_1 z_2 - x_2 z_1)\right] & (3.70) \\ w_b = \Phi_{1_b}\left(n_{y_{out}} x_2 - n_{x_{out}} y_2\right) + \Phi_{2_b}\left(n_{x_{out}} y_1 - n_{y_{out}} x_1\right) + \Theta_b(x_1 y_2 - x_2 y_1) & (3.71) \end{cases}$$

For the turbulence model, Dirichlet type boundary conditions are imposed at the inlet of the computational domain, with the turbulence variables being set to zero on each boundary node. In case of outlet boundary conditions, the inviscid turbulent fluxes are calculated via a simple upwind scheme as [Kim03] [Kou03]:

$$\vec{f}_{i_{SST}}^{out} = \Theta \vec{Q}_{SST} \tag{3.72}$$

where Θ is the vertical to the boundary surface velocity, as explained above. The viscous turbulent fluxes are neglected in the same way as with the flow model.

3.2.4.b Solid wall and symmetry boundaries

For solid wall boundaries, two cases of boundaries are considered; free-slip boundary conditions in case of inviscid flow, or no-slip boundary conditions in case of viscous flow. For the case of inviscid flow over a solid wall surface, the boundary condition is imposed implicitly by adding a flux with zero vertical to the boundary surface velocity, to the flux balance. The flux is calculated at the centroid of the face defined by boundary nodes *P1*, *P2* and *P3* as [Mav94]:

$$\vec{f}_i^{wall} = \begin{bmatrix} \beta\Theta \\ u\Theta + n_x p \\ v\Theta + n_y p \\ w\Theta + n_z p \end{bmatrix} = \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \end{bmatrix} \tag{3.73}$$

The same boundary condition is imposed in case of symmetry boundary conditions, where the flow is expected to evolve perpendicular to the boundary. For laminar and turbulent viscous flows, no-slip conditions are applied on solid wall boundaries, by explicitly setting the velocity components at the boundary nodes, equal to zero. For the turbulence model, turbulent kinetic energy and the turbulent kinetic viscosity v_t are also set to zero, while the specific dissipation rate ω is evaluated as [Men03]

$$\omega_{wall} = \frac{60}{\beta_1 {y_b}^2 Re} \tag{3.74}$$

where y_b is the distance of the boundary node from the closest non-boundary one.

3.3 Temporal discretization

For the integration in time of the flow PDEs, as well as the equation of the turbulence model, an explicit four-stage Runge-Kutta scheme (RK(4)) [Lal88] [Kal96] [Bla01] [Lyg14a] [Sar14] was incorporated in the proposed methodology. Since the flux balance is evaluated for each node of the computational grid, Equation 3.8 is transformed as:

$$\begin{aligned} &-\left(\frac{\partial \vec{Q}}{\partial \tau}\right)_P \Omega_P = -\Omega_P \frac{\Delta \vec{Q}_P^{n+1}}{\Delta \tau_P} = \\ &\sum_{Q \epsilon K_N(P)} \vec{f}_i(\vec{Q}_L, \vec{Q}_R; \hat{n}) - \sum_{Q \epsilon K_N(P)} \vec{f}_v(\vec{Q}_L, \vec{Q}_R; \hat{n}) - \iiint_{\Omega} \vec{S}\, dV = \overrightarrow{RHS}_P^n \end{aligned} \tag{3.75}$$

where $\Delta\tau_P$ is the local time step at node *P*, which is differently calculated in case of inviscid or viscous flows. The corresponding formulas for the calculation of the time step are presented below [Kal05]:

$$\Delta\tau_P^{inv} = CFL \cdot \frac{0.5 l_{\text{minedge,P}}}{\left|\vec{\text{V}}_\text{P}\right| + \text{c}_\text{P}} \tag{3.76}$$

$$\Delta\tau_P^{vis} = CFL \cdot \frac{\Omega_P}{\text{A}_\text{x} + \text{A}_\text{y} + \text{A}_\text{z} + \text{D}} \tag{3.77}$$

$$A_x = (|u| + c_x)S_x, \qquad A_y = \left(|v| + c_y\right)S_y, \qquad A_z = (|w| + c_z)S\_z \tag{3.78}$$

$$D = \frac{2}{Re}\frac{\Omega_P}{S_x + S_y + S_z}$$

In the above equations, the length $l_{minedge,P}$ is the length of the shortest edge connected to node *P*, while c_P is the artificial speed of sound at the same node. For the calculation of the CFL number in case of viscous flows, the quantities denoted by c_x, c_y and c_z represent the artificial speed of sound calculated in each Cartesian coordinate direction at node *P*, as [Kal05]:

$$c_x = \sqrt{u^2 + \beta}, \qquad c_y = \sqrt{v^2 + \beta}, \qquad c_z = \sqrt{w^2 + \beta} \tag{3.79}$$

The quantities S_x, S_y and S_z represent the projections of the node *P* node-dual volume surfaces normal vector along the three Cartesian directions. They can be calculated as [Kal05]:

$$S_x = \frac{1}{2}\sum_e |S_x|_e\,, \qquad S_y = \frac{1}{2}\sum_e |S_y|_e\,, \qquad S_z = \frac{1}{2}\sum_e |S_z|_e \tag{3.80}$$

The local time-stepping scheme ensures that time integration on each node can be performed at a maximum acceptable pseudo-time step, ensuring a fastest convergence to a steady-state solution. Employment of the explicit Runge-Kutta scheme must then be considered separately in case of a time-accurate solution.

3.3.1 Steady State solution

In case of a steady-state solution, Equation (3.75) is solved iteratively with an explicit four-stage Runge-Kutta scheme as:

$$\begin{gathered} \vec{Q}_P^{n+1,0} = \vec{Q}_P^n \\ \vec{Q}_P^{n+1,k} = \vec{Q}_P^n - a_k \frac{\Delta\tau_P}{\Omega_P}\overrightarrow{RHS}_P^{n+1,k-1}, \qquad k = 1, \dots, 4 \\ \vec{Q}_{\mathrm{P}}^{\mathrm{n+1}} = \vec{\mathrm{Q}}_{\mathrm{P}}^{\mathrm{n+1,4}} \end{gathered} \tag{3.81}$$

where *k* is the number of the internal Runge-Kutta iteration, while the four constants α_k have the values α_1=0.11, α_2=0.26, α_3=0.5, α_4=1.0, that achieve a second-order temporal discretization scheme [Bla01].

3.3.2 Unsteady State solution – dual time stepping.

In case of a time-marching scheme, Equation (2.10) is discretized instead. Spatial discretization is carried out in the same manner as with the steady-state equations. However, a different procedure must be followed in case of the temporal discretization, where two temporal variables are preset in the flow equations. Through a similar formulation with the steady-state formulas, Equation (2.10) would then become [Rog91] [Kal05]:

$$\begin{gathered} -K\left(\frac{\partial\vec{Q}}{\partial t}\right)_P \Omega_P - \left(\frac{\partial\vec{Q}}{\partial\tau}\right)_P \Omega_P = \\ \sum_{Q\epsilon K_N(P)} \vec{f}_i(\vec{Q}_L, \vec{Q}_R; \hat{n}) - \sum_{Q\epsilon K_N(P)} \vec{f}_v(\vec{Q}_L, \vec{Q}_R; \hat{n}) - \iiint_\Omega \vec{S}\, dV = \overrightarrow{RHS}_P^n \end{gathered} \tag{3.82}$$

The right hand side of Equation (3.82) can then be evaluated separately for the case of the continuity equation and the momentum equations. In case of the momentum equations the real time derivatives of the flow variables are analyzed with a second-order, three-point backward-difference formula as [Kal05]

$$\left(\frac{\partial \vec{Q}}{\partial t}\right) = \frac{\left(3\vec{Q}_P^{m+1} - 4\vec{Q}_P^{m} + \vec{Q}_P^{m-1}\right)}{2\Delta t} \tag{3.83}$$

where the superscripts *m* corresponds to flow quantities derived at times $t=m\Delta t$. The time-accurate formulation can then be written as [Kal05]:

$$-\frac{\Delta \vec{Q}^{m+1,n+1}}{\Delta \tau}\Omega_P = K\frac{\left(3\vec{Q}_P^{m+1,n} - 4\vec{Q}_P^{m} + \vec{Q}_P^{m-1}\right)}{2\Delta t} + \overrightarrow{RHS}_P^{m+1,n+1} \tag{3.84}$$

In order to calculate the field variables in a new pseudo-time step *n+1* and real time step *m+1*, the real-time derivative of the variables vector has to be evaluated beforehand, by utilizing the variables vector at real time steps *m* and *m-1*, as well as the variable vector at real time step *m+1* and pseudo time step *n*. Then the right hand side term $\overrightarrow{RHS}_P^{m+1,n+1}$ has to be evaluated as the sum of fluxes at the current pseudo-time step, as with Equation (3.81). For the solution of the steady-state problem defined in Equation (3.85) a four stage Runge-Kutta scheme is used as with steady state solutions.

CHAPTER 4

ACCELERATION TECHNIQUES

Numerical simulations of incompressible flows with the artificial compressibility method tend to be extremely time-consuming, and even very simple test cases may need a huge amount of pseudo-time iterations to converge into a satisfactory solution. Especially in the case of time-dependent solutions, where a large number of real time steps may be needed for the unsteady flow phenomenon to develop, special measures must be taken for the acceleration of the numerical evaluation procedure. In this work a domain decomposition parallel processing approach was adopted, along with an agglomeration multigrid strategy. Both techniques contribute greatly to the acceleration of the incompressible solver and prove to be essential tools in any simulation.

4.1 Parallel Processing

4.1.1 Domain Decomposition - Partitioning

With the domain decomposition approach the initial computational grid is divided into a number of smaller ones, each attributed to a single processor core, constituting in this way a semi-autonomous computational grid on which the flow PDEs may be applied [Smi04] [Gro92]. For the complete definition of the flow, appropriate data, such as flow variables and their gradients must be defined at the inner boundaries of each sub-domain, as those data are computed at a neighboring grid, by another processor core [Ven95] [Lan96]; this task is performed via a message-passing inter-core communication method provided by the MPI (Message Passing Interface). In Figure 4.1 partitioning of a domain into eight smaller sub-domains is presented.

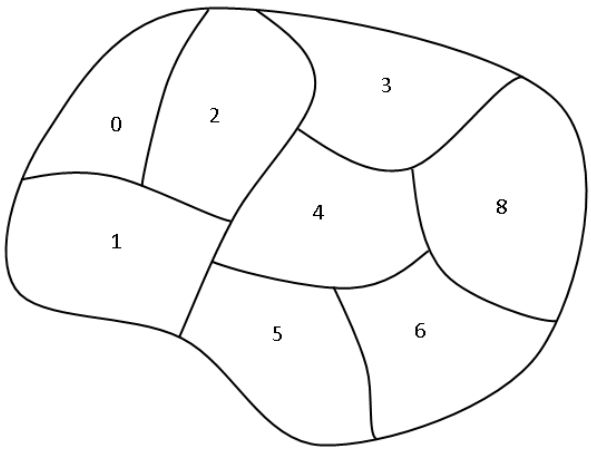


Figure 4.1 – Eight-way partitioning of initial domain.

The whole procedure begins with the implementation of METIS, which is a software program for partitioning large irregular graphs, large meshes, and computing fill-reducing orderings of sparse matrices, with algorithms based on the multilevel graph partitioning paradigm [Kar95]. For the partitioning of the initial domain, the mesh is handled as an undirected graph, where only the nodes and the edges connecting them are essential. The division is performed in a node-wise manner and the resulting sub-domains have more or less equal number of nodes, resulting in a load-balanced parallelization, while the minimum number of edge-cuts is performed ensuring the least possible exchange of data between partitions [Sar15].

In this work a single domain decomposition approach with overlapping is adopted [Smi04] [Mou11]. The sub-domains produced by the partitioning procedure are actually a number of sets of non-common nodes, called *core* nodes. For the implementation of the solver on these partitions, the element-based structure of the initial grid must be reproduced. However, due to the edge-cut that was performed, the elements that share nodes at the internal boundaries of a newly created sub-domain cannot be fully defined, since some of their initial connections reside on a neighboring partition. To remedy this, the missing nodes are added to the existing node list as *ghost* nodes, and the now well-defined elements will exist on both neighboring partitions, thus creating the so called *overlapping layer* [Lan96] [Sar15]. The *overlapping layer* between three adjacent partitions is presented for a 2D unstructured grid in Figure 4.2.

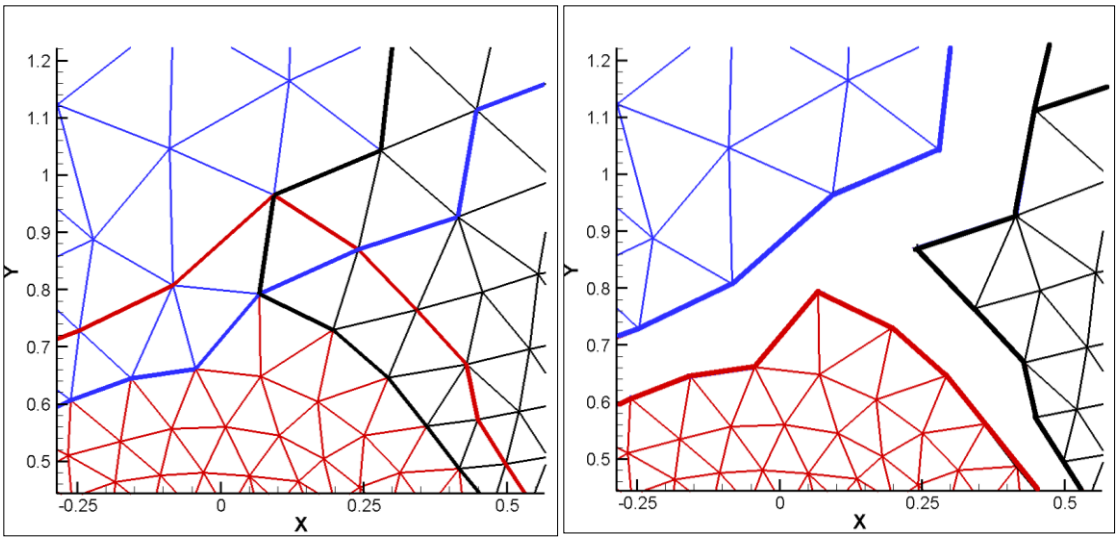


Figure 4.2 – Overlapping layer between three adjacent partitions for a 2D unstructured grid.

For this procedure, a serial algorithm assigned to a single process was created. The steps of the overlapping region construction algorithm are the following:

a) The *core* nodes of each sub-domain are renumbered based on the output of METIS.
b) The elements of the initial grid are assigned to sub-domains. Those that will form the overlapping region, as described above, are arbitrarily assigned a partition (if the first

node of an element is a core node of a sub-domain, then the element is assigned to that domain). The partition that receives an element is now called *owner* of that element.

c) The nodal connectivities of all elements of each sub-domain are defined from the renumbered *core* node indexes, with respect to the initial grid. If an element's connectivities are not fully defined this element will be part of the *overlapping layer*; a list of those elements is constructed.

d) Loop over the nodes of each element in the overlapping element list. If a node is a *core* node owned by a partition different than the element's owner then that element is added to that partition's list, but no nodes are defined for it. At the end of the loop all elements of the overlapping layer are distributed to neighboring sub-domains, but information for their nodes is missing.

e) Loop over the nodes of all the elements of each sub-domain. If current node information is missing from the element connectivities, then this node is added as a *ghost* node at the end of the list of nodes of the sub-domain and the missing information is adjusted. *Ghost* nodes are separated from *core* ones, as they are put at the end of the node list. At the end of the loop the overlapping layer is defined for all sub-domains.

4.1.2 Communication data structures

After the partitioning of the initial domain and the definition of the overlapping elements, information that each partition requires for communication at its inner boundaries must be computed [Ven95]. For this task, an $n \times n$ *communications matrix* is constructed, where n is the number of sub-domains. An element (i, j) of this matrix is equal to unity if the partition j is neighbor to i and zero otherwise. Construction of the *communications matrix* is achieved with a simple loop over the *ghost* nodes of each partition (i), where for each such node, the equivalent *core* node's *owner* can be easily identified as j. From this information, appropriate data structures, essential for the communication between partitions may be composed. These data structures consist of:

- **nneighbours**: an integer that represents the number of adjacent sub-domains.
- **ineighbours(i)**: a list of the adjacent sub-domains, in the form of an integer array with size equal to *nneighbours*.
- **nreceivenodes(i)**: an integer array with size equal to *nneighbours*, where element *i* holds the number of the partition's ghost nodes that will receive information from an equal number of *core* nodes of partition *i*.
- **nsendnodes(i)**: an integer array of size *nneighbours*, that lists the number of the partition's *core* nodes that will have to send information to an equal number of *ghost* nodes of adjacent partition *i*.
- **ireceivenodes(i, j)**: a matrix with *nneighbours* rows, where each row consists of pointers to the *ghost* nodes of this sub-domain that must receive information from the corresponding *core* nodes of the *i*th partition of array *ineighbours()*.
- **isendnodes(i, j)**: a matrix with *nneighbours* rows, where each row is filled with pointers to the partition's *core* nodes that must send information to the respective *ghost* nodes of the *i*th partition found in the array *ineighbours()*.

In order to ensure that exchange of data will be precise, and that information will be transferred without error from a *core* node to its equivalent *ghost* nodes of adjacent partitions, it is essential

that there exists an exact mapping between *core* nodes in matrix *isendnodes()* of a partition and *ghost* nodes in matrix *ireceivenodes()* of an adjacent partition. Therefore, it is ensured that the *core* node represented by the *j*th element in row *i* of matrix *isendnodes()* of a partition *ipart* will be the same node as the *ghost* one found in the *j*th column of row *k* of matrix *ireceivenodes()* of the partition *ineighbours(i)*, where *k* will satisfy *ineighbours(k)=ipart*.

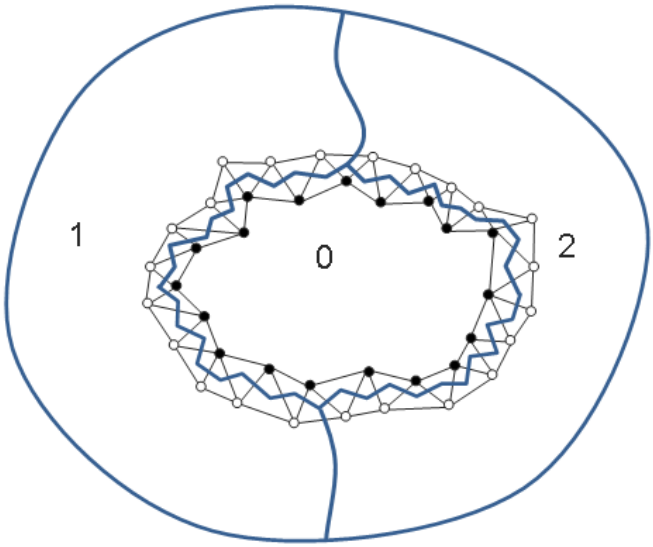


Figure 4.3 – The connectivities between the core and ghost nodes of adjacent partitions.

In Figure 4.3 a domain is divided into three partitions (*0*, *1*, and *2*). The overlapping region for partition *0* is visible. The *core* nodes of partition *0* marked with a solid circle, while its *ghost* nodes are marked with a hollow circle. According to their description the data structures for that partition would have the following values:

- **nneighbours** = 0
- **ineighbours(1)** = 1
- **ineighbours(2)** = 2
- **nreceivenodes(1)** = 12
- **nreceivenodes(2)** = 11
- **nsendnodes(1)** = 10
- **nsendnodes(2)** = 11

4.1.3 Communication Procedure – MPI

As mentioned in paragraph 4.1.1, the exchange of data required for the correct evaluation of the flow PDEs on all partitions is performed via the Message Passing Interface (MPI), which is a language based communications protocol implementing the message-passing model for the means of parallel computing [Gro99]. The message-passing model assumes a set of processes with local

memory only, which are able to communicate with other processes by sending and receiving messages. All processes executed in parallel have separate address spaces. Communication between two processes occurs when data from the address space of one is copied to the address space of another. The above operation occurs only when the first process posts a *send* operation and the second one posts a *receive* operation, so that the communication process is cooperative [Gro99].

The MPI protocol, as an implementation of the message-passing model provides a list of functions, for the point-to-point communication between processes, collective communication, as well as a number of utility functions for the organization of the processes and the exchanged data. A sample program structure using this protocol is presented in Figure 4.4. While for acceleration reasons it is always preferable for the biggest part (or actually the most calculations-heavy part) of the code to be implemented in parallel, initialization of the MPI environment, and hence the parallel program, is not necessary to happen at the start of the code, nor finish at the end of it. Parts of serial coding may be inserted before the MPI initialization or after its finalization. The parallel part of the code is separated in sections where the processes work separately with data from their address space, while it is possible for data exchange between them in any configuration (either point-to-point, or collectively).

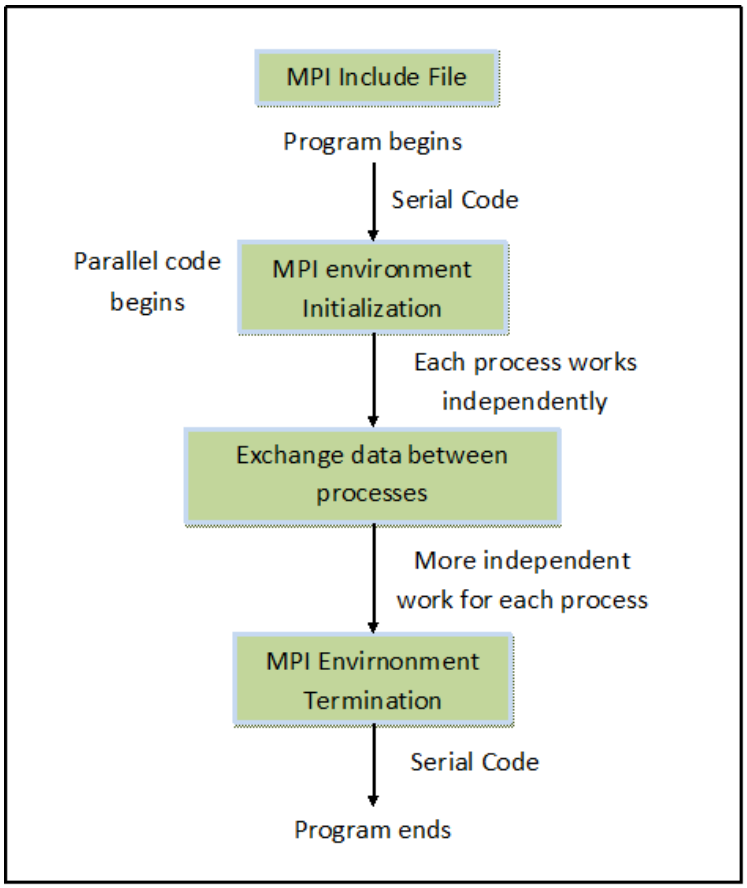


Figure 4.4 – Sample MPI program structure.

Each implementation of MPI either vendor or public domain, provides the means for execution of the parallel code from an operating system either through command line, or with a GUI. With this procedure the appropriate number of processes is generated and run in the operating system. In this work each sub-domain is assigned to a single process, which is assigned essentially on a single processor core. Since the message-passing procedure happens between processes and not between processor cores, it would be possible to generate more parallel processes than the available cores; however that would be unwise, since more than one process would take turns to be executed on a single core and the parallel program's efficiency would be degraded.

Each generated process runs the same parallel code, categorizing this parallel program as SPMD (Single Program Multiple Data) on Flynn's taxonomy classification [Flynn72] [Dar88]. For the identification of the processes and the context in which the exchange of data between processes will take place, the MPI execution environment must be initialized. This is performed with **MPI_INIT()**, which is a function that must be executed in an MPI program only once before any other MPI routine is called. With the execution of this function the **MPI_COMM_WORLD** *communicator* is defined.

A *communicator* is and opaque object with a number of attributes as well as simple rules that define its creation, use, and destruction, while it is responsible to determine the scope and *communication universe* in which a point-to-point or collective exchange may take place [Gro96]. It contains a group of valid participant processes, which is an ordered set of processes, each with a unique *rank* within the same communicator. The *rank* of a process within a certain *communicator* object is an integer number with value ranging from zero to N-1, where N is the number of processes in the communicator. The **MPI_COMM_WORLD** *communicator* contains all available processes of the parallel program.

The *rank* of a process within the **MPI_COMM_WORLD** *communicator* is retrieved with a call of the function **MPI_COMM_RANK(comm, rank)** by all available processes, where the **comm** argument defines the *communicator* object in which the *rank* is queried (hence in this case equal to **MPI_COMM_WORLD**). The size of the MPI universe (the number of processes – partitions in which the initial flow domain was divided) may be recovered by calling the function **MPI_COMM_SIZE(comm)**, where the argument **comm** must be equal to **MPI_COMM_WORLD**. A call of the same function with a different *communicator* object as an argument would provide the number of processes in that communicator.

As each process runs the same code, but eventually receives a unique *rank* within the **MPI_COMM_SIZE** universe – communicator, it is easy to assign different tasks to each process with simple if-blocks, as is demonstrated below:

```
call MPI_INIT()
call MPI_COMM_SIZE(MPI_COMM_WORLD, size)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank)

if(rank = = 0)
   do some preliminary work
   call MPI_COMM_BARRIER(MPI_COMM_WORLD)
   do some more work
```

```
else
   call MPI_COMM_BARRIER(MPI_COMM_WORLD)
   do some other work
end if

all processes do some work
```

In the above sample code, the function **MPI_COMM_BARRIER(comm)** helps for the synchronization of all processes, as it serves as a "waypoint" on which a process pauses execution until all other processes in the *communicator* **comm** execute the same function. Additionally, with an if-block the process with rank equal to zero is separated from the other processes and performs different tasks. In the *Galatea-I* program, *process zero* is assumed as a *master* process and it is charged with additional work regarding calculation of the PDEs residuals and presenting them on the computer screen, several other visual outputs, as well as the complete initial construction of the *overlapping layer* between partitions, the *communications data structures* and the distribution of all initial data to all the other processes. Distribution of data is then easy to perform within the MPI environment.

When process with *rank* zero finishes all the initial work of gathering the partition data and constructing the *overlapping layer*, it is charged with distributing all the required partition data to all the other processes. This includes the nodes' (*core* and *ghost*) coordinates, the element connectivities, boundary conditions, and the flow domain initial data, such as attack angles, the reference length, the real time step in case of an unsteady simulation, and the value of the artificial compressibility β. When distribution of data is complete, all the processes proceed with the initialization of their own sub-domains; calculation of appropriate control volumes, initial normalization of the variables and the computational domain, initialization of flow variables and imposition of boundary conditions on solid wall boundaries. Finally, flux calculation for the relaxation of the governing equations may be performed iteratively by each process on the *core* nodes of its domain. At the first pseudo-time iteration (t=0) the variable values on all nodes of the computational grid are equal to their initial values, therefore there is no need for exchange of data between *core* and *ghost* nodes, since their variable values are known. However, in case of a higher order scheme, the *core* nodes' gradients have to be calculated and their values passed to the respective *ghost* nodes of adjacent sub-domains, prior to the iterative procedure, to account for the existence of solid wall boundaries. At the end of each pseudo-time step, appropriate data exchange is performed between *core* and *ghost* nodes of adjacent sub-domains. The exchanged data consists of the flow variable vector $\vec{Q}$, the velocity derivatives in case of viscous flow and the Green-Gauss gradients in case of a higher spatial order scheme. The complete parallel program (the same instance run by each process) is described in pseudo-code form below:

```
Program Start

Initialization of the MPI environment
Obtain rank in MPI_COMM_WORLD for each process
Obtain the number of parallel processes

if(rank = = 0) then
   Read initial grid data
```

```
   Read METIS partition data
   Recreate all sub-domains
   Construct the overlapping layer for each partition
   Construct the communications data structures
   Send partition and initial flow data to the other processes
   call MPI_BARRIER(MPI_COMM_WORLD)
else
   Receive partition and initial flow data from process 0
   call MPI_BARRIER(MPI_COMM_WORLD)
endif

//Iterative procedure
Do i = 1 to Number of iterations
   Do j = 1 to iterative method steps
      Perform relaxation of governing equations
      call MPI_BARRIER(MPI_COMM_WORLD)
      Exchange appropriate data
   End Do
End Do

End Program
```

Data exchange with the MPI may be performed in two ways; either with **point-to-point communication**, where a process sends data to another single process, or with **collective communication**, where a process may send the same set of data or distribute pieces of a set of data to all the other processes sharing the same *communicator* object. Whichever type of communication is used, the equivalent *send* or *receive* functions must be executed by all the participating processes [Gro96].

With point-to-point communication, messages are passed between two processes, where one of them serves as the *sender* and the other as the *receiver*. While there are different types of send and receive routines used for different purposes, such as *synchronous* send, *blocking* or *non-blocking* send/receive, *buffered* send and *combined* send/receive, in this work only *blocking* send/receive routines are used, for means of security and synchronization. With this type of point-to-point communication a send routine always waits for the corresponding receive one to reply that the message has passed. The syntax of *blocking* send and receive routines is presented below:

- **MPI_SEND(buf, count, datatype, dest, tag, comm)**
- **MPI_RECV(buf, count, datatype, source, tag, comm, status)**

The message data passed with the **MPI_SEND** routine consists of **count** successive entries of the type indicated by **datatype** derived from the *sender* address space, starting with the entry at address **buf**. The message is accompanied with information that can be used to distinguish messages and selectively receive them, called the *message envelope*. This includes the **source** of the message, its destination **dest**, the **tag** argument which is an integer used to distinguish different types of messages and is user defined, and the *communicator* **comm** within which the *sender* (**source**) and *receiver* (**dest**) *ranks* may be found. Respectively, the message that will be received by the **MPI_RECV** routine will consist of **count** elements of **datatype** type that will be

stored in the memory addresses starting at **buf** (defined in the MPI_RECV routine). The receive process will "identify" the correct message by the *message envelope*, which was sent by the correct **source** within the same *communicator* **comm**, and carry the correct **tag**. Finally, additional information about the message may be derived by the **status** object, such as the message length, the tag, source, and error code of the received message. As the blocking point-to-point communication method involves only two processes (*sender* and *receiver*), and the *sender* must wait for the *receiver's handshake*, a routine where exchange of data between all the neighboring partitions would take place, must at a time select a *sender* from the list of processes (*ranks*) that will send its data to all the eligible *receivers*. Such a routine is presented below in pseudo-code format.

```
Do i=0 to Number of processes – 1
   if (rank= =i) then
      Do j=1 to nneighbours
         Send data of isendnodes(j,k) to ineighbours(j)
         //k represents all the sending nodes
      End Do
   Else
      if(i exists in ineighbours())then
         Receive data from i and store it to ireceivenodes(j,m)
         //m represents all the receiving nodes
      End if
   End if
End Do
```

While the above routine provides accurate means of transferring data between partitions, it leaves most of the processes idle most of the time, especially when large amounts of data is transferred (in cases of large partitions), as exchange takes place between a *sender* and a *receiver*. For that reason in this work transfer of data was performed with collective communication methods. These methods involve all the processes in the scope of a *communicator*, therefore a collective transfer routine must be executed by all the processes in a communicator. MPI provides a number of different routines that cover different types of collective operations such as *synchronization* (MPI_BARRIER), *data movement*, and *collective computation*. In the *Galatea-I* program, apart from the already discussed **MPI_BARRIER** routine that provides synchronization, collective communication routines were used only for data transfer. Of the available routines the ones used were MPI_BCAST and MPI_SCATTERV [Gro96]. Their syntaxes along with the syntax MPI_SCATTER that will help for the better understanding of MPI_SCATTERV are presented below:

- **MPI_BCAST(buffer, count, datatype, root, comm)**
- **MPI_SCATTER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)**
- **MPI_SCATTERV(sendbuf, sendcounts, displs, sendtype, recvbuf, recvcount, recvtype, root, comm)**

Routine **MPI_BCAST** must be executed by all processes consisted in the *communicator* **comm**. Data is broadcast from process with *rank* equal to **root** and received by all other processes. The

data sent are **count** number of entries of **datatype** type beginning at the entry with address **buffer**. Data is received by all the processes inside **comm** that their *rank* is not equal to **root**. The message consists of the same number of entries of **datatype** type and is stored at the memory address that starts with the entry **buffer**. Since the exact same command is executed by all processes, measures have to be taken in advance, so that the arrays or matrices in which the data will be stored will be adequately allocated, so that the *receive* operation will not "starve". The procedure of the *broadcast* operation is more easily described in Figure 4.5.

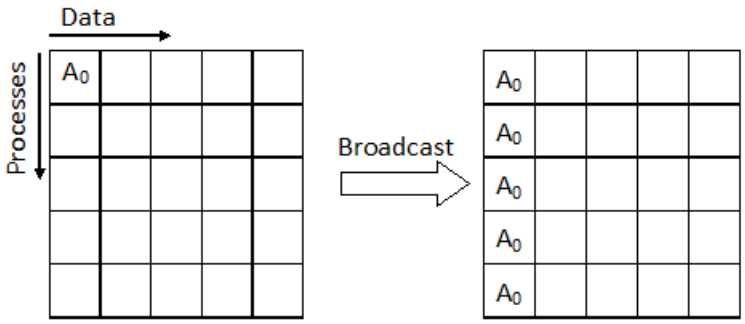


Figure 4.5 – Schematic representation of the Broadcast procedure.

With routine **MPI_SCATTER** data is distributed from process with *rank* equal to **root** to all the processes of the *communicator* **comm,** including **root**. The data are stored in the address space of root and amount to **sendcount**×**comm_size** number of elements of type **sendtype**, starting from the address **sendbuf**, where **comm_size** is the number of processes in the communicator **comm**. The message sent to each process, including **root** consists of **sendcount** number of elements of type **sendtype** and will be retrieved from **root**'s address space sequentially, according to the *rank* of each receiver. The received message will be stored to the address space of each process at memory space that begins at **recvbuf**, has a length of **recvcount**, and is of type **recvtype**. The whole procedure can be schematically described in Fugure 4.6.

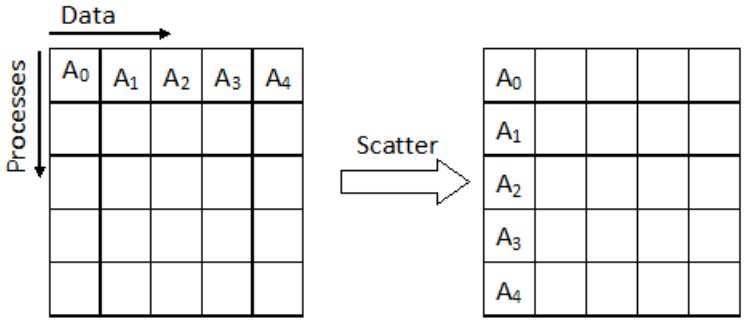


Figure 4.6 – Schematic representation of the Scatter procedure.

Routine **MPI_SCATTERV** works in the same way as **MPI_SCATTER**; data from the address space of **root** starting at **sendbuf** are distributed to all the processes within the *communicator* **comm**, including **root**. However, in this case, the count of data may vary, and also the location from where the data is taken from may vary, depending on the process it is sent to. This is accomplished by the arguments **sendcounts** and **displs**, respectively, which are arrays of size equal to the number of processes in the *communicator* **comm**. Therefore, data sent to process *i*, where *i* is the *rank* of the process within *communicator* **comm,** has size equal to **sendcounts(i)** and can be retrieved from the address space of **root** from a position starting at **sendbuf+displs(i)**. In this way, different amounts of data can be sent to different processes, thus making this function more attractive to the needs of the proposed algorithm. The received data have the same properties as with **MPI_SCATTER**, only this time it may have different size.

In the program *Galatea-I* exchange of data is mainly performed using the **MPI_SCATTERV** routine. For the implementation of this routine, and considering that it is always preferable to send large amounts of data at once, as it improves the algorithms efficiency, prior to the exchange a special array is constructed by each process. The array, named **scatterarray** contains all the data that should be sent to all adjacent partitions. The size of **scatterarray** depends on the type of the data sent (variable values, Green-Gauss gradients, velocity gradients), and the sum of *core* nodes sending data to all partitions, which is equal to the numbers in **nsendnodes()**. Therefore, the size of this array would be equal to the size of data that each core node would send (number of variables), multiplied by the sum of nodes sending data to each adjacent partition.

$$size(\boldsymbol{scatterarray}) = (size\ of\ data\ sent) \cdot \sum_{i=1}^{nneighbours} nsendnodes(i)$$

For example, if the data sent is the flow variables vector of each *core* node, the size of data would be equal to *5*, while if the velocity gradients were sent, the size of data would be equal to $3 \times 3 = 9$, as there are three velocity components and three derivatives for each component. The order of data in **scatterarray** should be ordered according to the ranks of all processes, so that the **displs** array would be simply defined by the **nsendnodes** and **ineighbours** arrays. Specifically, arrays **displs** and **sendcounts** would be constructed as follows:

```
Do i=1 to Number of processes
   exists =0
   Do j=1 to nneighbours
      if(i= =ineighbours(j)) then
         sendcounts(i-1) = (size of data sent)·nsendnodes(i)
         exists = 1
      End If
   End Do
   If(exists = = 0) then
      sendcounts(i-1) = 0
   End If
End Do

displs(0) = 0
```

```
Do i=2 to nneighbours
   displs(i-1) = idispls(i-1) + sendcounts(i-2)
End Do
```

In the above algorithm, adjustments to the arrays' indexes are made since due to the MPI implementation they have to be zero-based. The same applies for array **scatterarray** that should then have the structure presented in Figure 4.7:

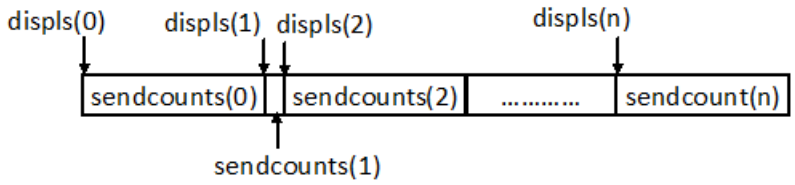


Figure 4.7 – Schematic representation of the scatterarray.

Finally, a routine for distributing information among all the adjacent sub-domains should only contain a loop over the processes, as each one would take the role of **root** and distribute the appropriate data to all the processes (including it). Such a routine is presented in pseudo-code form below:

```
All processes construct displs
All processes construct sendcounts
All processes construct scatterarray

Do i=1 to Number of processes
   i becomes root and distributes scatterarray to all the processes
Enddo
All processes use recvbuf to fill their needs
```

4.2 Agglomeration multigrid method

With the agglomeration multigrid method, a number of successively coarser meshes are generated from the initial mesh and relaxation of the partial differential equations is performed on all generated meshes in a specific pattern [Mav99] [Car00] [Lyg14b] [Lyg14c] [Lyg14d] [Lyg14e]. Its main idea derives from the fact that convergence of most iterative methods is slower on finer resolutions, as information is transferred only in one way, and this transfer must be repeated numerous times in an iterative solution for it to converge [Fer02] [Lyg14e]. In this way high and low frequency errors are generated, of which the solver is capable of eliminating the former, while it is inefficient against the latter. With the relaxation of the equations on successively coarser resolutions, these low frequency errors are transformed into high frequency ones, which the iterative method is capable to encounter [Fer02, Dar06, Lyg14c]. The obtained solution from the coarser grids is then merged with that from the finer ones to produce a more detailed resolution. In this way the convergence of the iterative algorithm is sufficiently improved.

4.2.1 Agglomeration methodology

The first concern for the implementation of the multigrid method is the generation of the successively coarser meshes [Mav97] [Mav98] [Car00] [Bla01] [Nis10] [Nis11] [Nis13] [Lyg14b] [Lyg14c] [Lyg14e] [Lyg15]. In this study the agglomeration procedure is used for that step, in which each coarser mesh is generated automatically inside the *Galatea-I* program, by merging neighboring nodes' control volumes and generating larger super-nodes. The procedure is performed right after the implementation of the domain decomposition methodology, so each sub-domain is considered an initial fine grid and successively coarser meshes are generated based on it. Control volume fusion is performed in a topology-preserving framework that resembles the advancing front technique, providing the means to be applied on purely tetrahedral or hybrid meshes, using isotropic or directional agglomeration, respectively.

The agglomeration procedure is guided through a set of general rules that are pre-defined in order to preserve the consistency of the solution between coarser and finer grids at external and internal boundaries [Lyg15]. These are summarized below:

- Each non-boundary node's control volume can only be fused with other adjacent non-boundary control volumes [Nis10] [Nis11] [Lyg15].
- Each boundary control volume can only be fused with its adjacent boundary nodes of the same type (e.g. solid wall, inlet, outlet, etc.) [Lyg15].
- A node that is located at the intersection of two or more surfaces with different boundary type is not agglomerated and remains a singleton throughout all the coarser meshes. An exception to the rule arises when in a two-boundary node one of the two boundaries is a symmetry one. In that case that control volume can be fused with other control volumes that share the same boundary types [Nis11] [Lyg15].
- Agglomeration is not performed on nodes that belong on two or more boundary surfaces that form slope discontinuities; agglomeration is permitted on a node that belongs on two surfaces that form a sharp edge with angle lower than 30^{o}.
- In order to ensure the validity of the data transfer between adjacent sub-domains, *ghost* nodes of each domain are not merged during the agglomeration procedure; they are fused subsequently, according to the merging of their corresponding *core* nodes at the adjacent mesh [Ven95] [Lyg15].
- For the simulation of turbulent flows, the maximum fusion of nodes to a *super-node* is limited to a number of eight, in order to preserve the topology of the initial mesh [Nis11] [Lyg15].

Based on these general rules, the isotropic and directional agglomeration can be defined.

With the isotropic agglomeration strategy [Lyg15] the nodes' control volumes are merged with no regard as to the type of elements to which the nodes belong. Agglomeration is initiated from the nodes of the solid wall boundaries (except those that were prohibited to merge in the aforementioned rules). The nodes of the solid wall boundaries that are eligible for agglomeration are marked as *seeds*; in case of internal boundaries due to domain decomposition, only the *core* nodes are marked as *seeds*.

Agglomeration is performed by looping over the *seed* nodes, by examining their eligibility for fusion with their adjacent ones. In case no limitation arises, the nodes' adjacent node control volumes merge with its own, and form a *super-node*. In case no adjacent node can be merged with the current node in loop, its adjacent super-nodes are examined, in order to include it. The loop is complete when all nodes of the *seed* list are either agglomerated or have remained singletons. A new list of *seeds* is then constructed from the nodes that the agglomeration front has touched and a new loop begins [Han02] [Lyg15]. The procedure ends when all nodes of the sub-domain (except *ghost* nodes) have been agglomerated or assigned as singletons. A schematic representation of the isotropic agglomeration procedure is presented in Figure 4.8 [Lyg15].

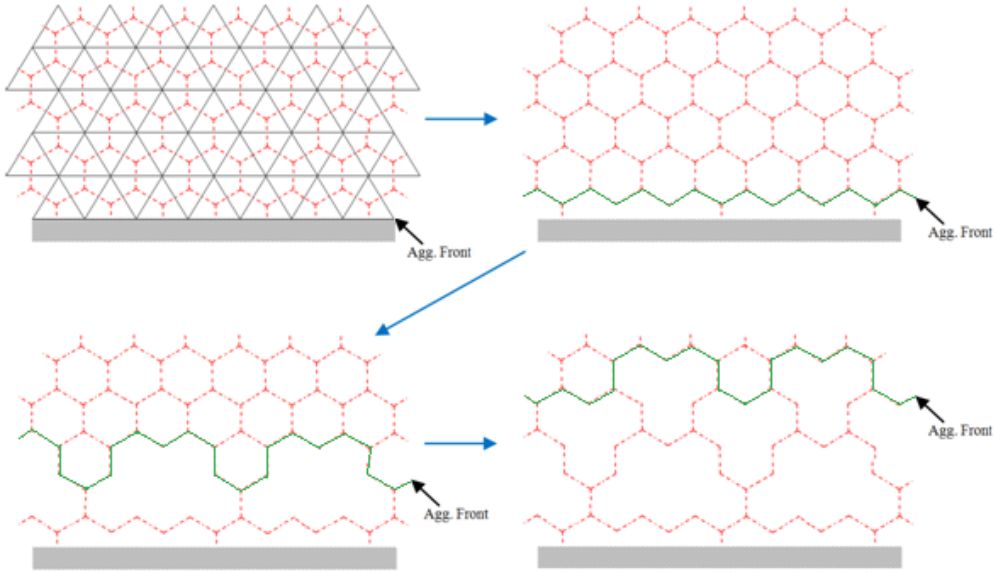


Figure 4.8 – Schematic representation of the isotropic agglomeration procedure.

When the basic agglomeration procedure has ended, *ghost* nodes are agglomerated or become singletons, depended on the behavior of their corresponding *core* nodes. With this procedure *ghost super-nodes* are generated, although their number of nesting control volumes may differ from that of their corresponding *core* nodes. However, that only happens so that there will be a one-to-one communication connection between *core super-nodes* and *ghost* ones; the correct amount of data will be transferred to each *ghost* node [Lyg15].

From the generated *super-nodes* the new *super-edges* are constructed, by deleting the internal edges in a *super* control volume and utilizing the external ones to calculate the normal vectors $\vec{n}_{PQ}$ at the interfaces between *super-node* control volumes. Each resulting normal vector is the sum of the vectors of the corresponding surfaces that belong to two neighboring super-nodes [Sor03] [Lyg15]. The same agglomeration procedure is repeated for the generation of the coarser grid, based on the newly generated one.

When the agglomeration procedure is completed, geometric data, such as the normal vectors at the boundary faces and *super* node control volumes have to be evaluated at each agglomeration level. These computations are performed with respect to the corresponding values at the merged nodes and edges. Additionally, communication data have to be established between the new *core* and *ghost* super-nodes at the neighboring agglomerated partitions [Lyg15].

In case of a hybrid mesh, for the simulation of viscous flow with boundary layers, the isotropic agglomeration scheme would not be preferable for the merging of node control volumes in the prismatic inflation area; full-coarsening directional agglomeration would be used instead [Car00] [Nis11] [Nis13] [Lyg15]. With this kind of agglomeration the nodes of each layer of the inflation region are characterized by an index number that groups nodes of different layers that are on the same column; these groups are also called *implicit lines* [Nis11] [Lyg15]. As with the isotropic agglomeration, a list of *seed* nodes is constructed, beginning with those that belong to the boundary region. The *seed* nodes are merged with eligible ones that reside on the same layer and do not share the same index, forming *super-nodes*.

Another list of *seed* nodes is constructed then, included nodes that were touched by the agglomeration front, and therefore belong to the next prismatic layer. These *seed* nodes are then allowed to merge with eligible nodes that reside on the same *implicit lines*, and therefore share the same indexes. This procedure is repeated until all the prismatic layer nodes have been merged and formed new *super-nodes*, or have remained singletons. The isotropic agglomeration procedure is then implemented to complete the process for the rest of the mesh. A schematic representation of the directional agglomeration method is presented in Figure 4.9 [Lyg15].

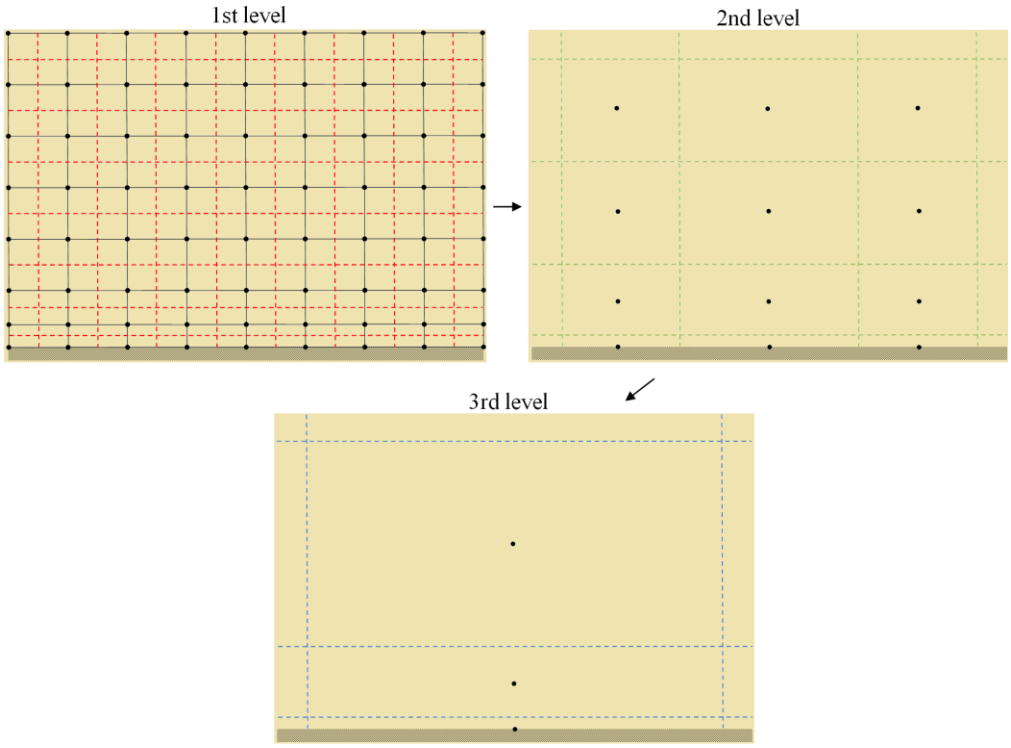


Figure 4.9 – Schematic representation of the full-coarsening directional agglomeration procedure.

4.2.2. Flux computation and numerical solution

The use of the agglomerated grids for the multigrid scheme is realized with the partial differential equations, with the implementation of the Full Approximation Scheme (FAS), according to which the PDEs are solved only at the finest grid, while an approximate version of those equations is relaxed at coarser levels [Fer02] [Sor03] [Lam04] [Nis10] [Nis11] [Nis13] [Lyg14b] [Lyg15]. So, in this case Equation 3.81, or Equation 3.84 (for unsteady flow simulations) is solved only at the finest grid, while an approximate version of those two is solved at the coarser grids. Flow variables are transferred between different agglomeration levels by restricting their values from finer to coarser grids and prolonging them from coarser to finer resolutions. This procedure is implemented in a form that resembles a V structure; PDEs are evaluated at the finer grid and their values are restricted towards the coarsest grid, then this procedure is repeated in reverse with the prolongation of the flow variables. A V cycle is defined as V(v1, v2), where v1 represents the number of relaxations before proceeding to a coarser mesh, while v2 denotes the number of relaxations after retrieving a solution from a coarser mesh [Lyg15].

Each V cycle begins with the approximation of the governing equations at the finest grid that provides updated values of the flow variables for each node *P*. These values are restricted to the next coarser grid by a smoothing equation, as follows [Mav97] [Car00] [Sor03] [Ni11] [Lyg14c] [Lyg15]

$$\vec{Q}_{P,restricted} = \left(I_Q\right)_h^H \vec{Q}_P = \frac{\sum \vec{Q}_P \cdot V_P}{V_P} \tag{4.1}$$

$$\vec{R}_{P,restricted} = (I_R)_h^H \vec{R}_P = \sum \vec{R}_P \tag{4.2}$$

where *H* denotes the finer mesh, while *h* represents the coarser one, and $\left(I_Q\right)_h^H$ and $(I_R)_h^H$ are restriction operators. After the restriction the relaxation of the PDE at the coarser mesh is performed by substituting the right hand side term of the equation with the following one [Mav97] [Car00] [Bla01] [Nis11] [Lyg14b] [Lyg15]

$$\vec{R}_{P,FAS} = \vec{R}_P\left(\vec{Q}_P\right) + [\vec{R}_{P,restricted} - \vec{R}_P\left(\vec{Q}_{P,restricted}\right)] \tag{4.3}$$

where $\vec{R}_P\left(\vec{Q}_P\right)$ is the flux balance of node *P* in the coarse grid. When restricting to an even coarser mesh, the same equations are used.

After half part of the V cycle is completed and the solution of the equations at the coarsest grid is obtained, the flow variables are transferred via prolongation to the finer grids. In case of inviscid flow a simple point injection scheme is used as follows [Car00] [Lyg15]

$$\Delta\vec{Q}_P = \left(I_Q\right)_H^h \Delta\vec{Q}_P = \Delta\vec{Q}_P = \vec{Q}_P - \vec{Q}_{P,restricted} \tag{4.4}$$

where $\left(I_Q\right)_H^h$ is a prolongation operator. In case of laminar or turbulent flows a distance-based procedure is used that considers the nodes of the same type (internal or boundary), as follows [Kat09] [Lyg15]

$$\Delta\vec{Q}_P = \left(I_Q\right)_H^h \Delta\vec{Q}_P = \frac{a_{p-P}\Delta\vec{Q}_p + \sum_{Q\in K_N(P)} a_{p-Q}\Delta\vec{Q}_Q}{a_{p-P} + \sum_{Q\in K_N(P)} a_{p-Q}} \qquad (4.5)$$

$$a_{p-Q} = \left|\vec{r}_{pQ}\right|^{-4}$$

where $\vec{r}_{pQ}$ is the vector that connects *super-node p* with the *super-node Q*, while $\Delta\vec{Q}_p$ and $\Delta\vec{Q}_Q$ represent the corrections of flow variables on these nodes. The updated flow variables of the nodes of the finer mesh are simply calculated as follows:

$$\vec{Q}_P^H = \vec{Q}_P^h + \Delta\vec{Q}_P \qquad (4.6)$$

In this study the FAS methodology is integrated with the Full Multigrid (FMG) scheme [Fer02] [Lyg14b] [Lyg15], in order to gain from the advantages that the two have and increase the obtained acceleration. With the FMG scheme, the relaxation of PDEs begins at the coarsest mesh, without completing the V cycle to the finest grid. Instead, the magnitude of the cycle increases progressively, until the finest mesh is included in the procedure [Lyg15]. In this way an initial guess of the equations' solution is achieved rather early, which is then fed to the finer grid and the FAS V cycle is continued for the rest of the simulation.

CHAPTER 5

NUMERICAL RESULTS

5.1 Steady-state numerical solutions

5.1.1 Inviscid flow over a rectangular wing with a NACA0012 airfoil

The first test case considers the inviscid flow over a rectangular wing with a NACA0012 airfoil [Lyg14b] [Nee97], with zero angle of attack, while the free-stream velocity is equal to unity, in order to be correctly utilized by the dimensionless equations in the developed solver. The computational mesh consisted of *625,250* nodes and *3,500,243* tetrahedrons, while the simulation was performed on a DELL T7500 workstation with two Intel(R) Xeon(R)-X5660 six-core processors at 2.80 GHz. The computational grid around the rectangular wing with NACA0012 airfoil is presented in Figures 5.1 and 5.2. The evaluation of the incompressible field was performed with an artificial compressibility parameter value equal to *10.0*, while the steady-state solution was achieved with the explicit four-stage Runge-Kutta scheme, using a Courant–Friedrichs–Lewy (CFL) number equal to *0.5*. A summary of the simulation parameters is presented in Table 5.1.

Table 5.1 – Simulation parameters for the inviscid flow over a rectangular wing with a NACA0012 airfoil case.

Parameters	
Type of flow	Inviscid
Reynolds number	-
Angle of attack	0^o
Grid density	625,250 nodes 3,500,243 tetrahedrons
Artificial compressibility parameter β.	10.0
Number of partitions	4
Number of agglomerations	3
CFL	0.5
Computer system	DELL T7500 workstation with two Intel(R) Xeon(R)-X5660 six-core processors at 2.80 GHz

For the qualitative evaluation of this case, the dimensionless pressure contours at the mid-span of the wing are presented in Figure 5.3, while Figure 5.4 illustrates the extracted chordwise pressure coefficient distribution, compared to corresponding numerical results found in the literature [Nee97], demonstrating good agreement between the two.

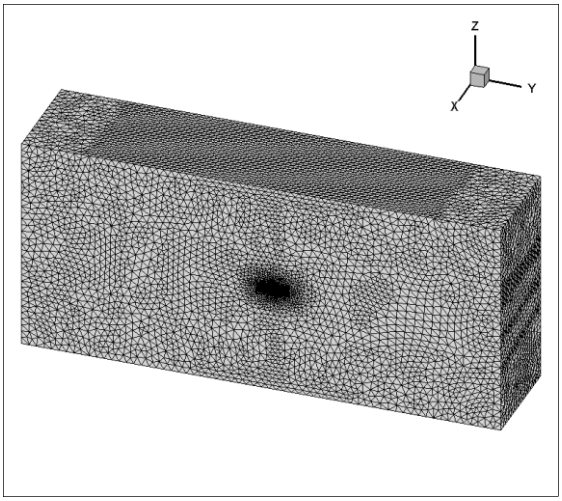


Figure 5.1 – Far View of the computational domain around the rectangular wing with NACA0012 airfoil

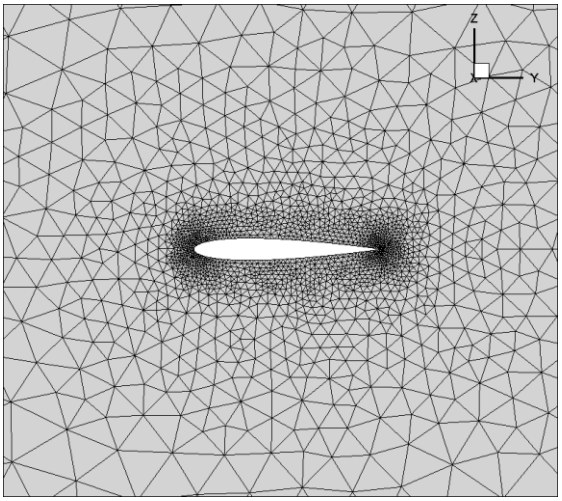


Figure 5.2 – Close view of the computational domain around the rectangular wing with NACA0012 airfoil.

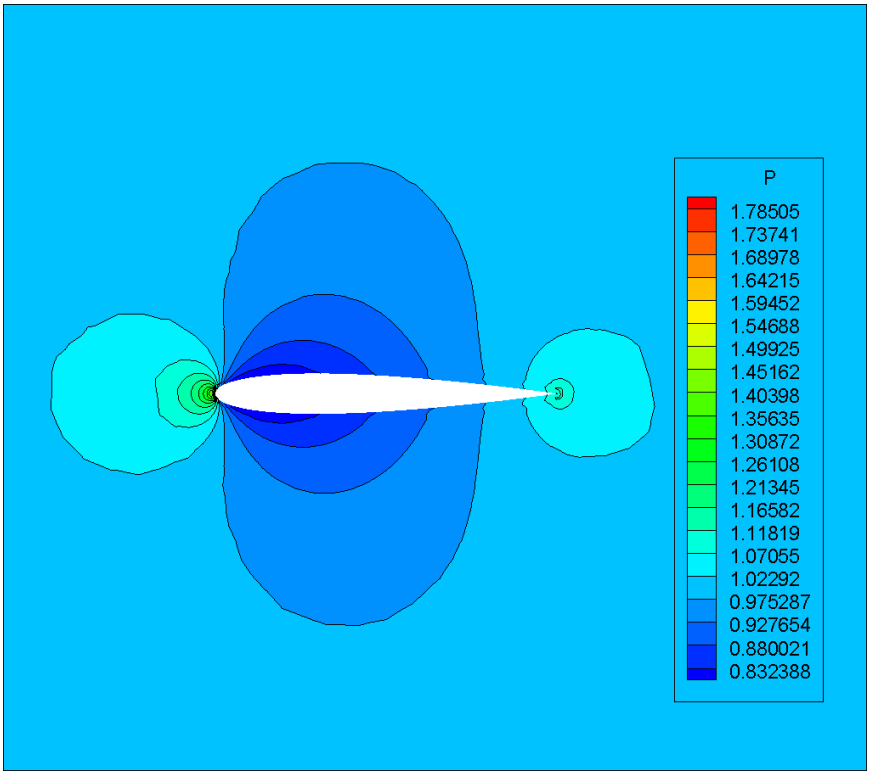


Figure 5.3 – Dimensionless pressure contours at mid-span of the rectangular wing with NACA0012 airfoil.

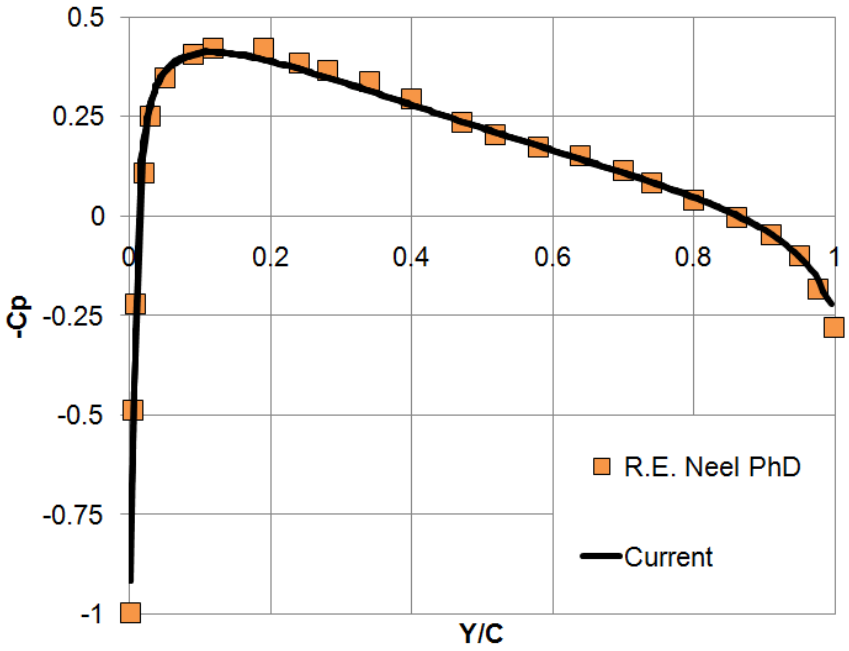


Figure 5.4 – Pressure coefficient distribution along the mid-span of the wing.

5.1.2 Three-dimensional lid-driven cavity flow

The second test case studied in this work concerns the three-dimensional lid-driven laminar viscous flow inside a cubic cavity. The geometry, as illustrated in Figure 5.5 consists of a cube with edge size equal to unity. All of the cube walls, except one, are considered as solid walls where no-slip conditions are imposed, while the last wall, which is arbitrarily selected as the upper surface of the cube, is considered as "inflow" where fluid moves with a velocity equal to unity; $|\vec{V}| = (u,\ v,\ w) = (1,\ 0,\ 0)$.

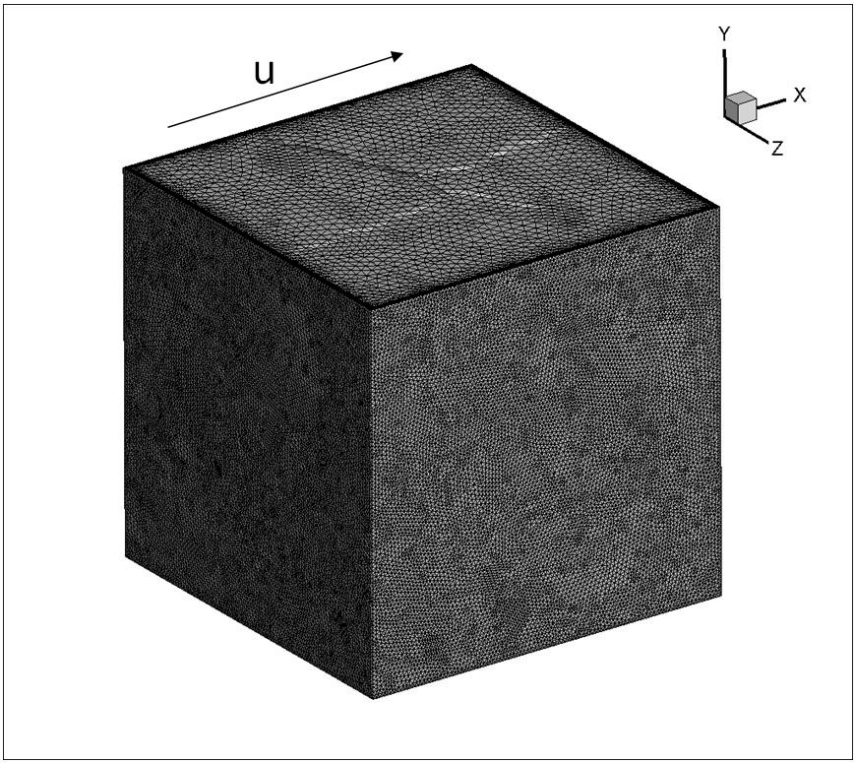


Figure 5.5 – Geometry and utilized grid for the lid-driven cavity flow case.

The lid-driven cavity flow problem is considered a classic recirculation problem, and may idealize many environmental, geophysical and industrial flows, while extensive work on this specific case can be found in [Kos84] [Jia94] [Yan98] [Mon01] [Sheu02] [Tai05] [Vra12]. The computational grid used in this work consists of *768,628* nodes, *1,280,935* tetrahedrons, while 10 layers of *1,050,140* prisms in total have been applied at the five walls of the cube where the no-slip conditions were imposed. Simulation was performed on a DELL T7500 workstation with two Intel® Xeon®-X5650 four-core processors at *2.67* GHz, while for acceleration of the procedure by parallel processing, the initial grid was divided into four sub-domains, with the METIS algorithm.

Many researchers have found that flows of this kind with Reynolds numbers ranging lower than 2000 (*Re<2000*) can result to steady-state solutions and no Taylor-Görtler-like (TGL) vortices [Tai05] appear. In this work, two simulations were performed with aim to reach at a steady-state solution; one with a Reynolds number equal to 400 (*Re=400*) and one at *Re=1000*, calculated with respect to the cube's edge length. The artificial compressibility parameter β was selected equal to *10.0*, while the steady-state solution of both described flows was achieved with the explicit four-stage Runge-Kutta scheme using a CFL number equal to *0.5*. The no-slip boundary

conditions in this case are imposed as it is described in Chapter 3, however due to the need to maintain constant speed of unity at the inlet, Dirichlet boundary conditions were enforced at the upper surface of the cubical cavity. A summary of the solution parameters is presented in Table 5.2.

Table 5.2 – Simulation parameters for the lid-driven laminar viscous flow inside a cubical cavity case.

Parameters	
Type of flow	Viscous – Laminar
Reynolds number	400, 1000
Angle of attack	–
Grid density	768,628 nodes 1,280,935 tetrahedrons 1,050,140 prisms
Artificial compressibility parameter *β*.	10.0
Number of partitions	4
Number of agglomerations	3
CFL	0.5
Computer system	DELL T7500 workstation with two Intel$^{(R)}$ Xeon$^{(R)}$-X5650 four-core processors at 2.67 GHz

Figure 5.6 illustrates the velocity profiles of component u on the vertical centerline of the *x-z* plane at *y=0.5*, along with the profiles of the v component of velocity on the horizontal centerline of the same plane at *Re=400*. The respective results for the case of *Re=1000* are presented in Figure 5.7. Both results are compared to corresponding numerical solutions by Tai and Zhao [Tai05]; sufficient agreement can be reported, demonstrating that the proposed code is capable of simulating accurately such complex flows.

In figure 5.8 the velocity streamlines on the *x-y* plane at *x=0.5*, along with the contours of the dimensionless *u* velocity component at *Re=400* (left) as well as *Re=1000* (right) is presented. The lower and two side edges of the plane are considered solid walls, where velocity has a value equal to zero, while at the upper edge fluid moves with a velocity equal to unity towards the positive direction of the *x*-axis. The flow at this plane is characterized by a primary vortex near the middle, and two smaller ones being created at the bottom of the cavity. It can be observed that the primary vortex moves towards the center of the cavity as the Reynolds number increases and two smaller vortices appear in the two lower corners, with the leftmost one being more intense in the case of *Re=1000*. In Figure 5.9 similar velocity streamlines along with the *u* velocity component contours are presented at *y=0.5* of the *x-z* plane. In both cases two symmetrical contra-rotating vortices can be observed towards the negative direction of the *x* axis. In the case of *Re=1000* a singularity point can be observed on the x-z plane near the center of the cubical cavity. This point is a discontinuity where the flow is moving vertically to the *x-z* plane and is also visible in the case of *Re=1000*. Finally, in Figure 5.10, similar streamlines and contours are sketched at x=0.5 of the *y-z* plane. Here, two pairs of mirroring vortices appear near the four corners of the cube. Each pair consists of a primary and a secondary vortex. The phenomenon intensifies with the increase of the Reynolds number and the vortices become more distinct.

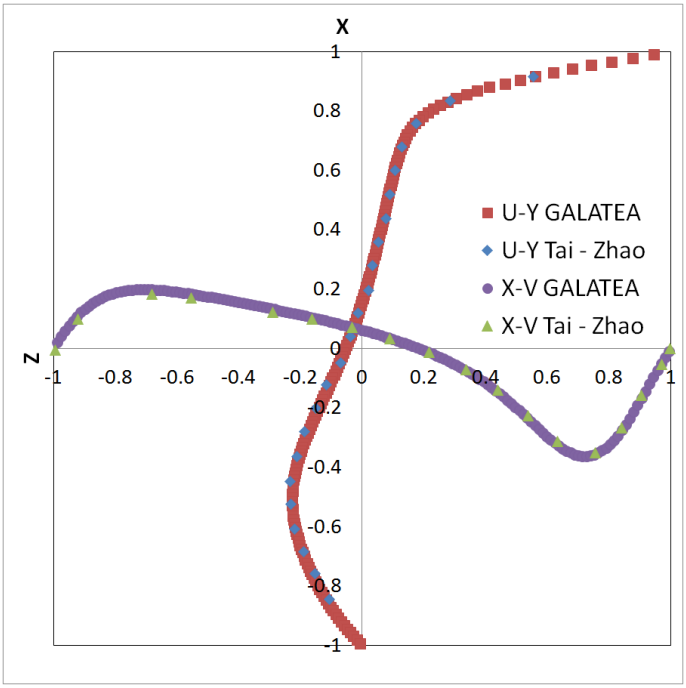


Figure 5.6 – Distributions of *u* and *v* velocity components along the *x* and *y* centerlines respectively, of the *x*-*z* plane at *z*=0.5 (*Re=400*).

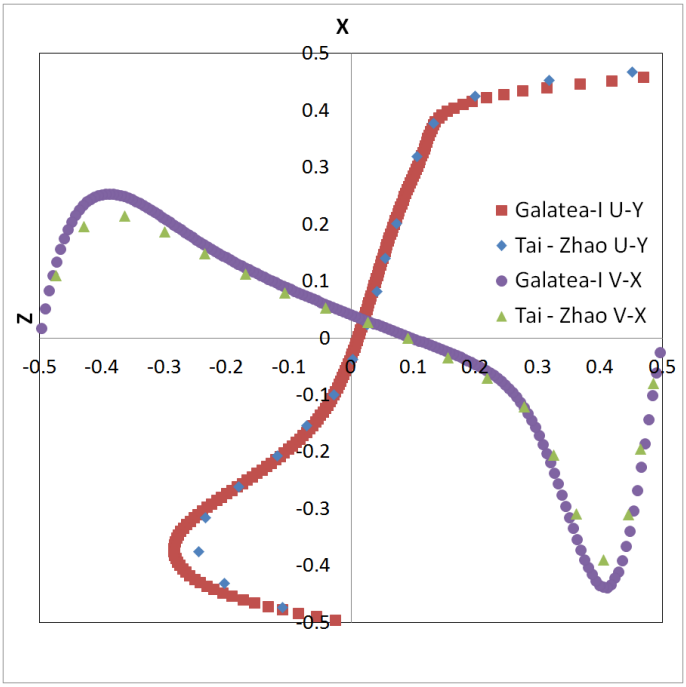


Figure 5.7 – Distributions of *u* and *v* velocity components along the *x* and *y* centerlines respectively, of the *x*-*z* plane at *z*=0.5 (*Re=1000*).

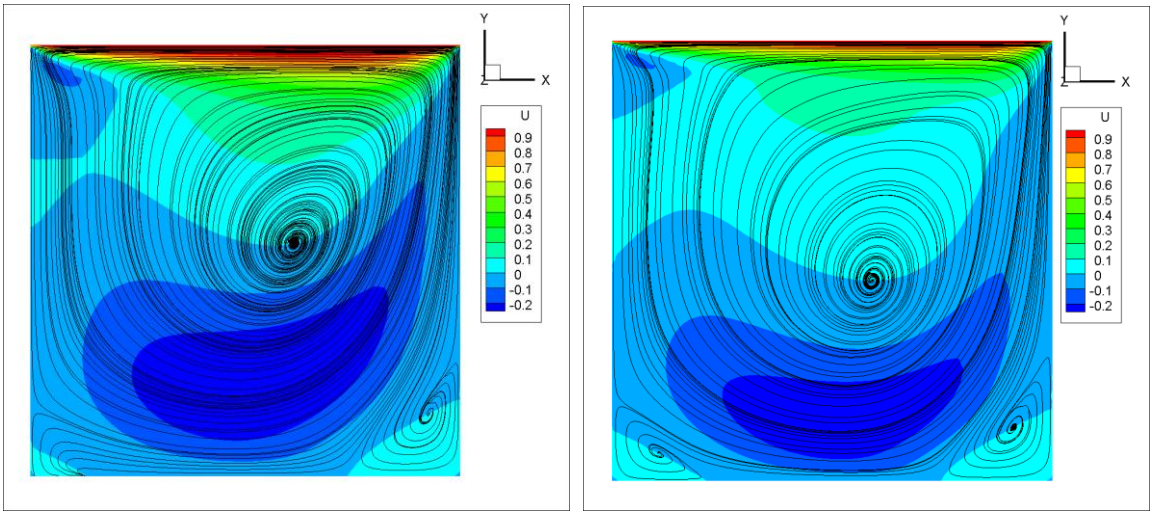


Figure 5.8 – Velocity traces and *u* velocity contours on *x-y* plane at *z=0.5* with *Re=400* (left) and *Re=1000* (right).

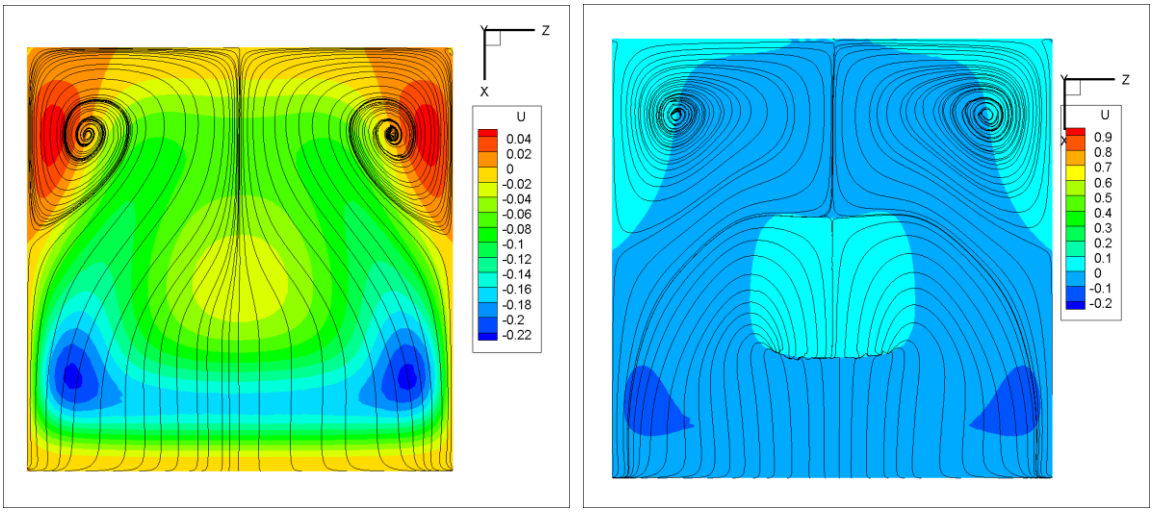


Figure 5.9 – Velocity traces and *u* velocity contours on *x-z* plane at *y=0.5* with *Re=400* (left) and *Re=1000* (right).

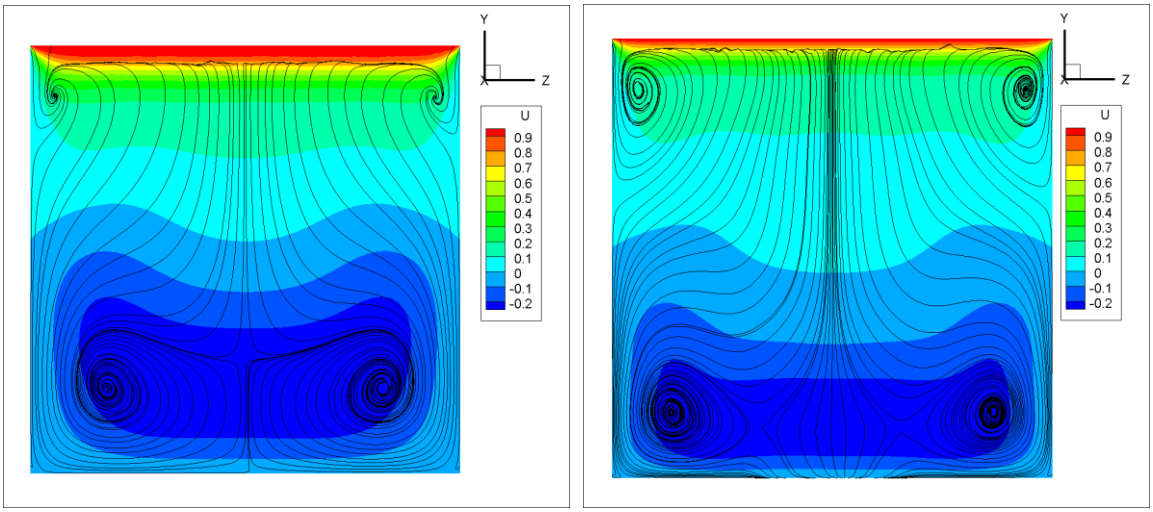


Figure 5.10 – Velocity traces and *u* velocity contours on *y-z* plane at *x=0.5* with *Re=400* (left) and *Re=1000* (right).

5.1.3 Steady viscous flow around a circular cylinder

The quasi-3D laminar flows over a circular cylindrical surface are popular test cases for incompressible flow solvers [Cou77] [Fra90] [Tai03] [Vra12], as they tend to produce unsteady results in most cases. Steady-state can be achieved only at very low Reynolds numbers, around the value of 40, calculated with respect to the cylinder's diameter. The computational grid, presented in Figure 5.11, consists of *474,540* nodes, *1,222,663* tetrahedrons and *494,130* prisms, while parallel computation was performed by dividing it into eight partitions. Simulation was performed on a workstation with an AMD FX™ 8150 eight-core processor at *3.62 GHz*. Further acceleration was achieved with the multigrid scheme, for which three coarser grids were generated with the directional agglomeration scheme.

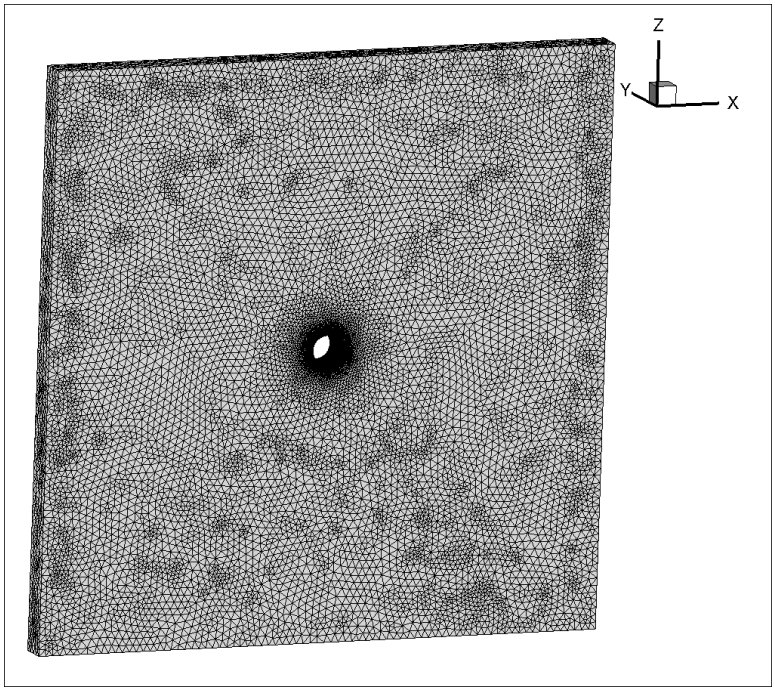


Figure 5.11 – Utilized grid density for the steady viscous flow over a circular cylinder.

As it was mentioned before, steady state solution in this case can be achieved at a very low Reynolds number, therefore simulation was performed at *Re=40*. The artificial compressibility parameter was set equal to *10.0*, while time integration was performed with the explicit Runge-Kutta scheme with a CFL number of *0.5*. A summary of the simulation parameters is presented in Table 5.3.

In Figure 5.12 the velocity streamlines at the mid-span of the cylinder is presented, along with the non-dimensional pressure contours. The two fully developed symmetrical vortices at the wake of the cylinder can be observed, while the contours of the *u* velocity component are presented in Figure 5.13. Assessment of this case can be performed by evaluating certain geometric parameters of those two vortices [Cou77], which can be easily measured. Figure 5.14 illustrates the basic geometric characteristics of the "twin vortices"; namely, L_s is the length of the "standing eddies", measured from the rear end of the cylindrical surface up to the end of the observed recirculation, α represents the horizontal length measured from the rear end of the cylinder up to

the vortices' center, b denotes the distance between the two vortices centers and θ_s is the separation angle.

Table 5.3 – Simulation parameters for the steady viscous flow around a circular cylinder.

Parameters	
Type of flow	Steady viscous
Reynolds number	40
Angle of attack	–
Grid density	474,540 nodes 1,222,663 tetrahedrons 494,130 prisms
Artificial compressibility parameter β.	10.0
Number of partitions	8
Number of agglomerations	3
CFL	0.5
Computer system	workstation with an AMD FX™ 8150 eight-core processor at *3.62 GHz*

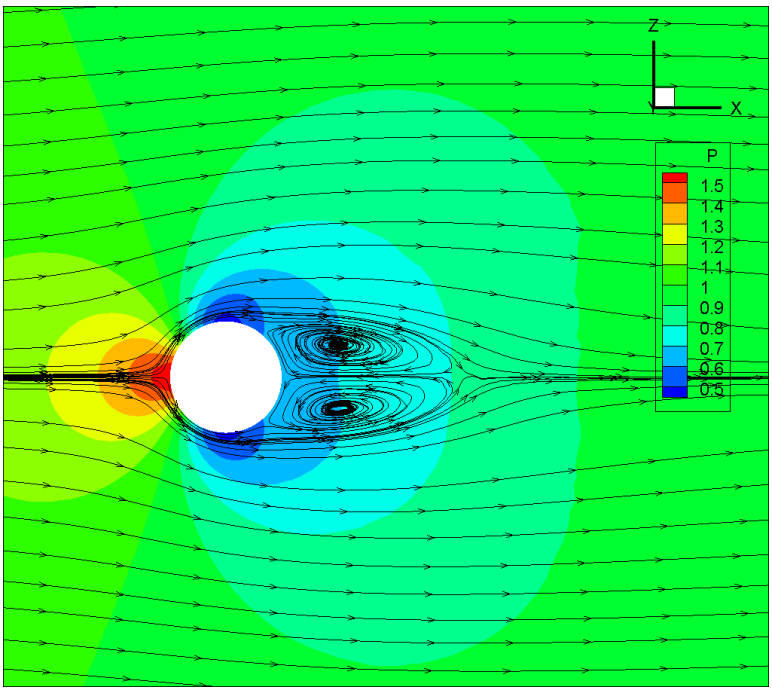


Figure 5.12 – Velocity streamlines at the cylinder's mid-span, and dimensionless pressure contours.

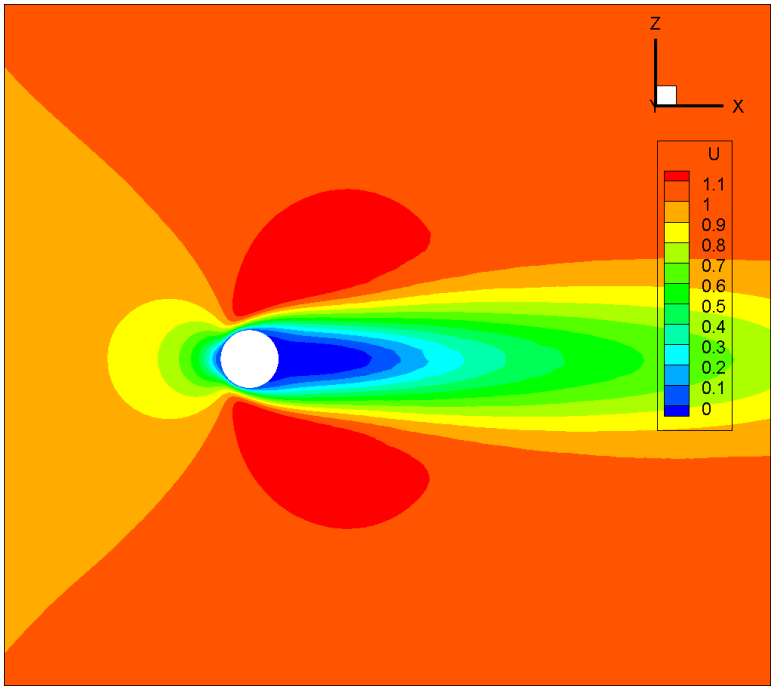


Figure 5.13 – Contours of the *u* velocity component at the mid-span of the cylindrical surface.

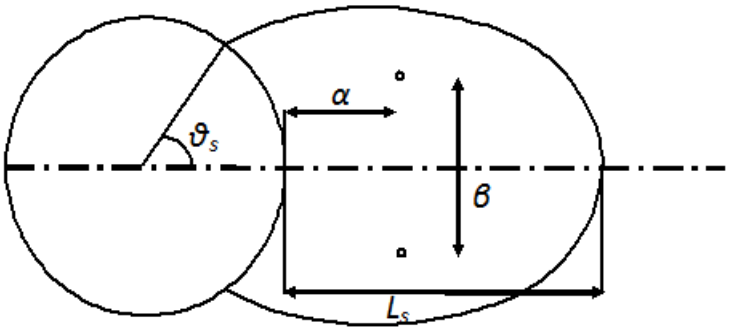


Figure 5.14 – Geometric characteristics of the "mirror vortices" appearing at the wake of the cylindrical surface.

In table 5.4 the geometric parameters of the symmetrical vortices at the wake of the cylindrical surface, obtained with *Galatea-I*, are presented in comparison with experimental and numerical ones found in the literature. The results for the separation length L_s are closer to the experimental ones by Coutanceuau et al. [Cou77], while good agreement with all reference results can be reported for the separation angle θ_s. Sufficient agreement can also be reported for the parameter *b*, while a discrepancy is observed for the length *a*. This can be attributed to the mesh density at the wake of the cylinder. It is believed that finer mesh can provide better results.

Table 5.4 – Simulation parameters for the steady viscous flow around a circular cylinder.

	Separation length *Ls*	Separation angle θ_s	*a*	*b*
Present solver	2.16D	53.2 ^{o}	0.51	0.56
Coutanceau et al. [Cou77] (experimental)	2.13D	53.0^o	0.76	0.59
Franke et al. [Fra90] (numerical)	2.36D	53.8^o	-	-
Vrahliotis et al. [Vra12] (numerical)	2.24D	53.0^o	0.71	0.59

Finally, the convergence history of the four dimensionless parameters is presented in Figure 5.15. The equations were converged up to a residual of *1.0E-9*, requiring a little over than *30,000* iterations.

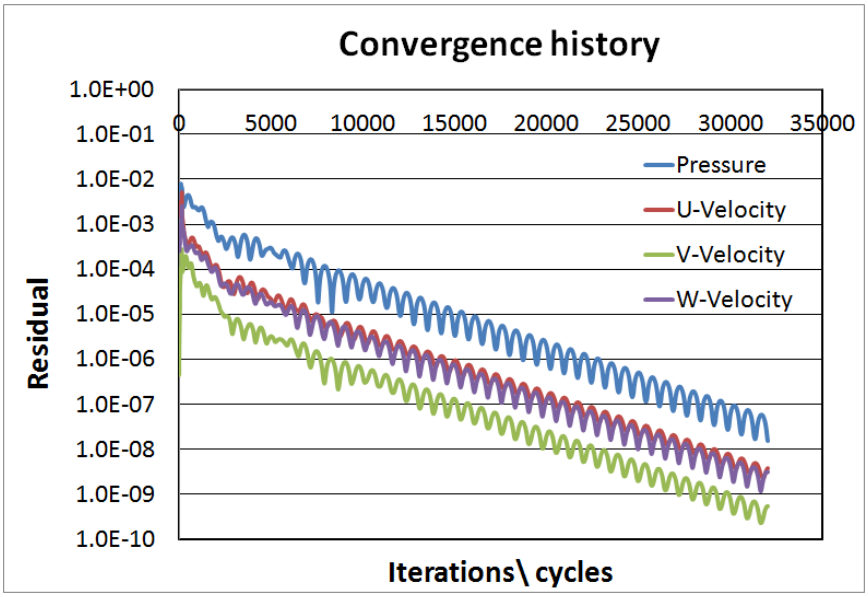


Figure 5.15 – Convergence history of the four flow parameters for the steady flow around a circular cylinder.

5.1.4 Steady viscous laminar flow around a sphere.

Viscous flow around a sphere at low Reynolds numbers is a simple test case with a theoretical solution concerning the drag force, as it was presented by Stokes in 1851 [Sto51]. However, significant interest is found in the simulation of the wake behind a sphere. Flow separation at the rear of a sphere begins at a Reynolds number equal to *24* [Tan56] [Joh99], resulting in an axisymmetric vortex ring. The flow is steady up to a Reynolds number in the region *210<Re<270*, where the vortex is transformed into a double-threaded wake [Mag61] [Joh99], steady in geometry, vortex shedding in nature.

In this test case the Reynolds number is equal to *100*, computed with respect to the sphere's diameter. The computational mesh consists of *265,492* nodes, *236,362* tetrahedrons and *444,288* prisms, as 32 layers of prisms were generated over the sphere surface, with the first layer distance

from the surface mesh equal to *0.025D*, where *D* is the sphere's diameter. The flow direction is along the *y* axis, with the artificial compressibility parameter being equal to *10.0* and the CFL number equal to *0.5*. The computational mesh is illustrated in Figure 5.16, while a summary of the test case configuration is presented in Table 5.5.

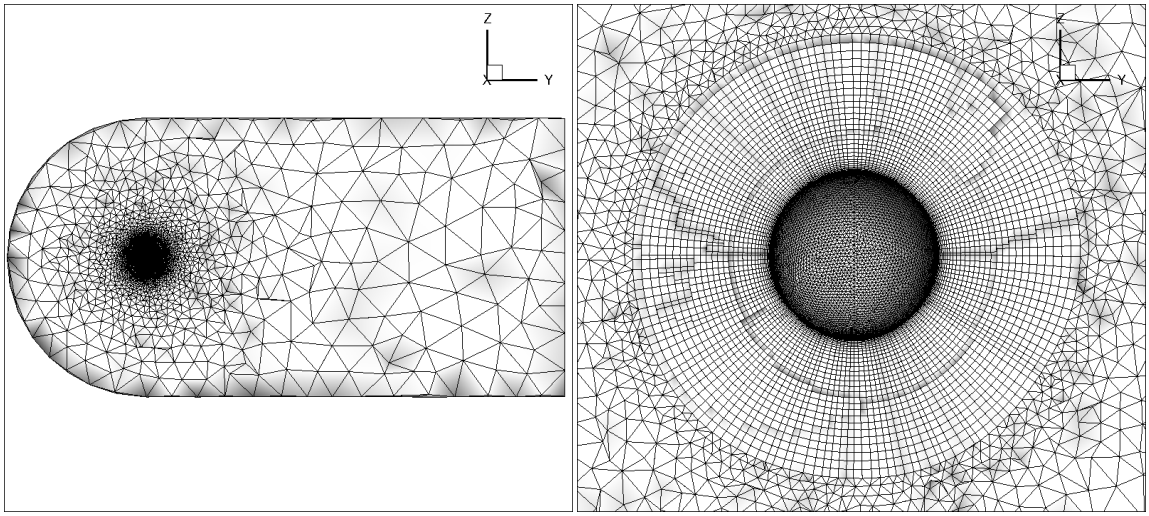


Figure 5.16 – Utilized mesh for the viscous laminar flow around a sphere.

Table 5.5 – Simulation parameters for the steady viscous flow around a sphere.

Parameters	
Type of flow	Steady viscous
Reynolds number	100
Angle of attack	–
Grid density	265,492 nodes 236,362 tetrahedrons 444,288 prisms (32 layers)
Artificial compressibility parameter *β*.	10.0
Number of partitions	4
Number of agglomerations	2
CFL	0.5
Computer system	workstation with an AMD FX™ 8150 eight-core processor at *3.62 GHz*

In Figure 5.17 the pressure contours around the sphere along with the velocity streamlines is presented. The flow separation is obvious, while a pair of symmetrical vortices can be seen at the wake of the sphere. Actually, what seems as a pair of vortices is in fact a single axisymmetric vortex ring, a slice of which is visible in the figure. This is more obvious in Figure 5.18 where the velocity streamlines have been sketched in a three-dimensional form and the axisymmetric vortex is clearly defined. The geometric characteristics of the wake region of the sphere have been documented by various researchers. The corresponding results of the present solver, compared to those reported by various researchers, are presented in Table 5.6.

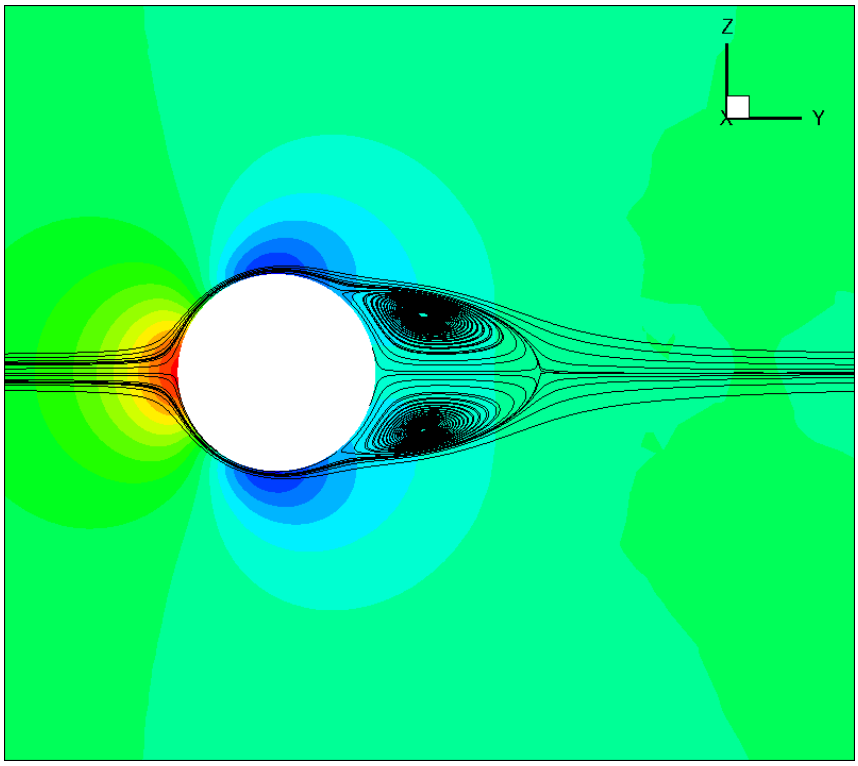


Figure 5.17 – Pressure contours along with velocity streamlines around a sphere at Re=100.

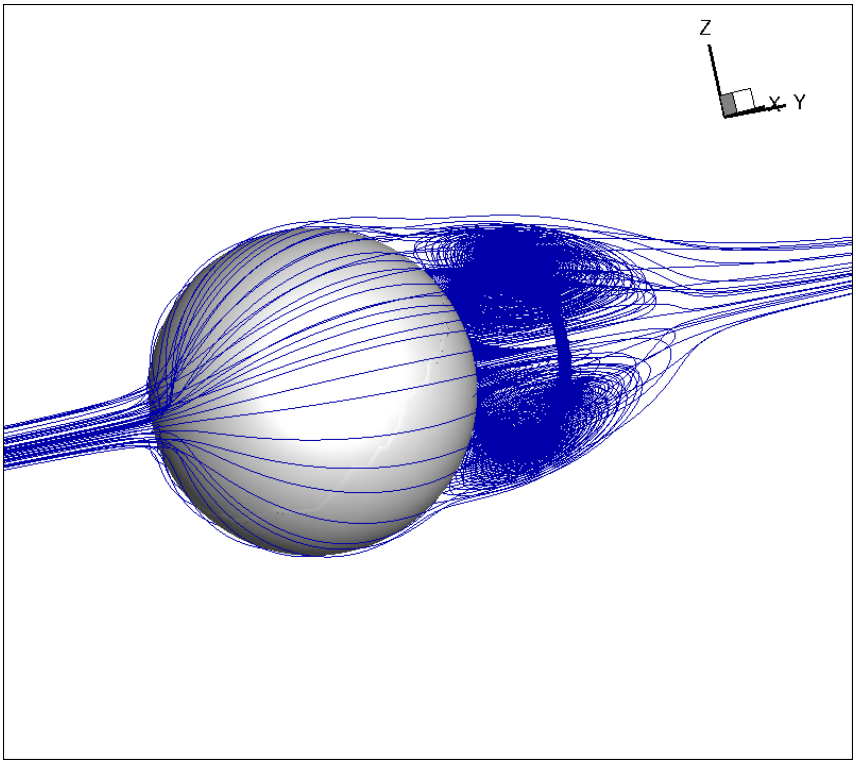


Figure 5.18 – Velocity streamlines at the wake of a sphere.

Table 5.6 – Characteristics of viscous laminar flow around a sphere.

	Separation length *Ls*	Drag coefficient C_d
Present solver	1.34*D*	1.098
Taneda [Tan56] (experiment)	1.39*D*	-
Lee [Lee00] (numerical)	1.34*D*	1.087
Le Clair [LeC70] (numerical)	-	1.096
Wang et al.[Wan08] (numerical)	1.36D	1.108

It is clear from the results on Table 5.5 that the present solver is capable of simulating accurately such flows as the one presented in this case. The length of the vortex at the wake of the sphere is clearly well predicted, as well as the drag coefficient on the sphere. In Figure 5.19 the pressure coefficient on the surface of the sphere is presented. The results show very good agreement to the numerical results by Lee [Lee00] and Le Clair [LeC70]. Additionally, the pressure coefficient at the centerline of the wake region is presented in Figure 5.20 and is compared to the results from Lee [Lee00]. Aside from minor discrepancies at approximately x/D=3, a sufficient accuracy is observed for the results of the present solver.

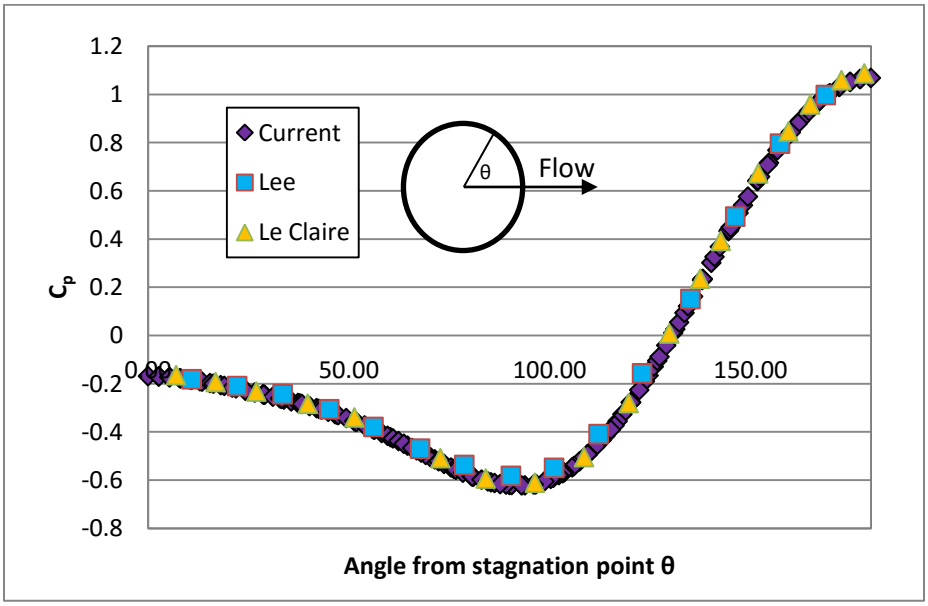


Figure 5.19 – Pressure distribution on the surface of the sphere at Re=100.

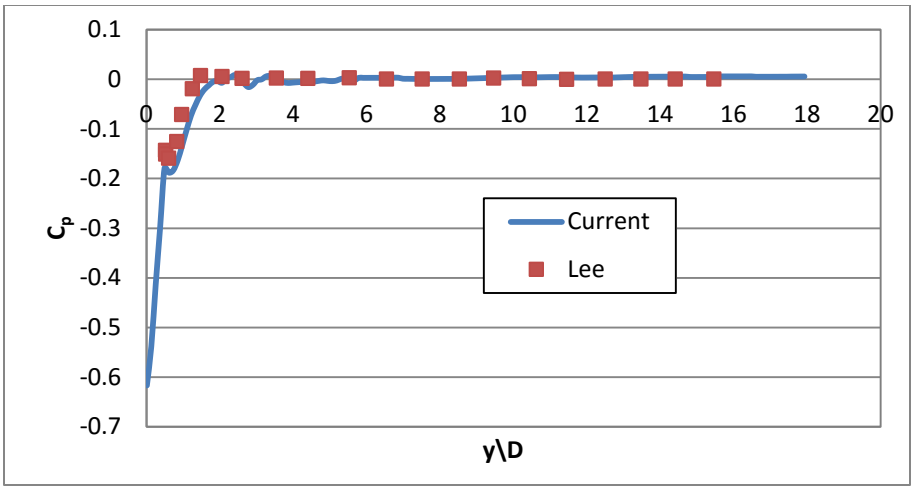


Figure 5.20 – Pressure distribution along the centerline of the wake behind the sphere, at Re=100.

5.1.5 Turbulent flow over a rectangular wing with a NACA0012 airfoil.

The first test case to introduce the turbulence equations is the quasi-3D problem of turbulent viscous flow over a rectangular wing with a NACA0012 airfoil. This specific case proved to be a challenge, due to the dense mesh that was required to be generated in order to simulate with accuracy the thin boundary layer. The computational mesh that was finally generated, consisted of *4,823,633* nodes, *5,182,381* tetrahedrons and *7,687,320* prisms, while it was divided in sixteen partitions for parallelized computation on a Dell™ R815 Poweredge server with four AMD Opteron™ 6380 sixteen-core processors at 2.50GHz. The computational mesh is presented in Figure 5.21. The solution was accelerated employing four coarser meshes for the agglomeration multigrid method. The Reynolds number was set equal to *3.0E+6*, the artificial compressibility parameter was set equal to *10.0* and the CFL number to *0.5*. A summary of the simulation parameters is presented in Table 5.6.

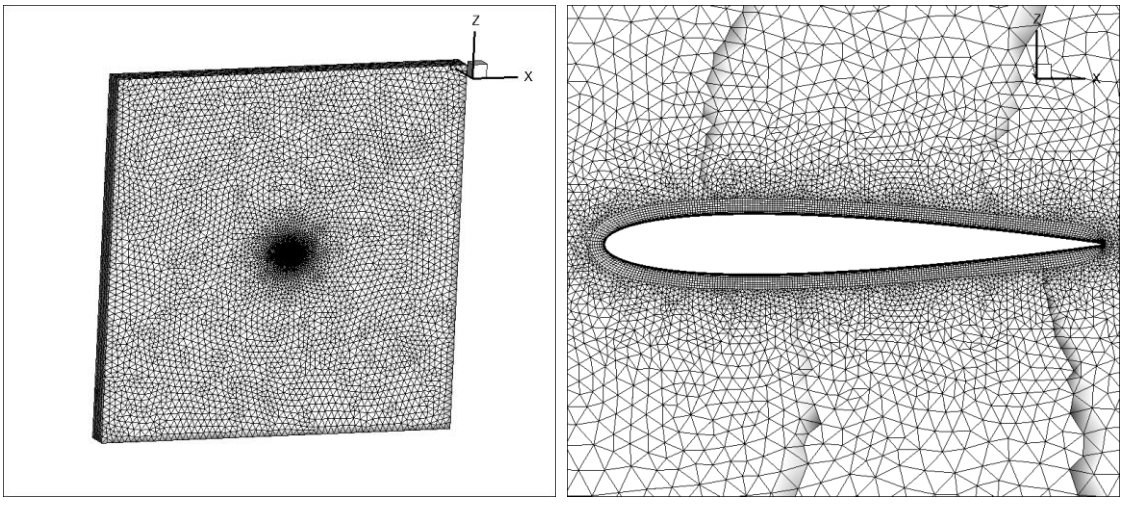


Figure 5.21 – Utilized mesh for the viscous turbulent flow around a rectangular wing with NACA0012 airfoil.

Table 5.7 – Simulation parameters for the steady viscous turbulent flow around a NACA0012 rectangular wing.

Parameters	
Type of flow	Steady viscous turbulent
Reynolds number	3.0E+6
Angle of attack	0
Grid density	*4,823,633* nodes *5,182,381* tetrahedrons *7,687,320* prisms
Artificial compressibility parameter β.	10.0
Number of partitions	16
Number of agglomerations	4
CFL	0.5
Computer system	Dell™ R815 Poweredge server with four AMD Opteron™ 6380 sixteen-core processors at 2.50GHz

The simulation in this case was evaluated against the experimental data by Gregory and O'Reilly [Greg70]. In Figure 5.22 the pressure coefficient distribution along the mid-span of the NACA0012 rectangular wing is presented. It is clear that the results from the proposed solver agree very well with the reference experimental data. In Figures 5.23 and 5.24 the contours of pressure, streamwise velocity and turbulent kinematic viscosity, respectively, are illustrated.

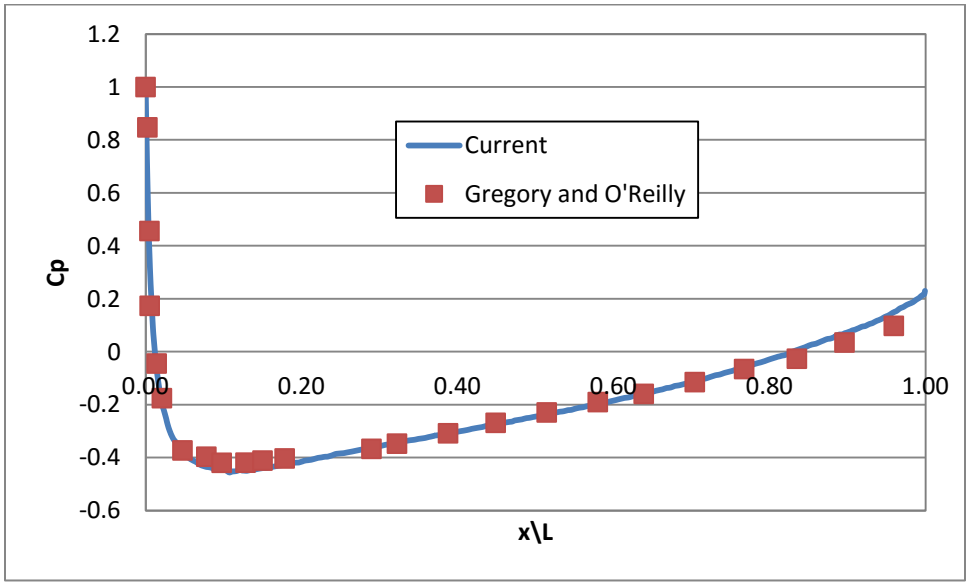


Figure 5.22 – Pressure distribution along the mid-plane of the NACA0012 rectangular wing.

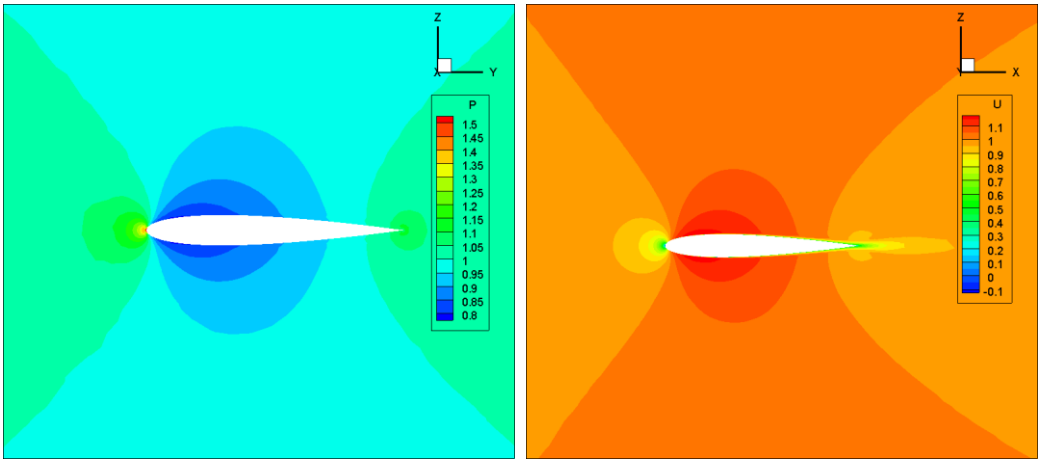


Figure 5.23 – Pressure contours (left) and velocity contours (right) at the mid-span of the NACA0012 rectangular wing.

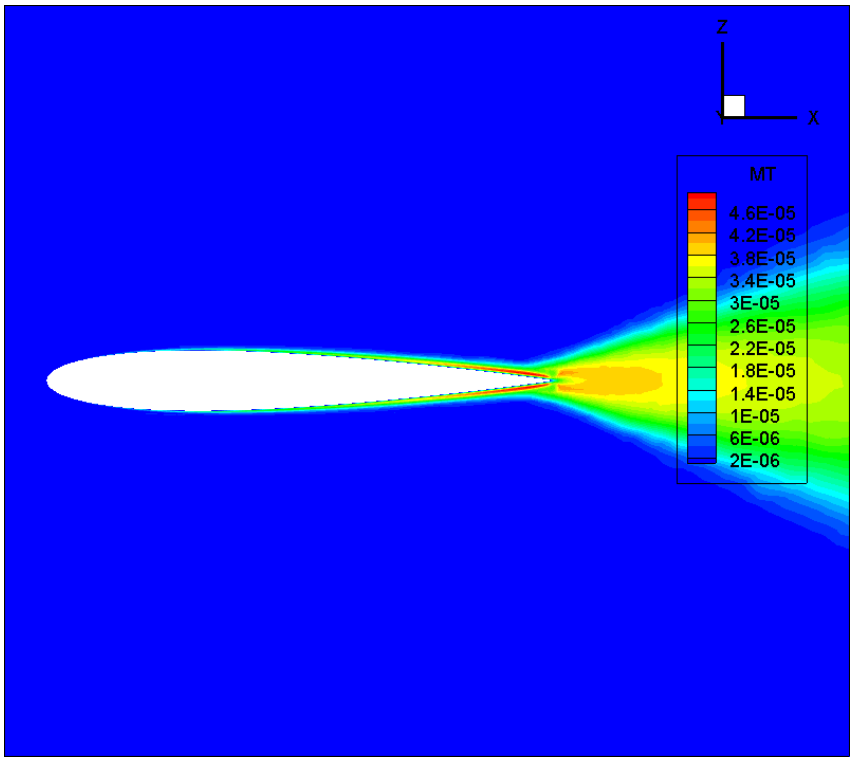


Figure 5.24 – Turbulent kinematic viscosity contours at the mid-span of the NACA0012 rectangular wing.

5.1.6 Steady turbulent flow around an axisymmetric submarine hull at 0^o, 18^o and 30^o angle of attack

The following benchmark test case is the first part of a family of test cases concerning the (pseudo-) steady turbulent flow around the surfaces of a submarine model. The model, named SUBOFF, was designed by the David Taylor Research Center (DTRC) [Gro89] [Hua89] [Gor90] [Liu98] on behalf of the Defense Advanced Research Projects Agency (DARPA). It was designed as an axisymmetric hull with appendable parts, such as a fairwater configuration, stern appendages and two different stern ring wings. Experiments in a wing tunnel as well as a towing tank were performed at the DTRC on different configurations of the SUBOFF model and on different attack angles [Hua89], and the obtained experimental flow results were cross-validated with corresponding numerical ones [Gor90].

In this study the flow around the axisymmetric hull of the submarine model has been simulated in different Reynolds numbers and angles of attack, while the flow around the submarine hull with a fairwater configuration has also been simulated at 0^o angle of attack and 0^o angle of yaw. For the generation of the geometric entities of the SUBOFF model, appropriate Fortran codes have been developed, based on existing ones published openly by the DTRC [Gro89].

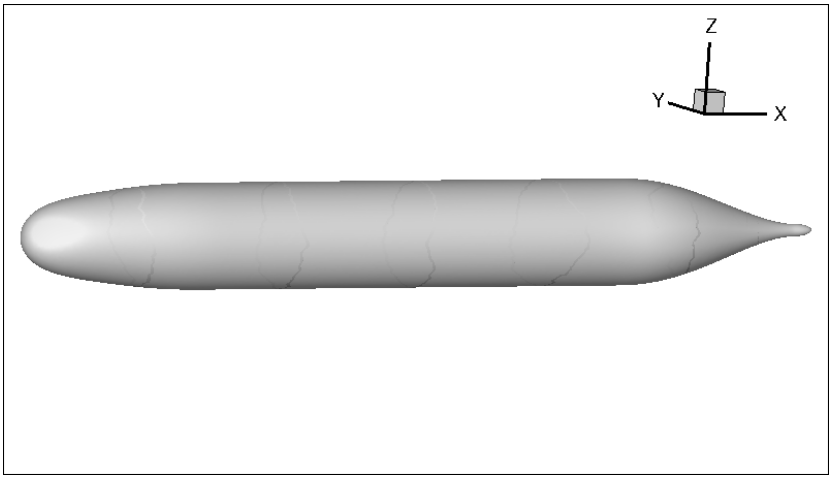


Figure 5.25 – The geometry of the SUBOFF axisymmetric hull.

In this case the axisymmetric hull of the SUBOFF model is initially simulated at 0^o angle of attack with the free-stream Reynolds number equal to *12.0E+6*. The flow around the same geometry is then simulated at a Reynolds number equal to *14.0E+6* at angles of attack equal to 0^o, 18^o and 30^o. All Reynolds numbers are calculated with respect to the length of the model. The geometry of the hull is presented in Figure 5.25. The original length of the model is equal to *14.291667 ft (4.356 m)* with a maximum diameter equal to *1.66666667 ft (0.508 m)*. The utilized mesh for this case is presented in Figures 5.26 and 5.27. The three-dimensional hybrid grid consisted of *2,132,623* nodes, *4,479,251* tetrahedrons and *2,713,100* prisms. Attention was directed to the stern area of the model, where flow separation phenomena were expected to occur; thickening of the surface mesh was applied in those regions. The same computational mesh has been utilized for all simulated flows.

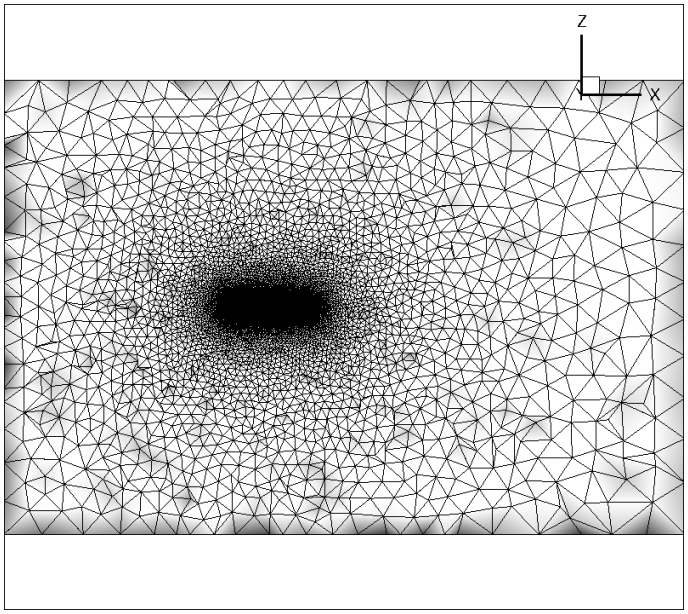


Figure 5.26 –Utilized mesh for the SUBOFF hull case.

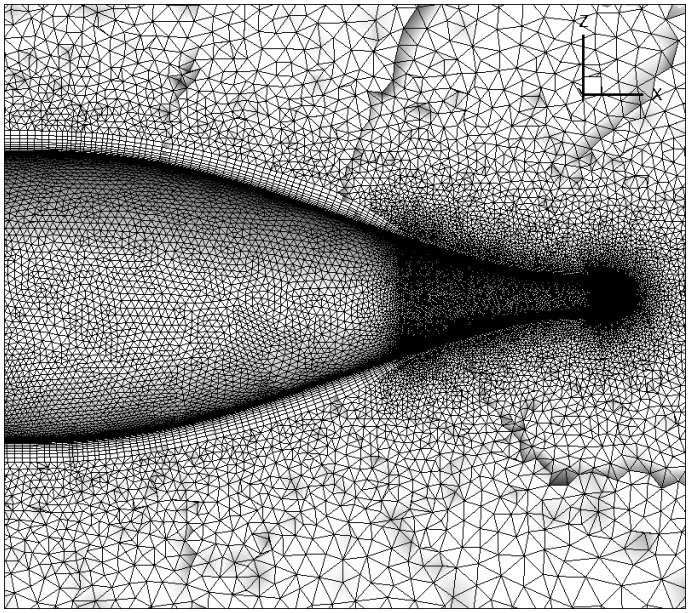


Figure 5.27 – Detail of SUBOFF hull mesh at the stern area.

Simulation was performed on a DELL T7500 workstation with two Intel(R) Xeon(R)-X5650 four-core processors at 2.67 GHz. The CFL number was set equal to *0.5* and the artificial compressibility parameter β was set equal to *10.0* for all cases. The original computational domain was divided into eight sub-domains for parallel processing. The differentiation in the angle of attack was imposed on each case through the boundary conditions and the initial conditions. The simulation parameters for this test case are summarized in Table 5.8.

Table 5.8 – Simulation parameters for the steady viscous turbulent flow around the SUBOFF bare hull.

Parameters	
Type of flow	Steady viscous turbulent
Reynolds number	12.0E+6, 14.0E+6
Angle of attack	0^o, 18^o, 30^o
Grid density	2,132,623 nodes 4,479,251 tetrahedrons 2,713,100 prisms
Artificial compressibility parameter *β*.	10.0
Number of partitions	8
Number of agglomerations	4
CFL	0.5
Computer system	DELL T7500 workstation with two Intel(R) Xeon(R)-X5650 four-core processors at 2.67 GHz

Figure 5.28 illustrates the pressure coefficient distribution along the middle of the hull geometry at 0^o angle of attack for a Reynolds number equal to *12.0E+6*, along with reference numerical results. Very good agreement is observed between the results obtained by the current solver and those found in the literature. The pressure coefficient distribution at the same region for the simulation with Reynolds number equal to *14.E+6* is presented in Figure 5.29, while Figure 5.30 contains the velocity profiles at x\L=*0.904*. The results for this case are compared to experimental ones [Tox08], as well as numerical ones [Gro11]. As it can be observed, there is a good agreement with the numerical results, but small discrepancies are observed when compared to the experimental ones. Concerning the discrepancies found at the pressure coefficient distribution, these are considered negligible as the general form of the pressure distribution is well predicted. The poor agreement with the experimental results at the velocity profile data was also observed by Gross et al. [Gro11].

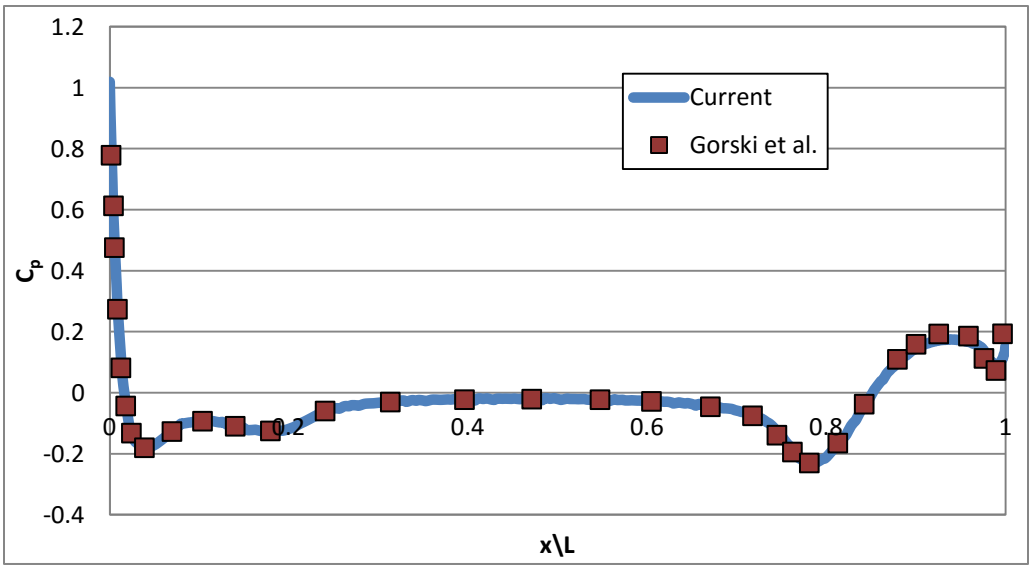


Figure 5.28 – Pressure coefficient distribution along the middle of the hull geometry at 0^o angle of attack (Re=12.0E+6).

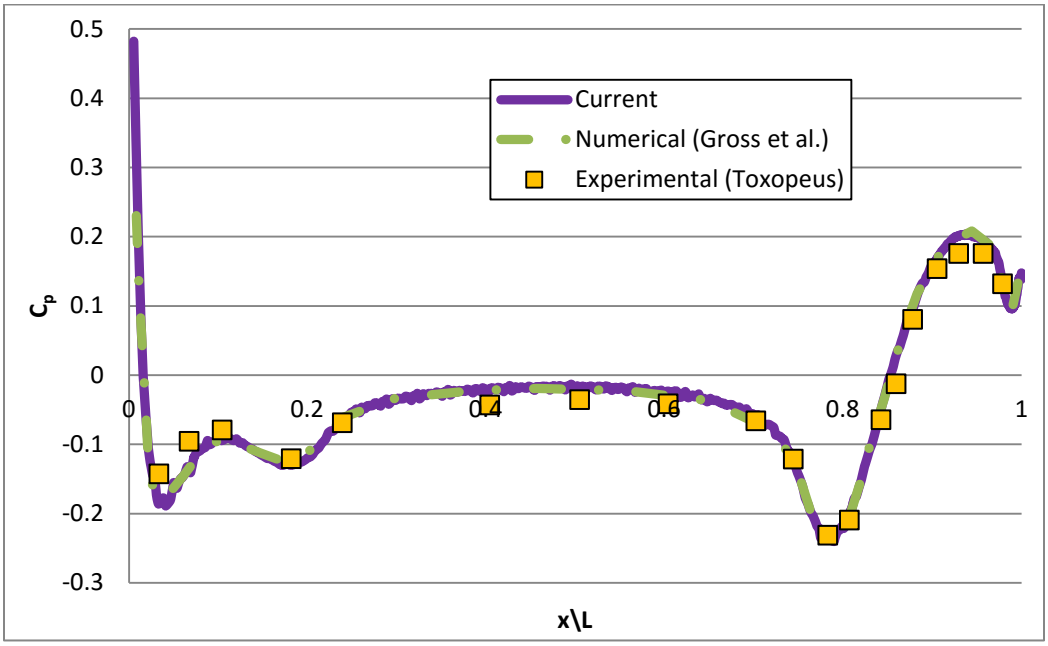


Figure 5.29 – Pressure coefficient distribution along the middle of the hull geometry at 0° angle of attack (Re=14.0E+6).

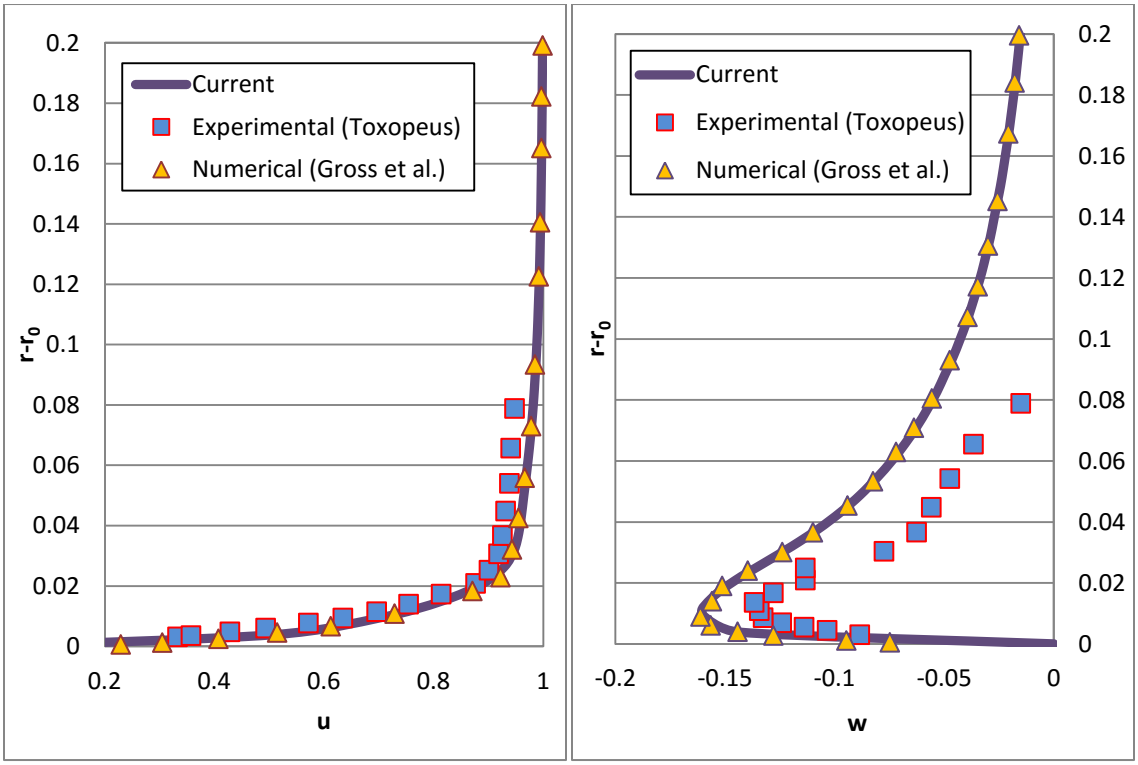


Figure 5.30 – Velocity profiles at x\L=0.904 of the bare hull model geometry at 0° angle of attack (Re=14.0E+6).

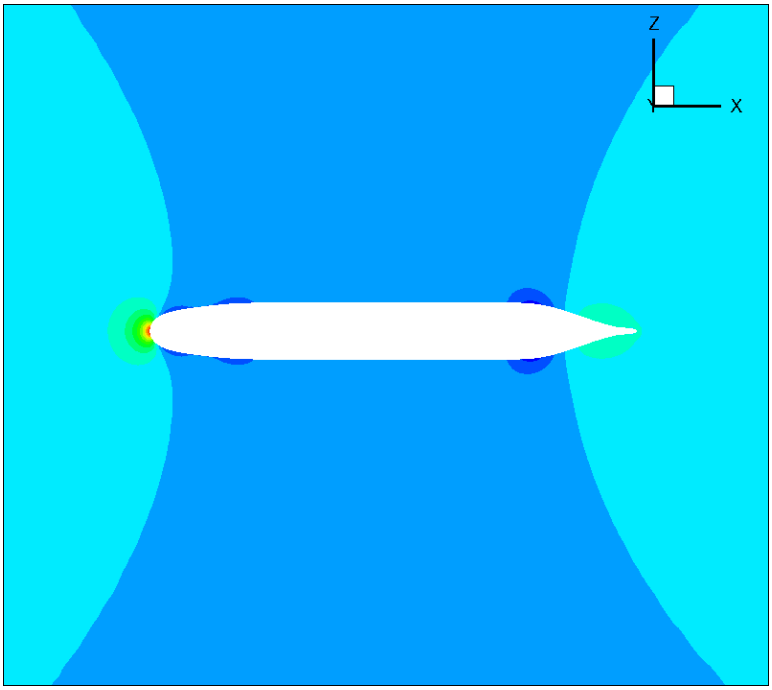


Figure 5.31 – Pressure contours at the mid-plane of the hull geometry at 0^o angle of attack (Re=12.0E+6).

Figure 5.31 contains the pressure contours around the hull model at 0^o angle of attack, for a Reynolds number equal to *12.0E+6*. The turbulent kinematic viscosity contours for the same test case are presented in Figure 5.32. Finally, in Figure 5.33 the streamwise velocity contours at the stern region of the bare hull model, along with the velocity streamlines are illustrated.

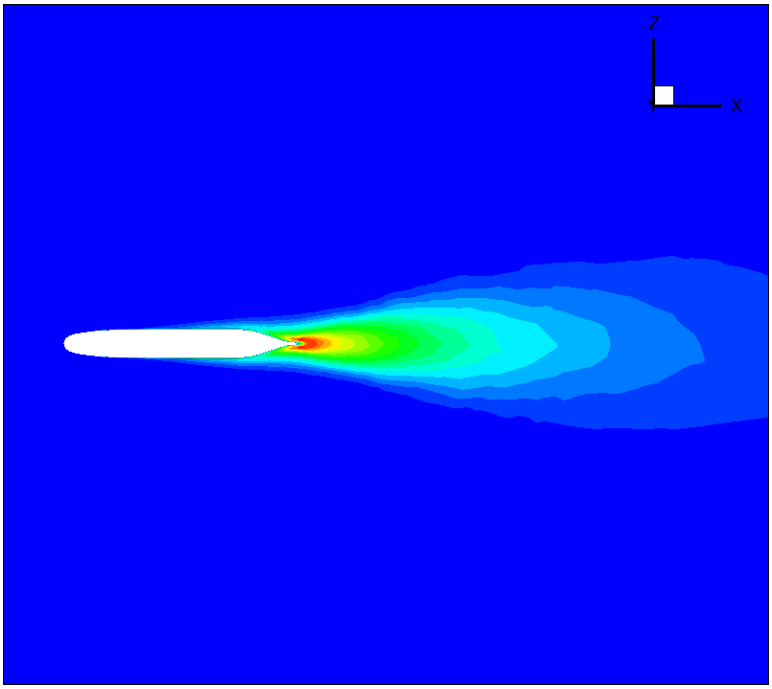


Figure 5.32 – Turbulent kinematic viscosity contours along the middle of the hull geometry at 0^o angle of attack (Re=12.0E+6).

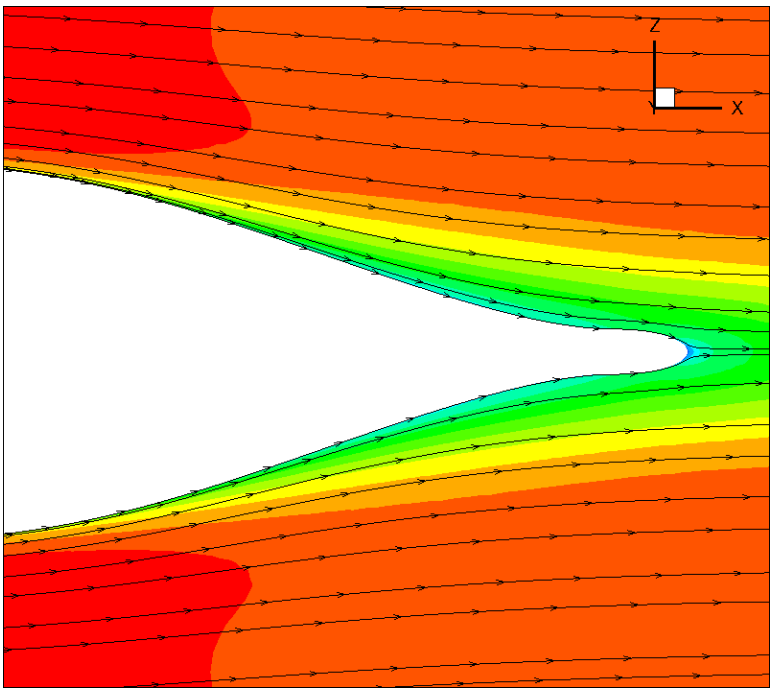


Figure 5.33 – Streamwise velocity contours along with velocity streamlines at the stern region of the SUBOFF bare hull geometry, at 0^o angle of attack (Re=12.0E+6).

While there are no significant changes in the flow when the Reynolds number increases to *14.0E+6*, there are major differences when the flow angle changes radically, to 18^o and then to 30^o. The differences in the streamwise velocity contours, as well as the turbulent kinematic viscosity contours, between the simulations with the three flow angles, with Reynolds number equal to *14.0E+6*, are presented in Figure 5.34. As it was expected, at 0^o angle of attack the turbulent kinematic viscosity increases at the stern area of the hull in a symmetric fashion, leading to a thicker turbulent layer. This increase in μ_τ is due to the minor flow separation phenomena that occur in that region, as the thickness of the hull decreases. As the angle of attack increases the boundary layer at the windward side becomes thinner, while it thickens at the leeward side. This observation can hint at the existence of three-dimensional flow separation phenomena at the leeward side of the hull with increasing the angle of attack. These phenomena can be easily identified by sketching the skin friction lines on the surface of the model. In Figure 5.35 the skin friction lines on the windward and leeward side of the SUBOFF hull are presented for the three different attack angles. At higher angles of attack the points where the skin friction lines converge indicate flow separation regions.

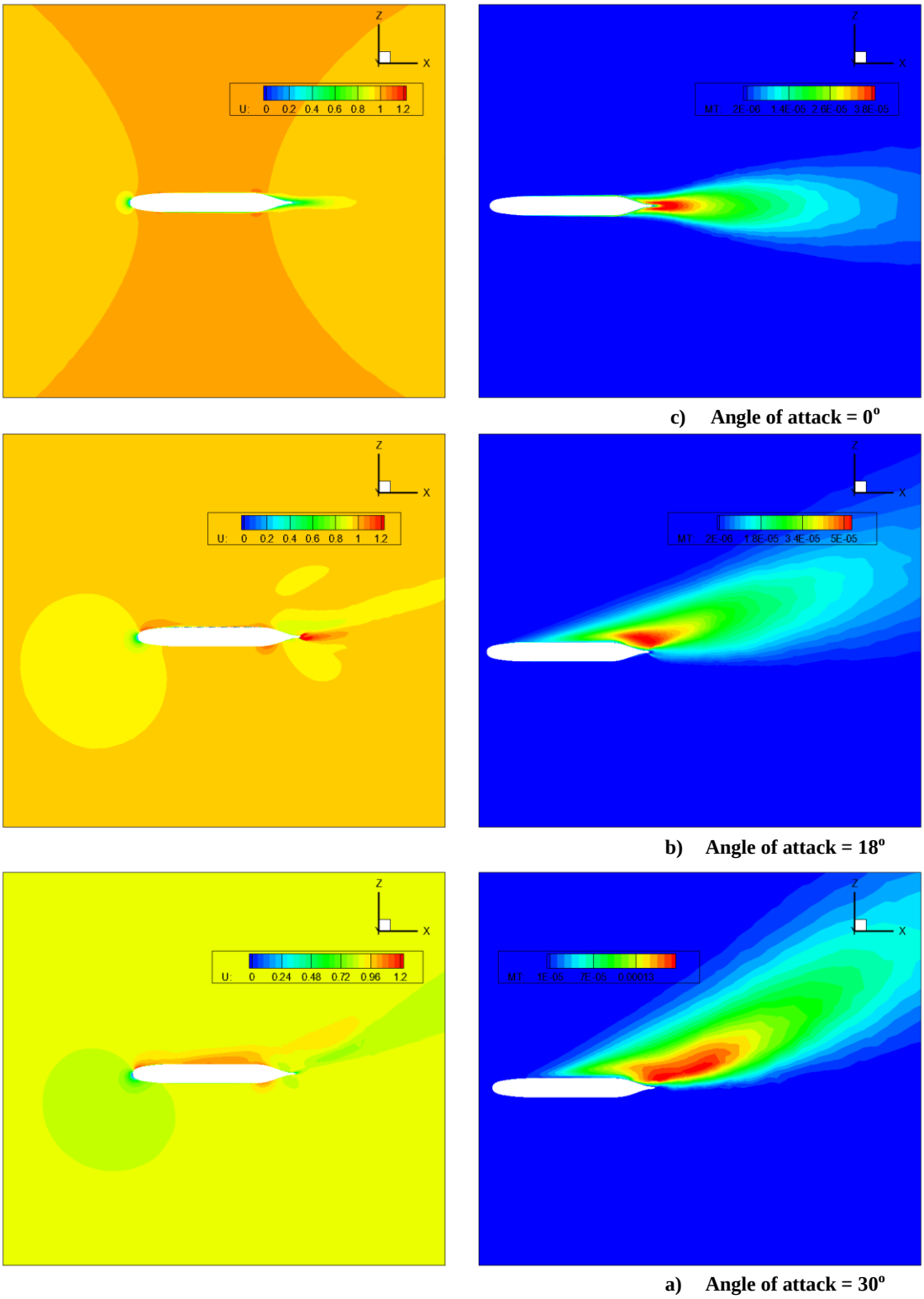


Figure 5.34 – Streamwise velocity contours (left) and turbulent kinematic viscosity contours (right) for the turbulent flow around a bare submarine hull at angles of attack equal to a) 0°, b) 18° and c) 30°.

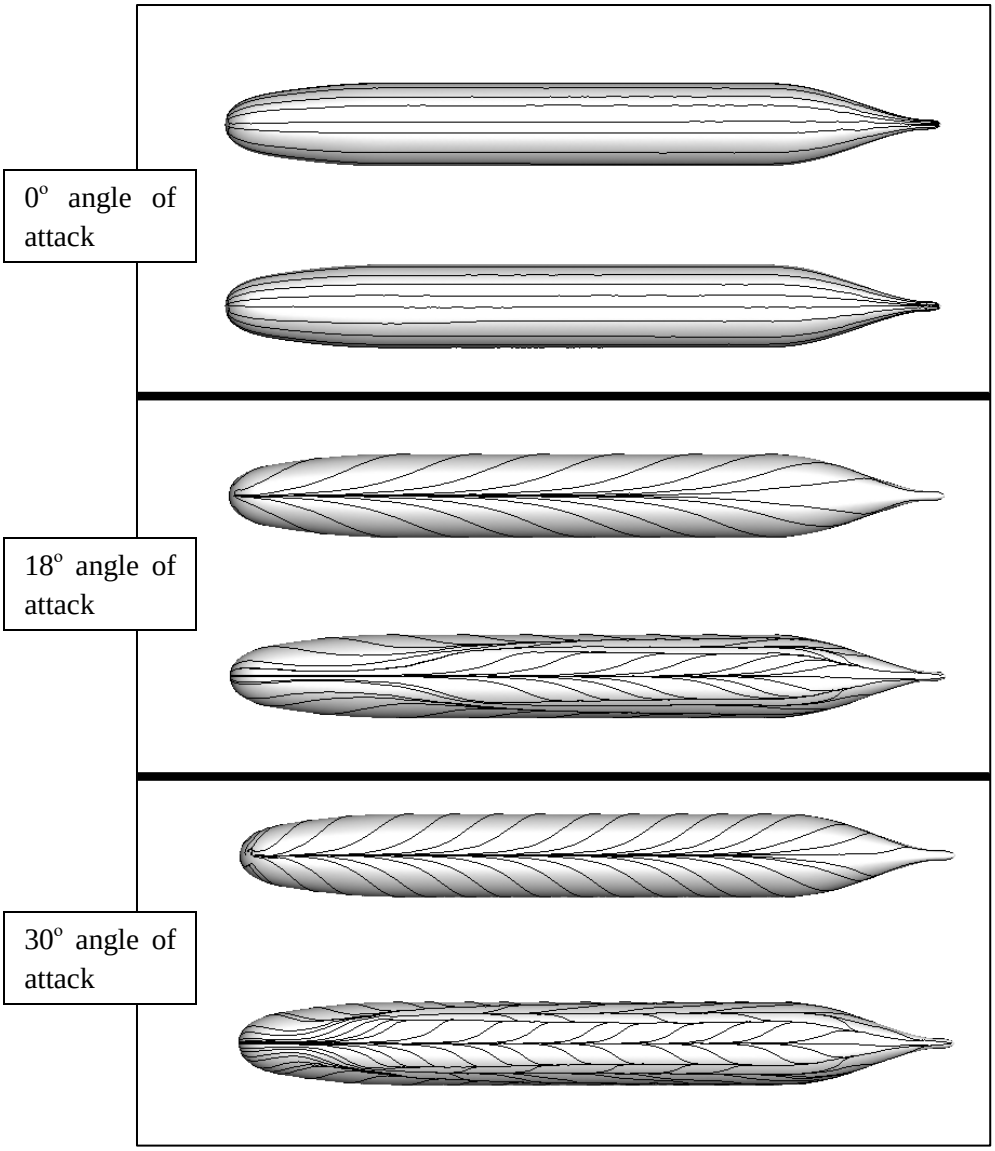
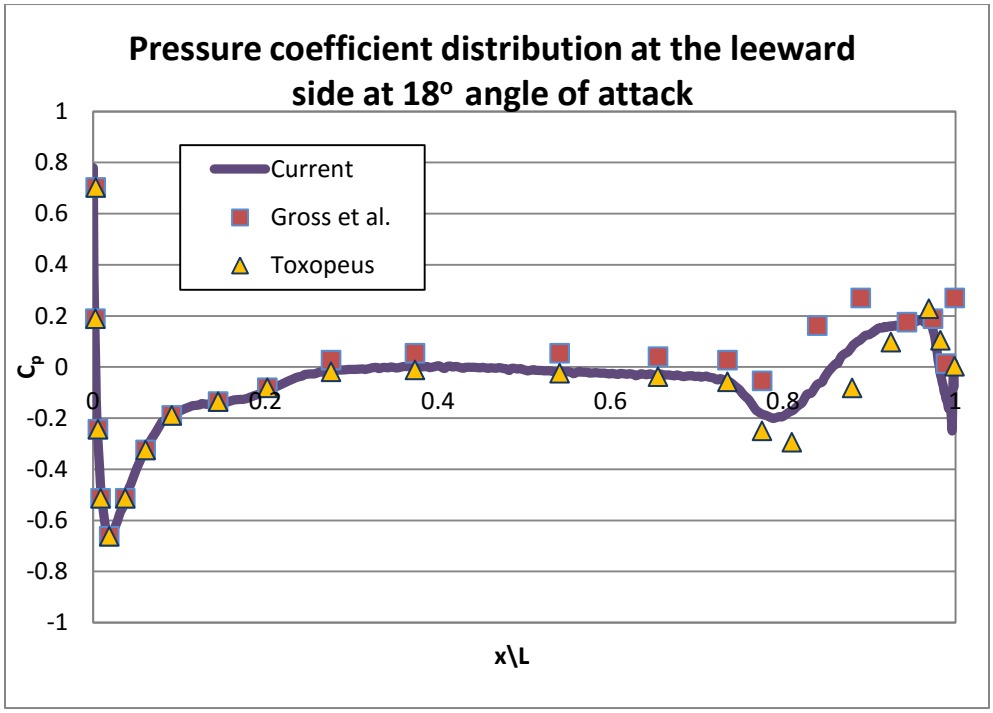


Figure 5.35 – Skin friction lines on the surface of the hull geometry at three angles of attack. Lines are sketched on the windward (upper) and leeward (lower) sides.

For the numerical evaluation of the cases with angles of attack equal to 18° and 30° the pressure coefficient distributions have been evaluated on the windward and leeward side of the hull, as well as the intermediate side. In figure 5.36 the pressure coefficient distribution along the midplane of the bare hull geometry at the leeward and windward side is presented. The results from the *Galatea-I* code are compared to numerical ones [Tox08] [Gro11]. A good agreement can be found with the corresponding results by Toxopeus with minor discrepancies at the stern of the hull, especially at the leeward side, where strong separation phenomena occur. Similar results are presented in Figure 5.37 for the case of 30° angle of attack. The results are compared with numerical ones by Gross et al. [Gro11] that were produced with their home code, as well as with those that they produced with ANSYS™ CFX™. Sufficient agreement can be found with the

results from both codes at the windward side of the hull, while at the leeward side a slight disagreement with the results from their homemade code can be observed; results that were produced with CFX™ are very close to the ones produced with *Galatea-I*.



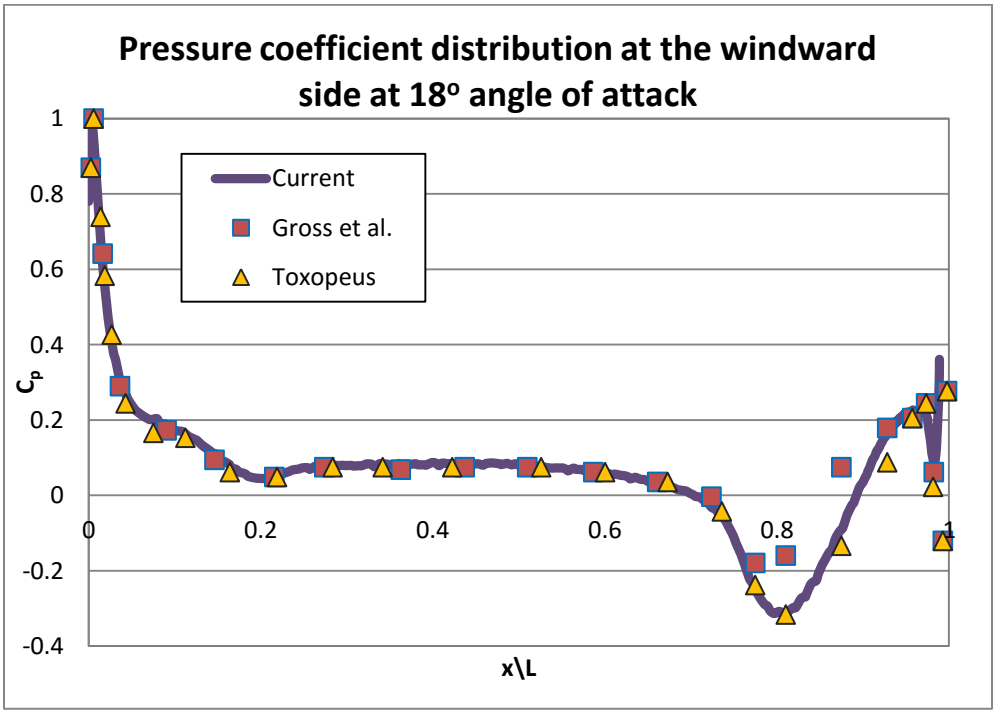


Figure 5.36 – Pressure coefficient distribution along the midplane of the SUBOFF hull at the leeward side (upper) and the windward side (lower) at 18° angle of attack.

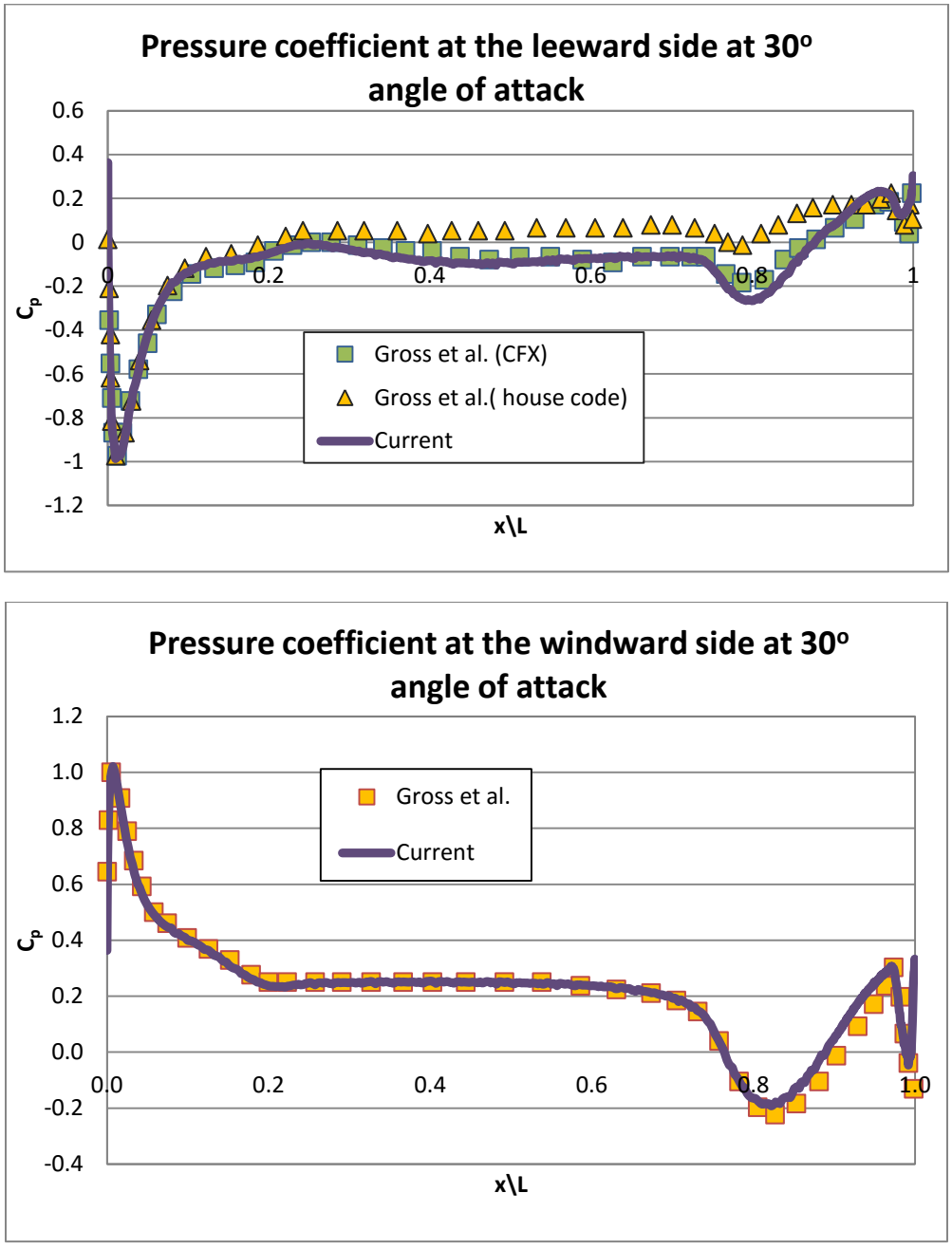


Figure 5.37 – Pressure coefficient distribution along the midplane of the SUBOFF hull at the leeward side (upper) and the windward side (lower) at 30° angle of attack.

5.1.7 Turbulent flow around a submarine hull with bridge fairwater configuration.

The next case comes from the SUBOFF family and concerns the turbulent flow around the bare hull geometry that was described in the previous section, attached with a bridge fairwater configuration. Similarly to the previous case, the fairwater sail geometry was produced using the appropriate Fortran codes. The fairwater sail is attached on the hull geometry at x\L=*0.207* and is modeled as forebody, a parallel middle body and an afterbody region; a sail cap attaches to the top of the geometry. Figure 5.38 illustrates the fairwater geometry, while the complete model is presented in Figure 5.39.

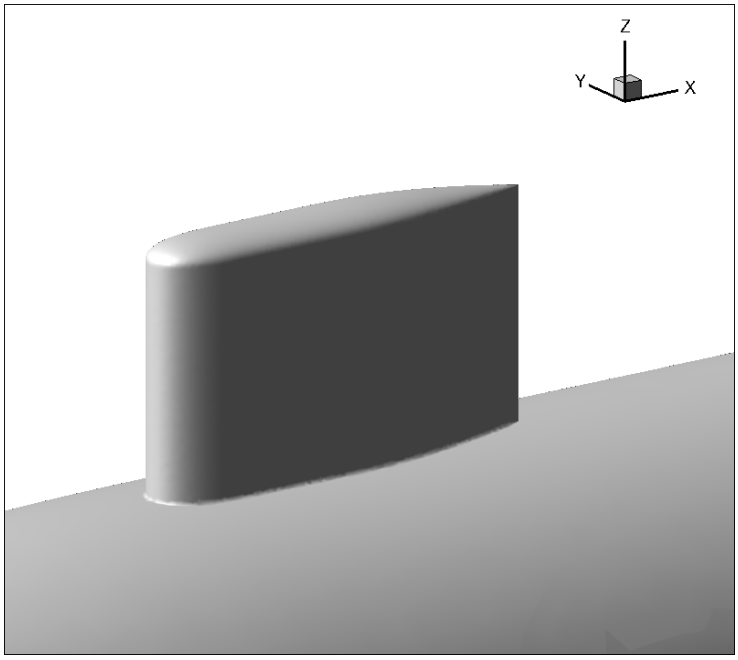


Figure 5.38 – Fairwater sail geometry attached on the SUBOFF hull geometry.

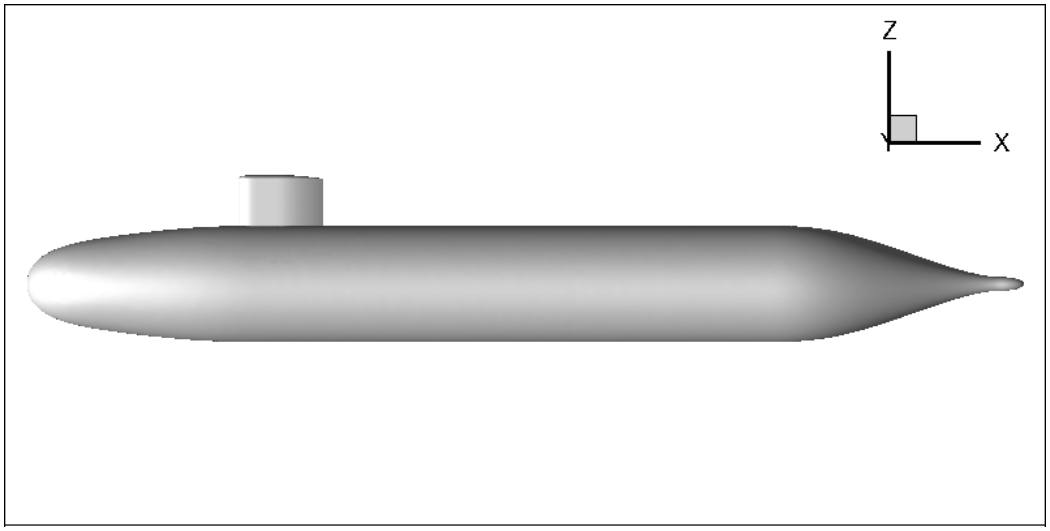


Figure 5.39 – Complete model of the SUBOFF hull with attached fairwater sail configuration.

In order to reduce the computational cost, as well as the memory consummation, the symmetry of the flow was exploited and half of the geometry was meshed. The utilized mesh consisted of *1,873,583* nodes, *5,830, 454* tetrahedrons, *1,689,493* prisms and *532* pyramids. The inflation mesh that consisted of prisms contained 15 layers, while the first layer's distance from the solid wall was equal to $5 \cdot 10^{-5} L$. Attention was focused on the fairwater area, as the most complex phenomena encountered in this case would appear near that region; thickening of the mesh was applied in certain regions of the sail geometry, especially the leading and trailing edges, as well as its base. As with the bare hull geometry, increase of the mesh density was applied at the stern area of the model. In Figure 5.40 the utilized mesh is presented as a whole, along with details of the mesh thickening that was applied in certain areas of the model. In Figure 5.41 detailed view of the mesh at the fairwater area is presented.

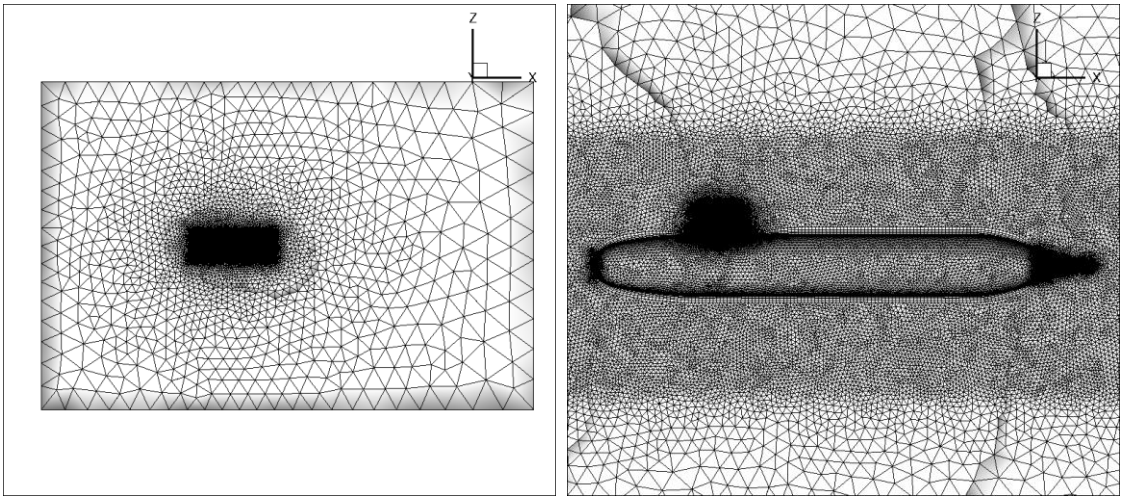


Figure 5.40 – Utilized computational mesh for the SUBOFF hull with fairwater configuration test case.

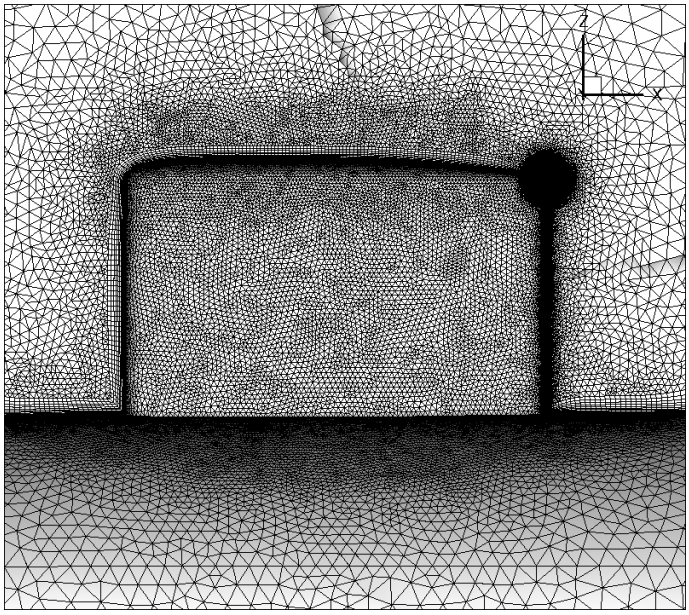


Figure 5.41 – Detailed view of the mesh at the fairwater sail area.

The Reynolds number for this case was set equal to *12.0E+6*, calculated with respect to the hull's length. The artificial compressibility parameter was set equal to *10.0* and the explicit Runge-Kutta temporal discretization scheme was realized with a CFL number equal to *0.5*. Simulation was performed on a Dell T7500 workstation with two Intel® Xeon® X5660 four-core processors at 2.80 Ghz. For parallel processing the utilized mesh was partitioned into eight sub-domains. A summary of the simulation parameters can be found in Table 5.9.

Table 5.9 – Simulation parameters for the steady viscous turbulent flow around the SUBOFF hull with fairwater sail configuration.

Parameters	
Type of flow	Steady viscous turbulent
Reynolds number	12.0E+6
Angle of attack	0^o
Grid density	1,873,583 nodes 5,830,454 tetrahedrons 1,689,493 prisms 532 pyramids
Artificial compressibility parameter *β*.	10.0
Number of partitions	8
Number of agglomerations	4
CFL	0.5
Computer system	Dell T7500 workstation with two Intel® Xeon® X5660 four-core processors at 2.80 Ghz.

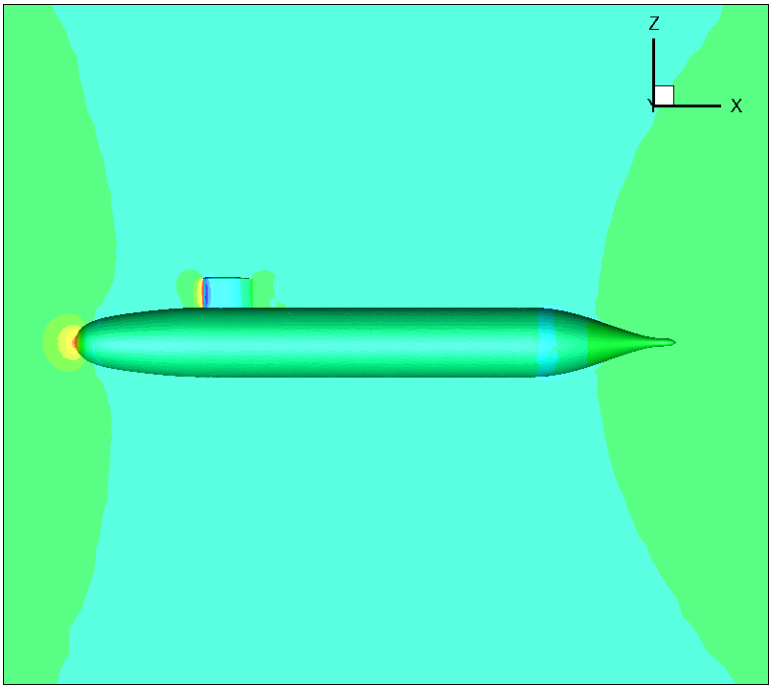


Figure 5.42 – Pressure contours on the surface and around the SUBOFF hull and fairwater sail model.

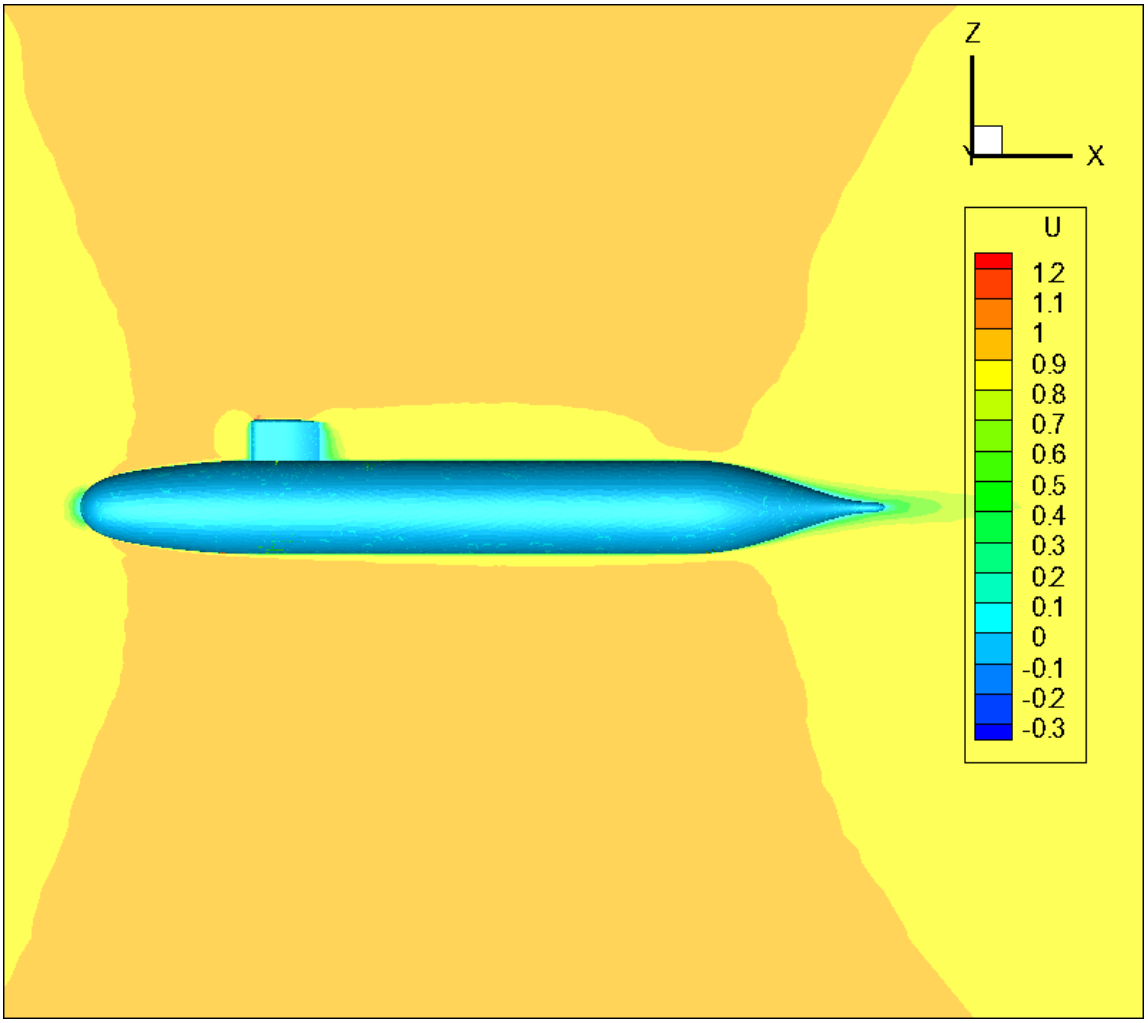


Figure 5.43 – Streamwise velocity contours on the surface and around the SUBOFF hull and fairwater sail model.

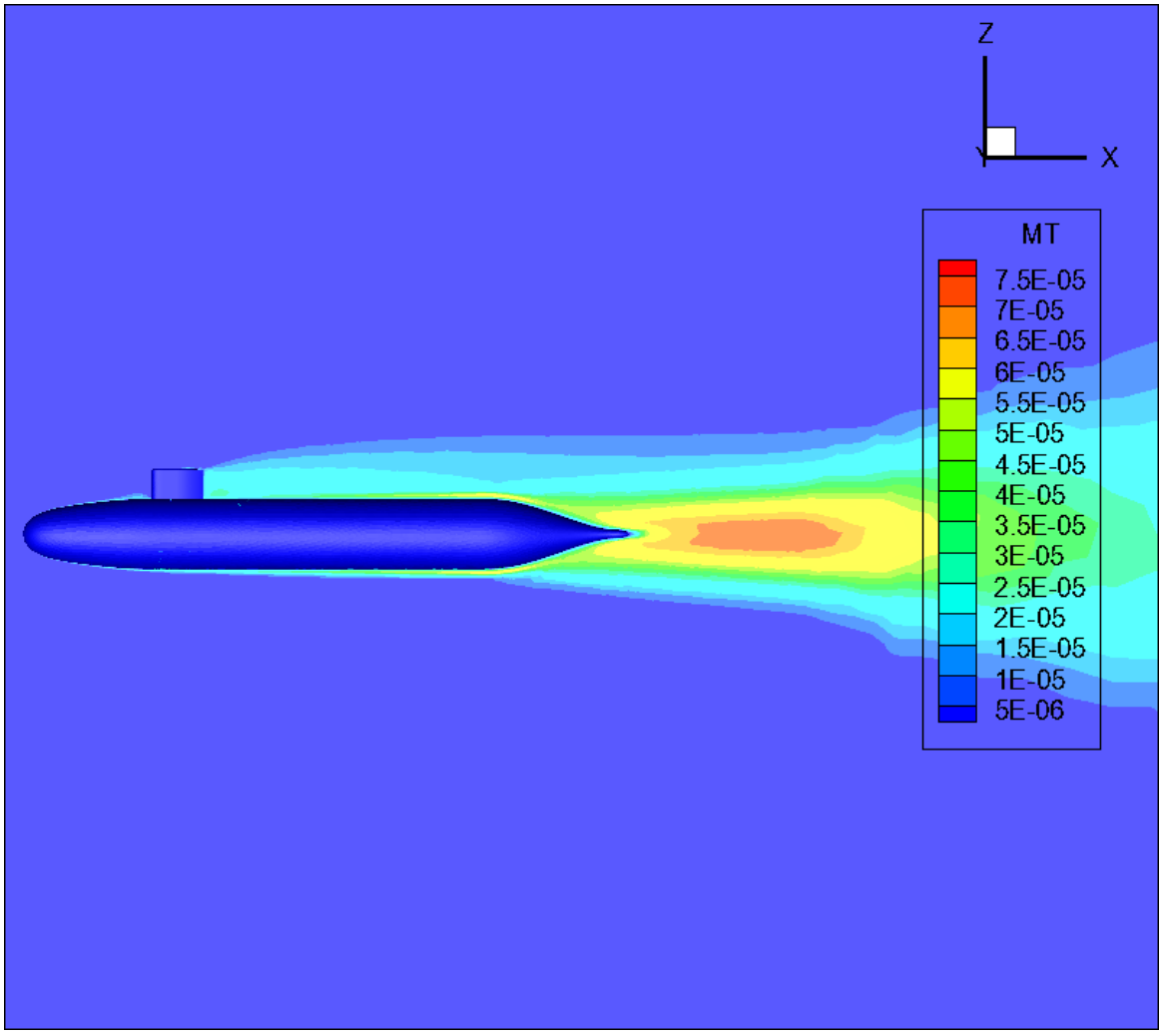


Figure 5.44 – Turbulent kinematic viscosity contours on the surface and around the SUBOFF hull and fairwater sail model.

As it was expected, due to the fairwater configuration, a thick turbulent boundary layer can be observed at the upper region of the hull, due to the wake of the sail. As it is visible in Figures 5.42 and 5.43, a stagnation point exists at the forebody region of the sail, that leads to high pressure values (Figure 5.42) and low streamwise velocity values (Figure 5.43). The generated turbulent boundary layer at the wake of the sail is visible in Figure 5.44, where the turbulent kinematic viscosity is plotted.

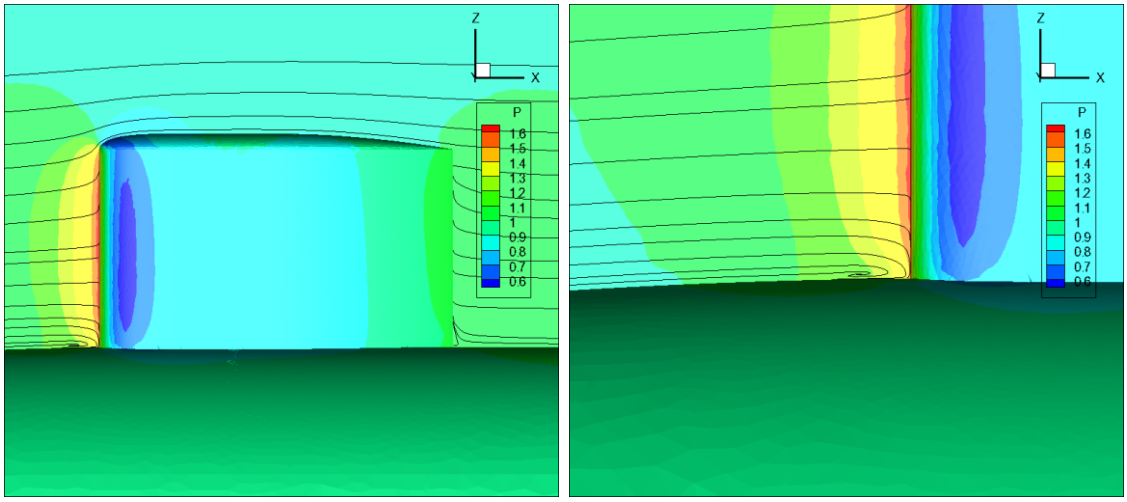


Figure 5.45 – Pressure contours and velocity streamlines around the fairwater sail (left). Detail of the horseshoe vortex developing before the stagnation point at the base of the sail (right).

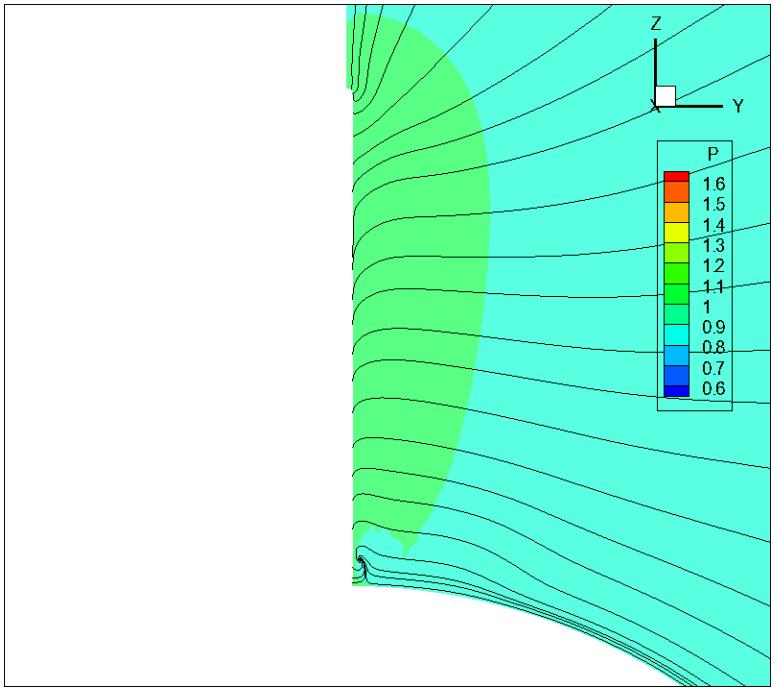


Figure 5.46 – Pressure contours and velocity streamlines on the Y-Z plane at the trailing edge of the fairwater sail.

At the base of the fairwater configuration a horseshoe vortex is developing as the flow at the stagnation point is swept downwards and interferes with the boundary layer at the hull. The vortex is presented in Figure 5.45, where on the left the pressure contours, along with the velocity streamlines are sketched around the fairwater sail, while on the right a detail at the base upwind the sail is shown, with a more obvious illustration of the horseshoe vortex. On the same figures there can be seen that there is considerable turning of the flow over the top of the sail. In Figure 5.46 the area at the trailing edge of the sail is illustrated, with the pressure contours and crossflow velocity streamlines at that area. The trailing vortex at the base of the sail, part of the horseshoe

vortex that is presented in Figure 5.47 is also evident. The flow in this figure is pictured as if the viewer would look forward, towards the bow of the submarine. Moreover, on the same figure, a secondary vortex can be observed, developing at the tip of the sail. The tip vortex is of minor intensity and dissipates towards the stern of the hull, either due to viscosity or numerical factors.

In Figures 5.47, 5.48 and 5.49, the pressure coefficient on various points on the fairwater sail and on various heights was calculated and compared to numerical results by Gorski et al. [Gor90]. Although a discrepancy is observed between the current and reference results near the leading edge of the sail, the pressure coefficient at the rest of the geometry seems to agree very well with the numerical results. As the point near the leading edge (*0.207L*) is near the symmetry plane of the model, the observed discrepancy can be attributed to numerical diffusion due to the symmetry boundary conditions in that area. In Figure 5.50 the velocity streamlines around the SUBOFF hull with fairwater sail configuration are presented in a three-dimensional perspective. The forming horseshoe vortex is visible at the base in front of the fairwater sail, while the flow envelopes the whole geometry with adequate turning over the presented obstacles.

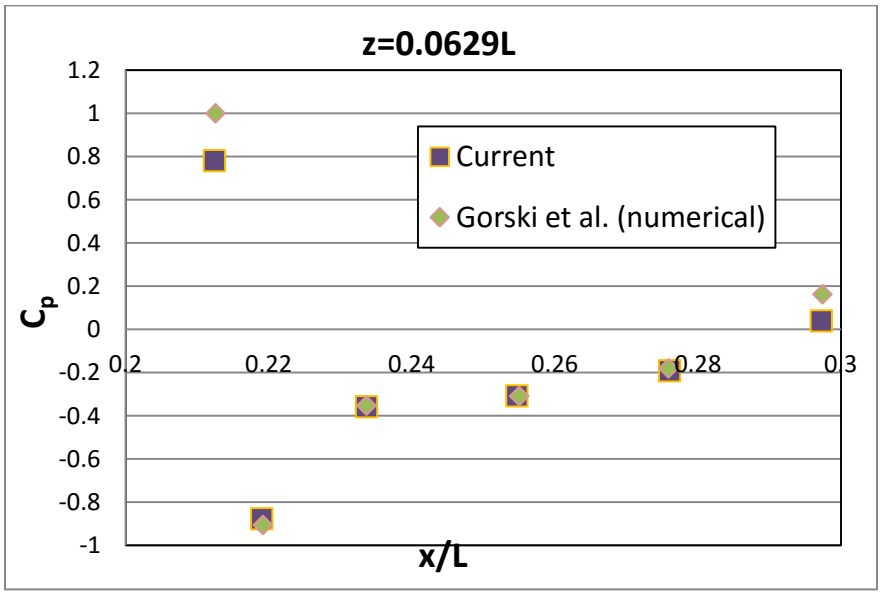


Figure 5.47 – Pressure coefficient on the sail at height z=0.0629*L*.

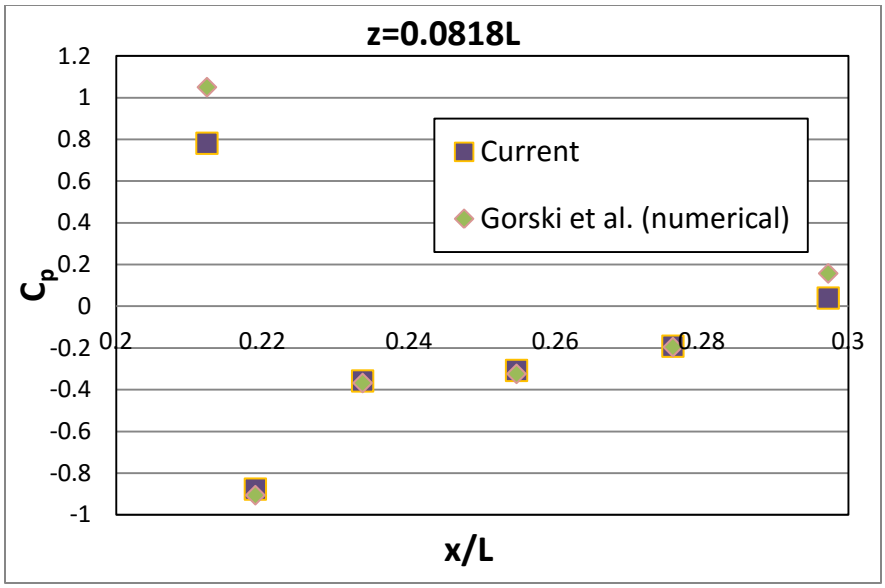


Figure 5.48 – Pressure coefficient on the sail at height z=0.0818*L*.

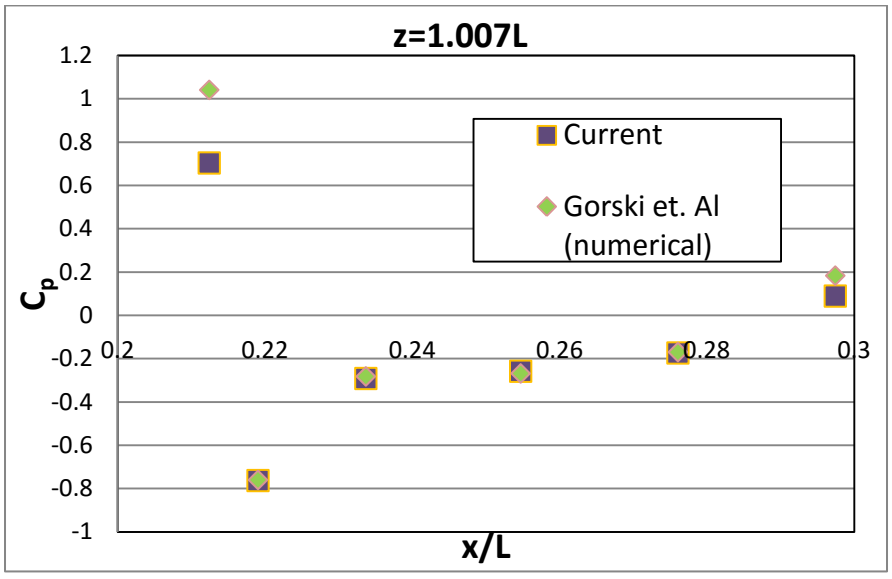


Figure 5.49 – Pressure coefficient on the sail at height z=1.007*L*.

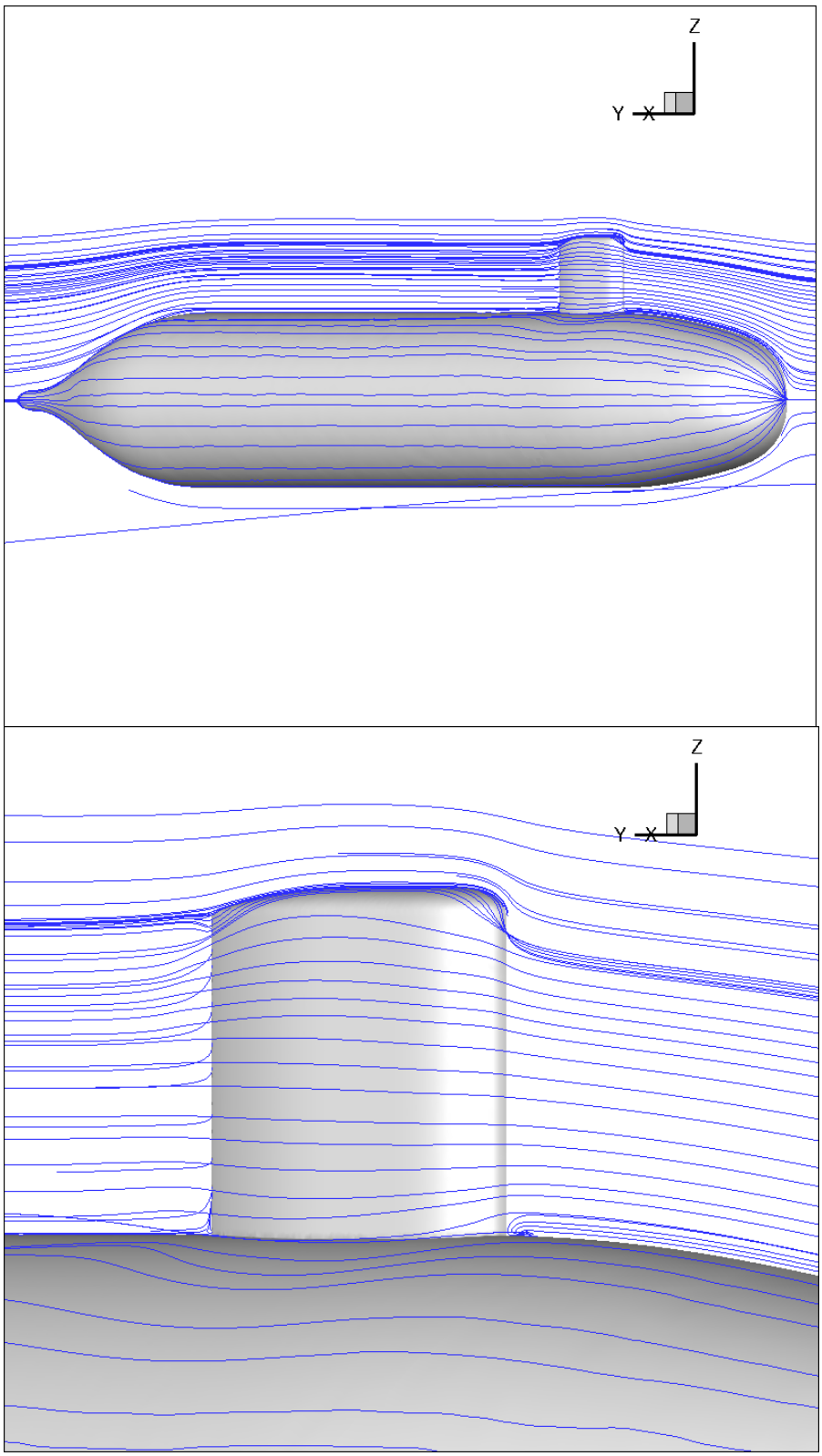


Figure 5.50 – Velocity streamlines around the SUBOFF hull with the fairwater sail configuration.

5.2 Unsteady numerical solutions

5.2.1 Unsteady laminar flow around a circular cylinder

The first test case that involves unsteady flow conditions is the popular among researchers [Rog90] [Bel95] [LiuC98] [Cha99] [Tai03] [Vra12] laminar viscous flow around a circular cylinder test case. This kind of flow produces unsteadiness at very low Reynolds numbers (just over *100*) and as such has been very well documented in the open literature. While as demonstrated in Chapter 5.1.3 with steady laminar flow around a circular cylinder a pair of symmetrical vortices appear at the wake of the cylindrical surface, when the Reynolds number increases beyond *100* these vortices begin to oscillate. In each real time step there is a large and a smaller vortex following oscillatory motion and shedding vortices at the wake of the cylinder, thus forming the so called *von Karman street* [VKa63].

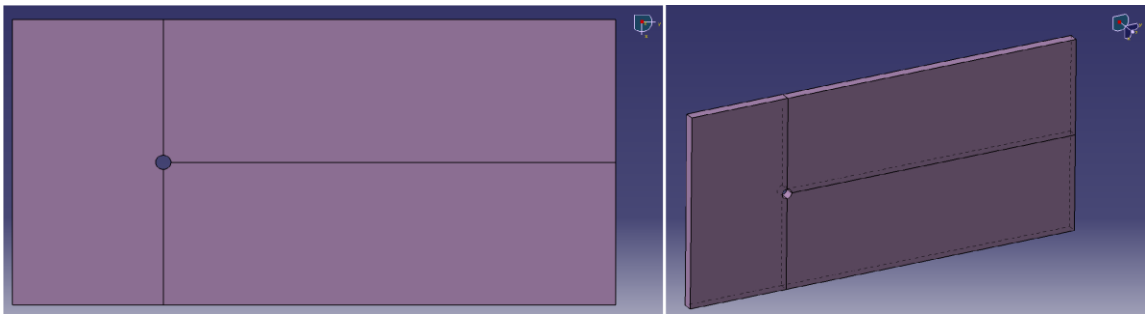


Figure 5.51 – Geometry of the computational field for unsteady flow around a circular cylinder.

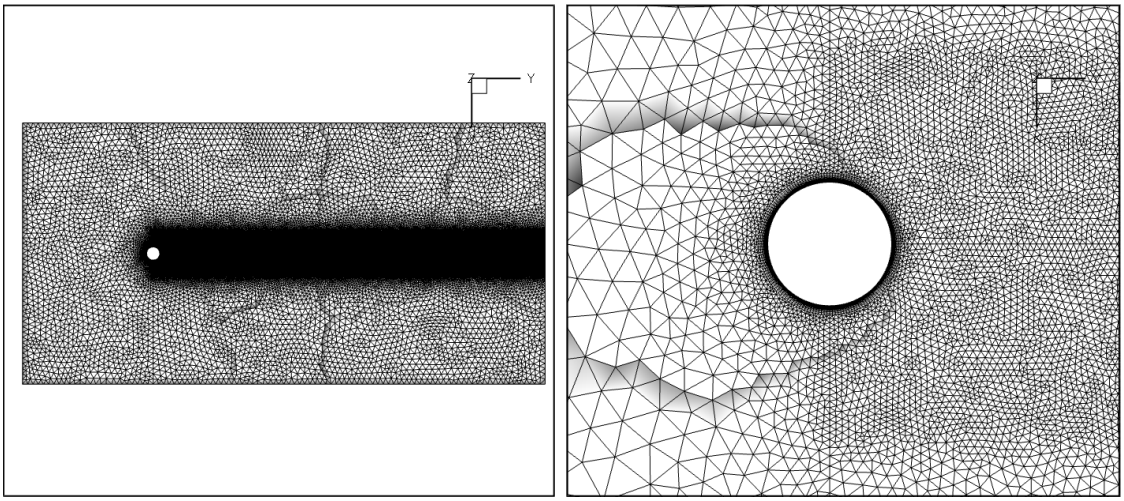


Figure 5.52 – Computational mesh for the simulation of unsteady laminar flow around a circular cylinder.

The utilized geometry for this model consisting of a circular cylinder of diameter *D=1m*, and length *L=2m* is presented in Figure 5.51. In order to fully capture the Von Karman street phenomenon the field at the wake of the cylinder was extended to *30D*, while at the upstream region, as well as over and under the cylinder the length of the computational field was equal to 10*D*. The generated computational mesh consisted of *1,361,895* nodes, *6,975,754* tetrahedrons and *274,770* prisms; the latter enveloped the cylindrical surface in 15 layers. The first layer height was equal to *1.0E-3*, while at the area of the wake sufficient thickening of the mesh was applied

to accurately calculate all the flow phenomena in that region. The computational mesh is presented in Figure 5.52.

Simulation for this case was performed on a workstation with an AMD FX™ 8150 eight-core processor at 3.62 GHz. The computational mesh was divided into eight sub-domains for parallel processing, while three coarser meshes were generated with the agglomeration method for employing the directional agglomeration multigrid methodology in order to further accelerate the simulation procedure. The Reynolds number was set equal to *200*, calculated with respect to the cylinder's diameter, while the artificial compressibility parameter was set equal to *10.0* and the CFL number was equal to *0.5*; the dimensionless real time step was equal to *0.05*. A summary of the simulation parameters can be found in Table 5.10.

Table 5.10 – Simulation parameters for the unsteady laminar flow around a circular cylinder.

Parameters	
Type of flow	Unsteady Laminar
Reynolds number	200
Angle of attack	–
Grid density	1,361,895 nodes 6,975,754 tetrahedrons 274,770 prisms
Artificial compressibility parameter β.	10.0
Number of partitions	8
Number of agglomerations	3
CFL	0.5
Dimensionless real time step	0.05
Computer system	Workstation with AMD FX™ 8150 eight-core processor at 3.62 GHz.

In Figure 5.53 the evolution of lift and pressure coefficients on the cylinder surface through real time is presented. It can be observed that unsteadiness is achieved from the first real time moments, where the oscillatory movement of the flow is at a minimum. The phenomenon increases in amplitude and achieves periodicity at approximately dimensionless real time equal to *45*. The frequency of the periodic oscillatory motion can provide the Strouhal number, which is a dimensionless number that describes the vortex shedding phenomenon and is calculated as:

$$St = \frac{f L_\infty}{U_\infty} \tag{5.1}$$

where f is the vortex shedding frequency, L_∞ is the characteristic length and U_∞ is the far field velocity. In Table 5.11 the Strouhal number, the mean lift and drag coefficients, as well as their deviations are compared to reference results found in [Rog90] [Bel95] [LiuC98] [Cha99] [Tai03] [Vra12]. It is obvious that there is sufficient agreement between the results from *Galatea-I* and the reference ones.

Table 5.11 – Comparison of flow numerical properties between current and reference solvers.

	St	**C_L**	**C_D**
Current	0.196	0.0±0.63	1.34±0.047
Rogers and Kwak [Rog90]	0.185	0.0±0.65	1.23±0.05
Belov et. al. [Bel95]	0.193	0.0±0.64	1.19±0.042
Liu et. al. [LiuC98]	0.192	0.0±0.69	1.31±0.049
Chan and Anastasiou [Cha99]	0.183	0.0±0.63	1.48±0.05
Tai et al. [Tai03]	0.195	0.0±0.64	1.31±0.041
Vrahliotis et al. [Vra12]	0.198	0.0±0.69	1.36±0.046

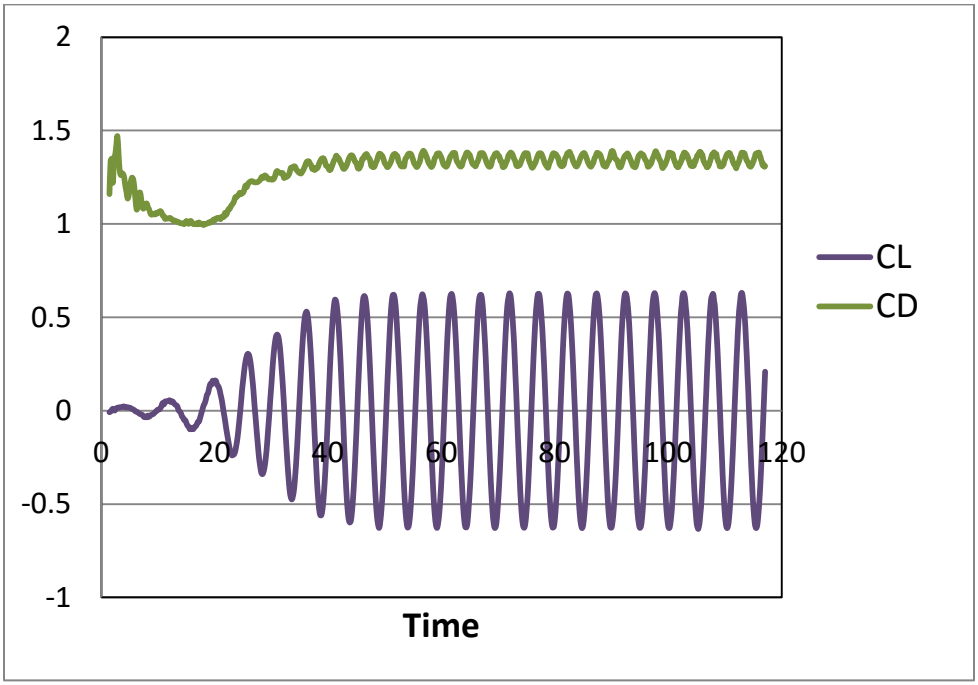


Figure 5.53 – Time history of lift and drag coefficients on the circular cylinder.

In Figures 5.54 through 5.57 the pressure contours and velocity streamlines around the circular cylinder are illustrated for four different moments in a period of the oscillatory phenomenon (*T=0, T=0.25, T=0.5* and *T=0.75*). The vortex shedding phenomenon is very clear in these illustrations, as there can be observed a periodic movement of two vortices (a major and a minor one) at the wake of the cylinder. The iconic Von Karman Street can also be observed as it propagates towards the rear boundary of the computational domain. The vortex shedding phenomenon is more clearly illustrated in Figure 5.58, where the evolution of the vorticity contours at the wake of a cylinder in a period of the phenomenon is presented.

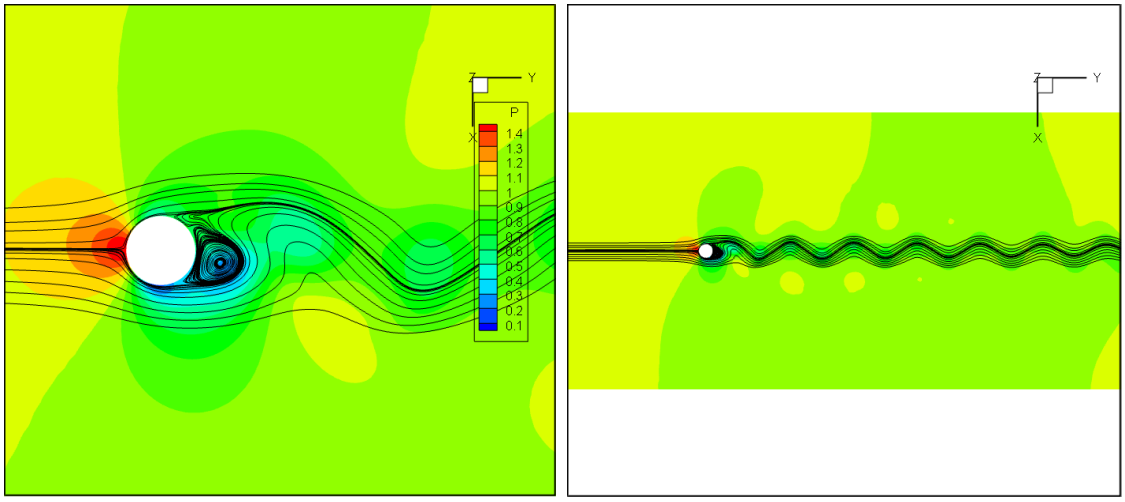


Figure 5.54 – Pressure contours and velocity streamlines around a cylinder at T=0.

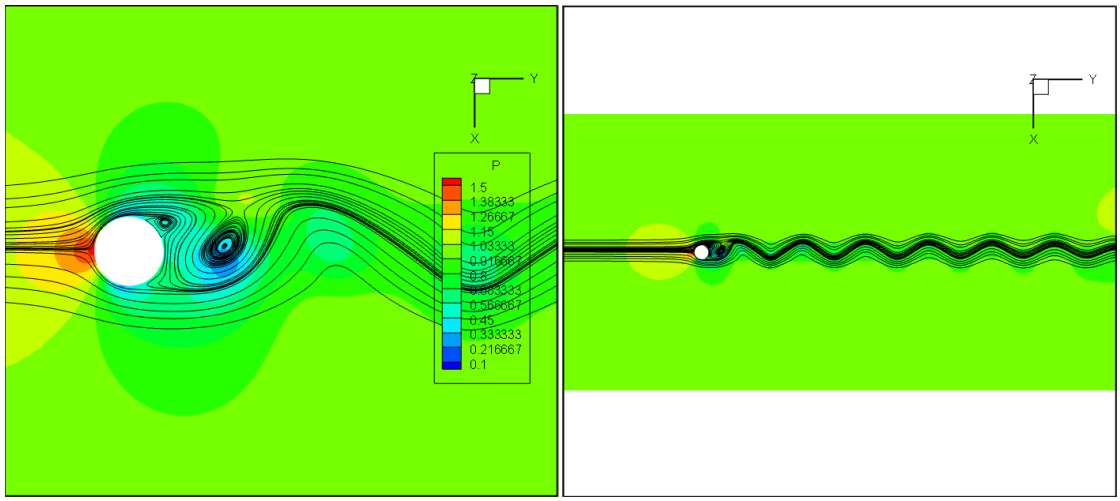


Figure 5.55 – Pressure contours and velocity streamlines around a cylinder at T=0.25.

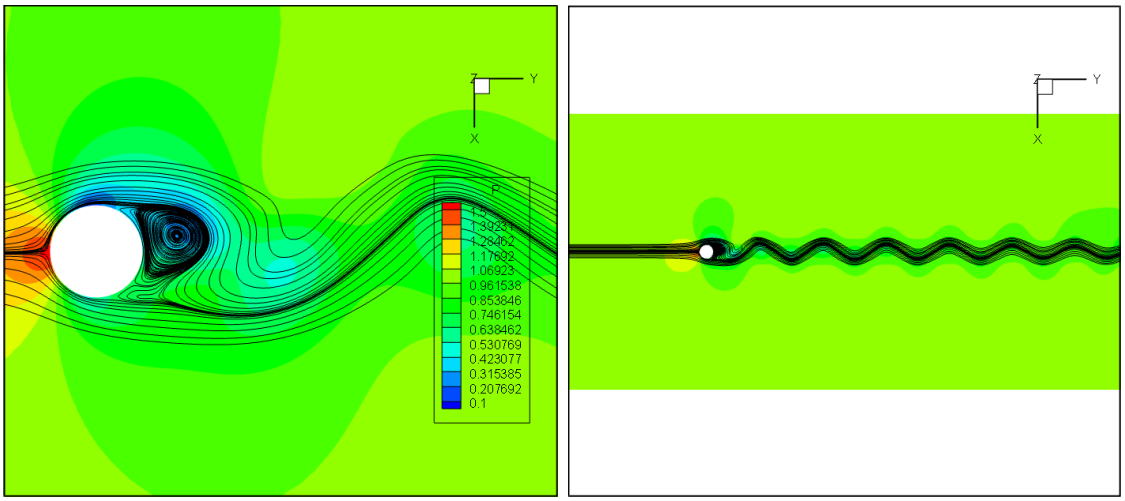


Figure 5.56 – Pressure contours and velocity streamlines around a cylinder at T=0.5.

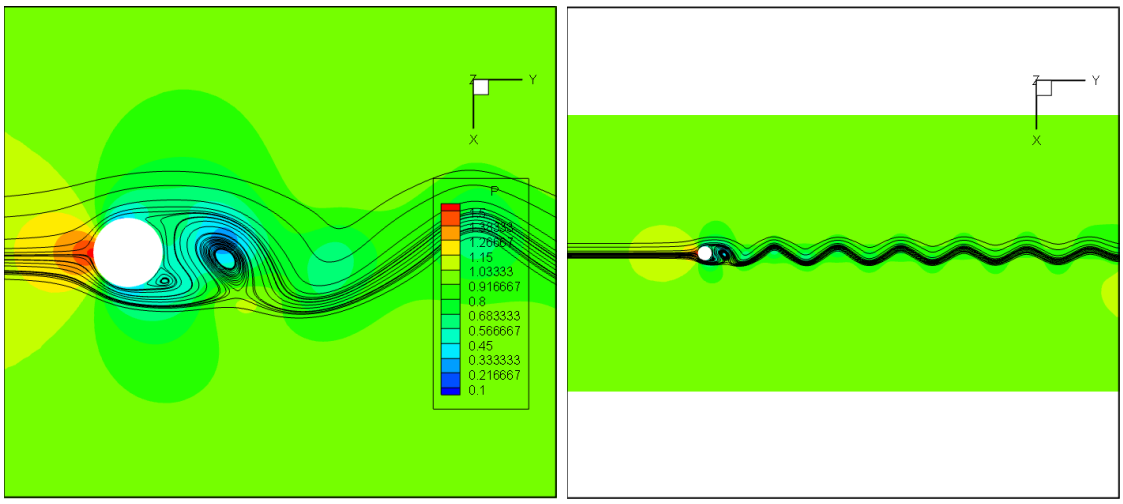


Figure 5.57 – Pressure contours and velocity streamlines around a cylinder at T=0.75.

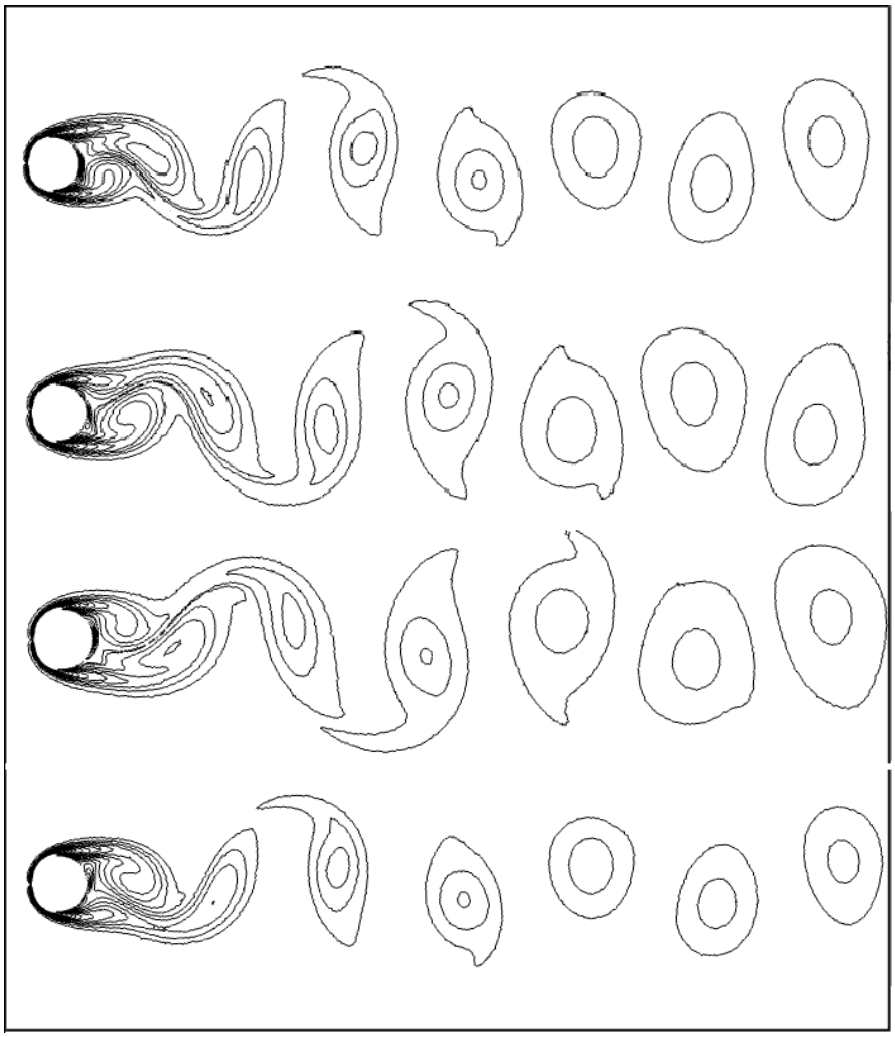


Figure 5.58 – Evolution of vorticity contours in a period of the vortex shedding phenomenon.

5.2.2 Unsteady turbulent flow around a circular cylinder.

Utilizing the same geometry as with the previous chapter, the turbulent viscous flow around a circular cylinder was simulated. In order to reduce the computation time, as expected by the turbulence modeling and the unsteady computation, a coarser mesh than the one utilized in the laminar viscous case was generated. The new mesh consisted of *654,103* nodes, *1,305,744* tetrahedrons and *810,650* prisms, with the height of the first prism being equal to *5·10-4D*, where *D* is the cylinder's diameter. The Reynolds number of the free-stream flow was set equal to *1.0E+6*, calculated with respect to *D*. The artificial compressibility parameter was set equal to *10.0*. Integration in pseudo-time was performed with a CFL number equal to *0.5*, while the simulation was advanced in time with a dimensionless real time step equal to *0.05*. The simulation was accelerated via parallel processing by dividing the initial mesh into eight sub-domains, while the agglomeration multigrid technique was employed, by generating two coarser meshes for each sub-domain, employing the full-coarsening directional agglomeration technique. A summary of the simulation parameters can be found in Table 5.12.

Table 5.12 – Simulation parameters for the unsteady turbulent flow around a circular cylinder.

Parameters	
Type of flow	Viscous Turbulent, Unsteady
Reynolds number	1.0E+6
Angle of attack	–
Grid density	654,103 nodes 1,305,744 tetrahedrons 810,650 prisms
Artificial compressibility parameter *β*.	10.0
Number of partitions	8
Number of agglomerations	3
CFL	0.5
Dimensionless real time step	0.05
Computer system	Workstation with AMD FX™ 8150 eight-core processor at 3.62 GHz.

As with the laminar viscous flow case, the resulting flow is oscillatory in nature, shedding vortices periodically and generating the iconic *Von-Karman Street*. The evolution in time of the flow around the cylindrical surface can be observed in Figure 5.59, where the lift and drag coefficients around the object are plotted as a function of time. Contrary to the results from the laminar viscous flow case, the oscillatory character of the turbulent flow is simulated from the very first real time moment, with the magnitude of the lift and drag coefficient intensifying in time and finally achieving periodicity.

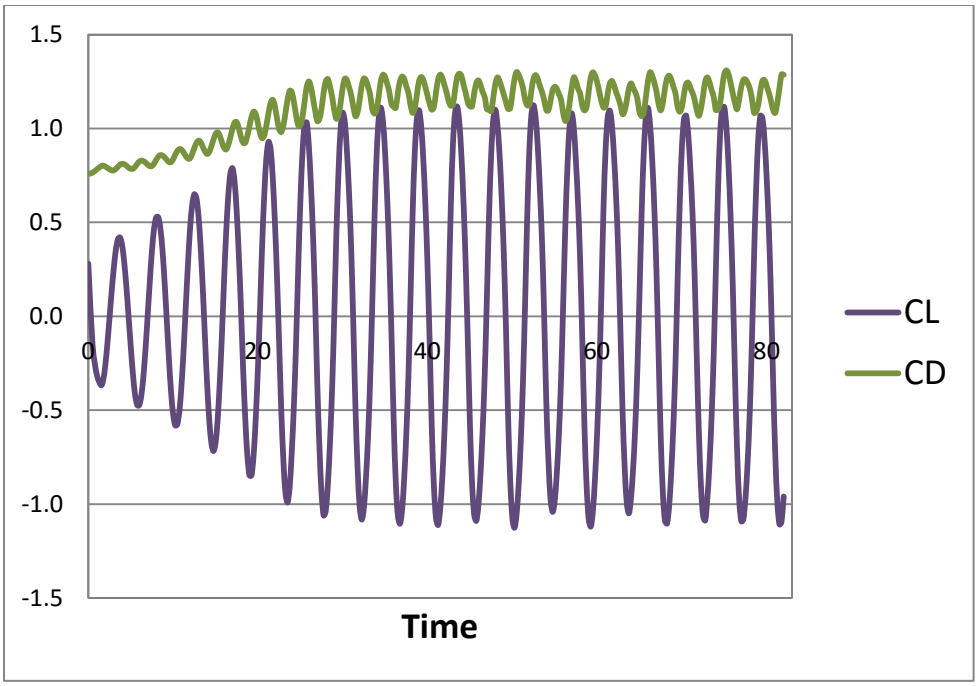


Figure 5.59 – Time history of lift and drag coefficient on circular cylinder under turbulent viscous flow.

The periodic nature of the flow can be also observed in Figures 5.60 to 5.63, where the pressure coefficient with the velocity contours is presented on the left side, while on the right side the turbulent kinematic viscosity is illustrated. The *Von Karman Street* phenomenon is obvious in these figures, especially in the illustrations of the turbulent kinematic viscosity; as the flow develops towards the rear side of the computational domain the phenomenon dissipates due to the magnitude of the turbulent kinematic viscosity. In Figure 5.64 the evolution of the vorticity contours at the wake of the cylinder are presented as they evolve in a period of the phenomenon.

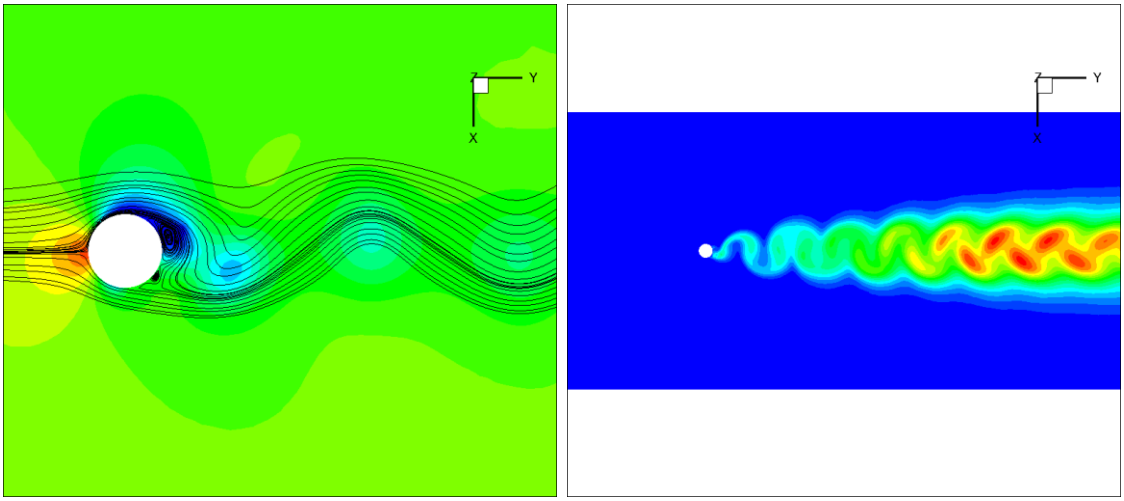


Figure 5.60 – Pressure contours with velocity streamlines (left) and turbulent kinematic viscosity contours (right) around the cylinder at T=0.

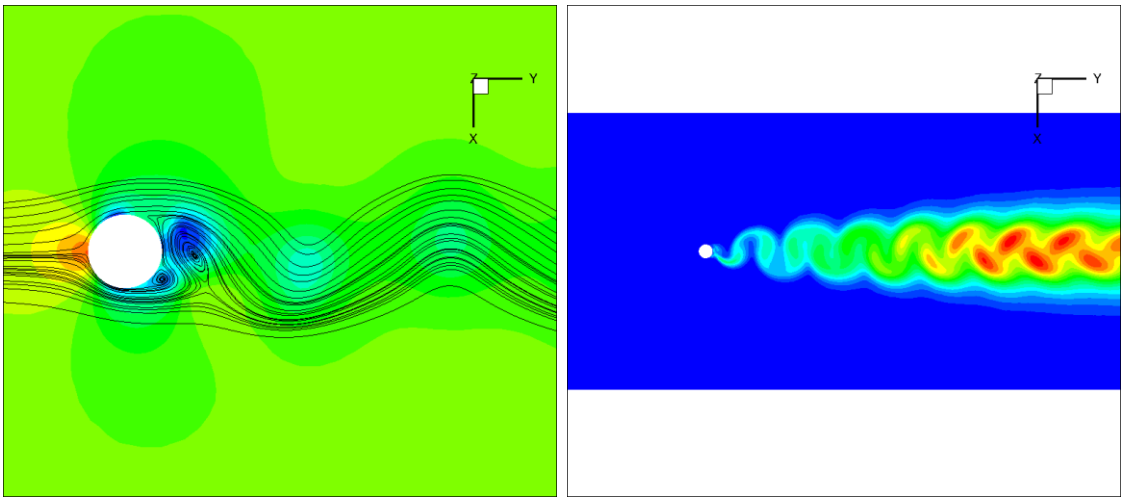


Figure 5.61 – Pressure contours with velocity streamlines (left) and turbulent kinematic viscosity contours (right) around the cylinder at T=0.25.

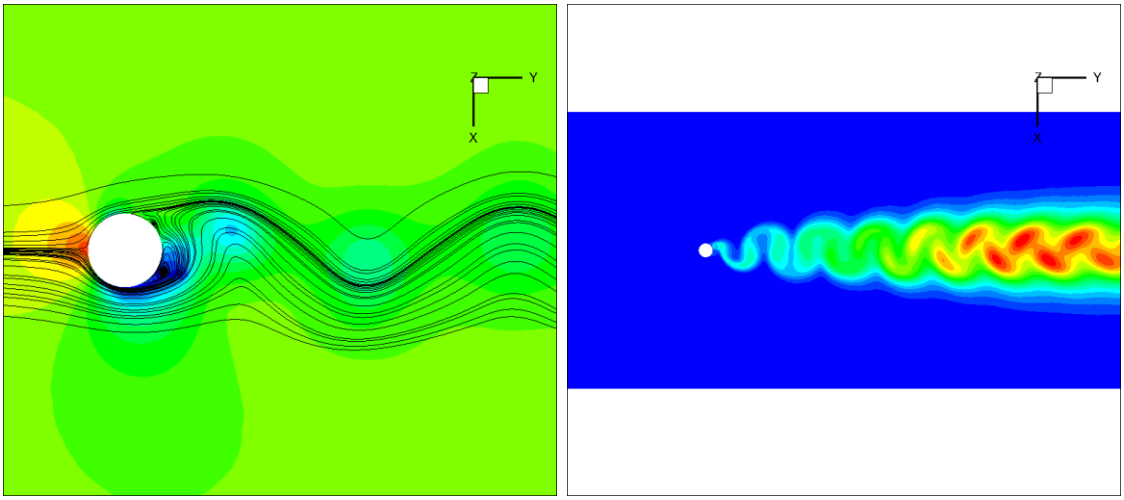


Figure 5.62 – Pressure contours with velocity streamlines (left) and turbulent kinematic viscosity contours (right) around the cylinder at T=0.5.

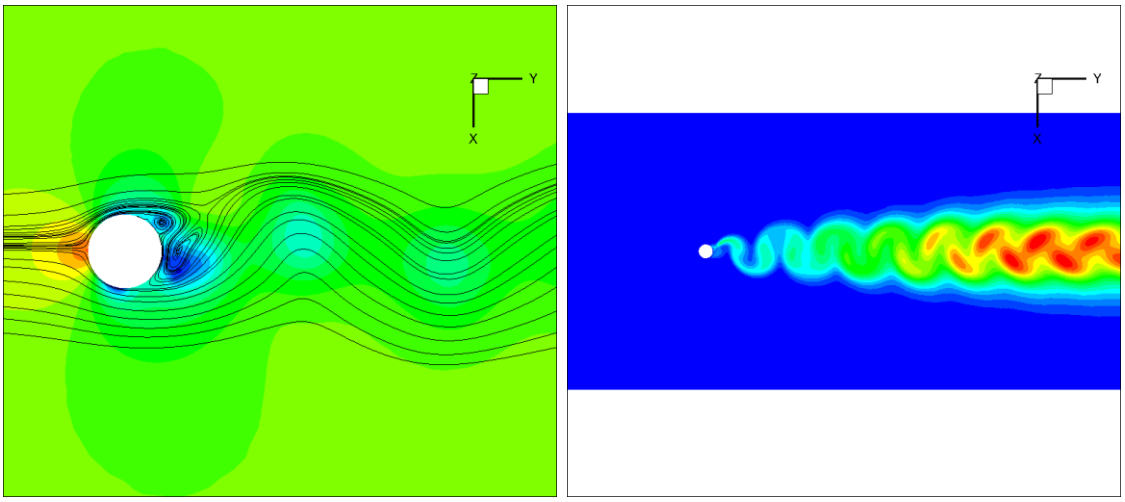


Figure 5.63 – Pressure contours with velocity streamlines (left) and turbulent kinematic viscosity contours (right) around the cylinder at T=0.75.

The qualitative evaluation of the results obtained with the current code can be found in Table 5.13, where the Strouhal number for the vortex shedding phenomenon, as well as the drag coefficient around the cylindrical surface are compared to corresponding results found in [Shi93] [Cat03]. While the result for the Strouhal number is in the same range with the reference ones, a large differentiation can be observed in the drag coefficient result.

Table 5.13 – Comparison of results for unsteady turbulent flow around circular cylinder at Re=1.0E+6.

	St	**C_D**
RANS (Current)	0.222	1.18
Shih et. al. [Shi93] (experimental)	0.203	0.24
Catalano [Cat03] (URANS)	0.31	0.40
Catalano [Cat03] (LES)	0.35	0.31

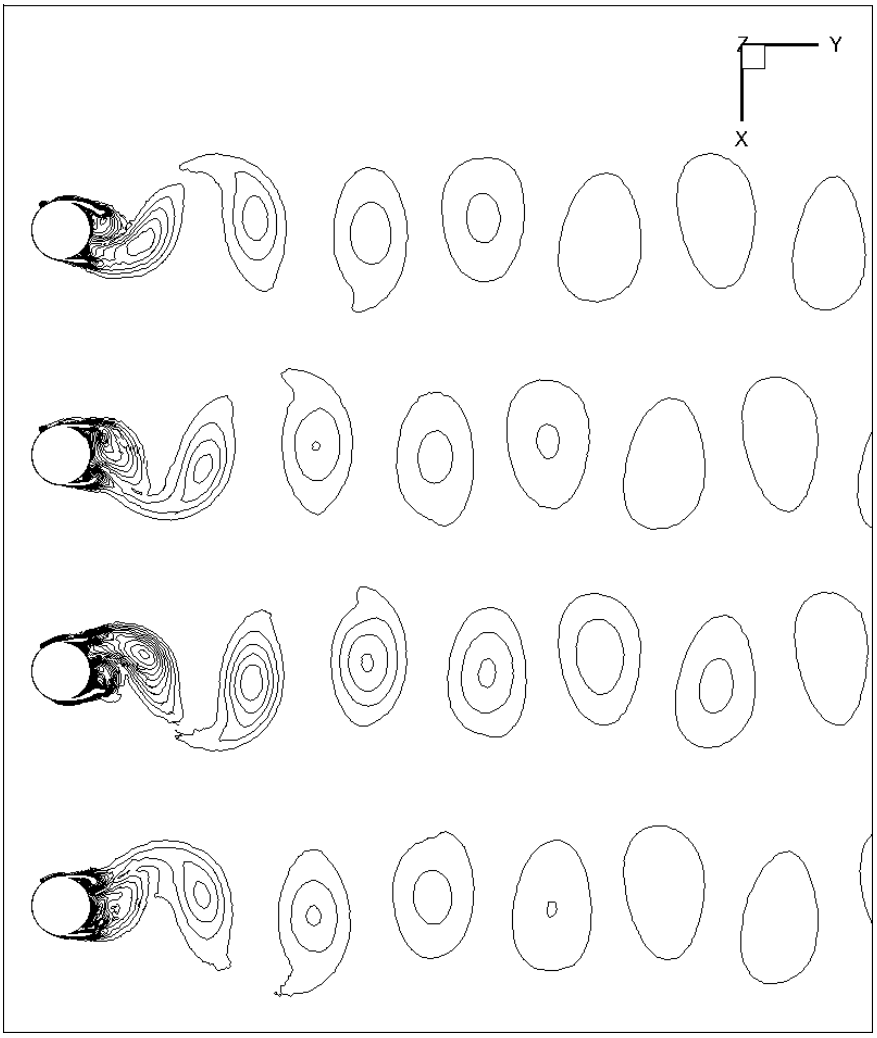


Figure 5.64 – Evolution of vorticity contours in a period of the vortex shedding phenomenon for turbulent flow around a circular cylinder.

5.2.3 Unsteady turbulent flow around a tall building.

The air flow around the Commonwealth Advisory Aeronautical Research Council (CAARC) standard tall building consists a popular test case among researchers [Pat86] [Che04] [Hua07] [Bra09] [Dag09] [Dag10] [Dan13] [Zha15], especially in cooperation with structural analysis codes and the simulation of the building deformation due to strong wind effects. Initially, the building model was proposed for the comparison of various wind tunnel techniques used for the simulation of natural wind characteristics [Mel80] [Syk83] [Bla85] [Tan86] [Oba92] [Xu92]. The original geometry of the building consisted of a simple rectangular prism with length (L) equal to *30.8m (100ft)*, width (W) equal to *45.72m (150ft)* and height (H) equal to *183.88m (600ft)*. Additional specifications for the building's geometric characteristics suggested that it had to have a flat top, without parapets, and the exterior walls had to be flat without mullions or other geometric disturbances [Mel80]. In Figure 5.65 a sketch of the geometric model that was prepared for this case is presented. The boundary walls of the computational domain were

modeled at a sufficient distance from the building, as proposed by Huang et al. [Hua07], in order to eliminate the flow obstacle effect that is caused with the interaction of the inflow velocity with the building geometry [Mur98].

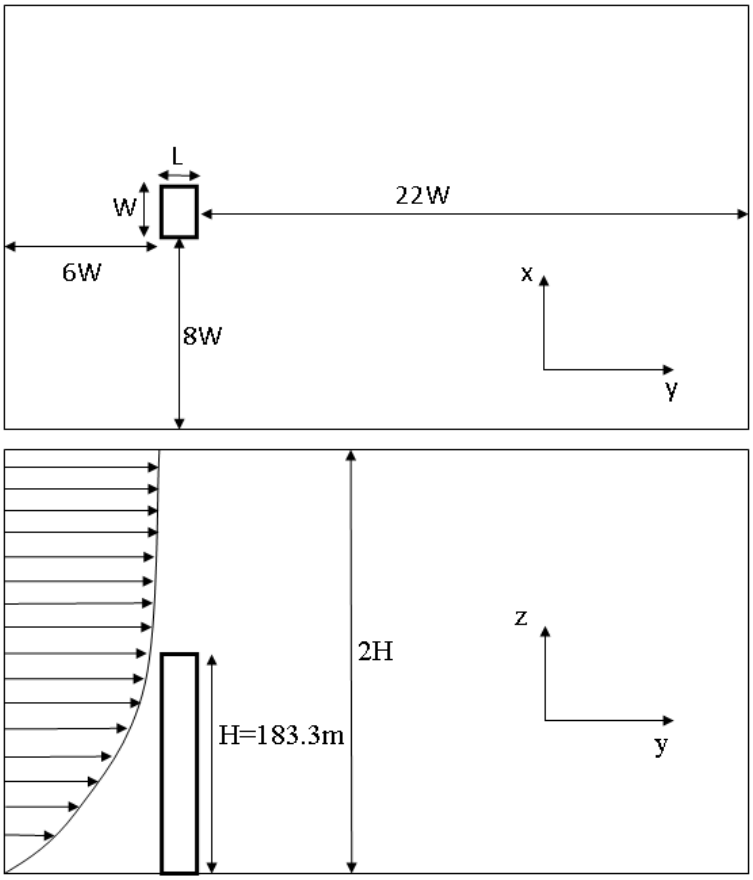


Figure 5.65 – Geometry of the CAARC standard tall building test case. Top View (up) and Side View (down).

In figures 5.66 and 5.67 the computational mesh generated for this case is presented. As the flow is three-dimensional with unsteady nature, no symmetry boundary condition could be applied and the complete geometry had to be included in the mesh. In order to keep computation time to a minimum and considering the amount of real time iterations that the simulation would have to complete, a rather coarse mesh was generated, consisting of *434,082* nodes, *623,801* tetrahedra and *640,685* prisms. As the base surface of the model constitutes the ground around the building, solid wall boundary conditions had to be applied there as well. Therefore, in order to accurately calculate the boundary layer effect at the ground around the building, as well as on the building surfaces, the inflation layer of prisms constituted of 20 layers, with the first layer height equal to *W/2500* and a growth factor of *1.2*. Details of the mesh inflation layer can be seen in Figure 5.67.

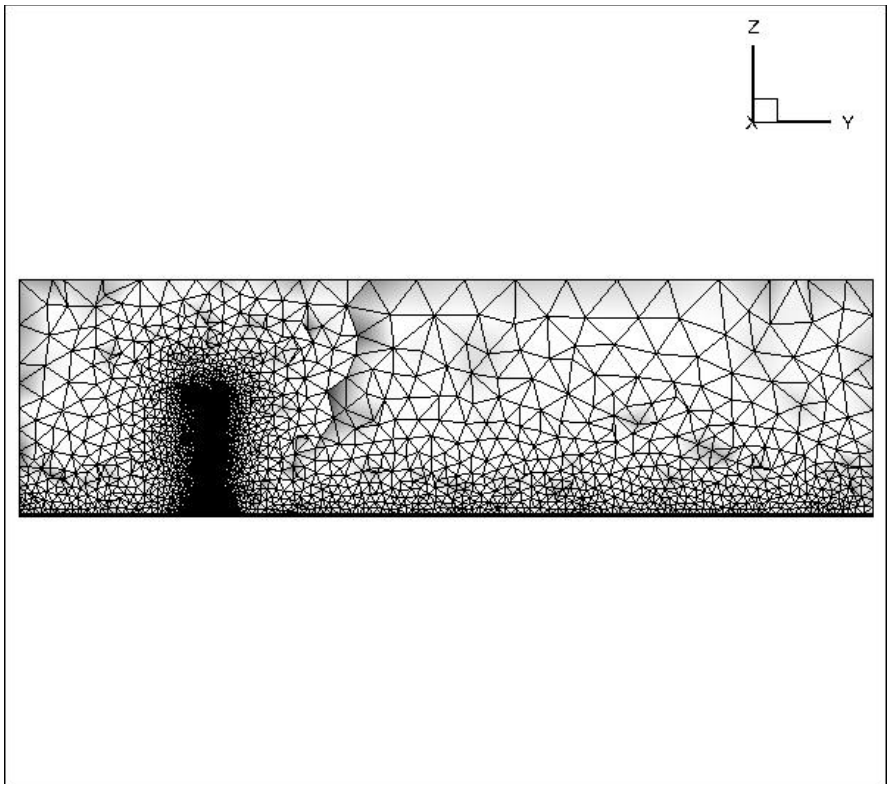


Figure 5.66 – Far view of the computational mesh for the flow around the CAARC standard tall building test case.

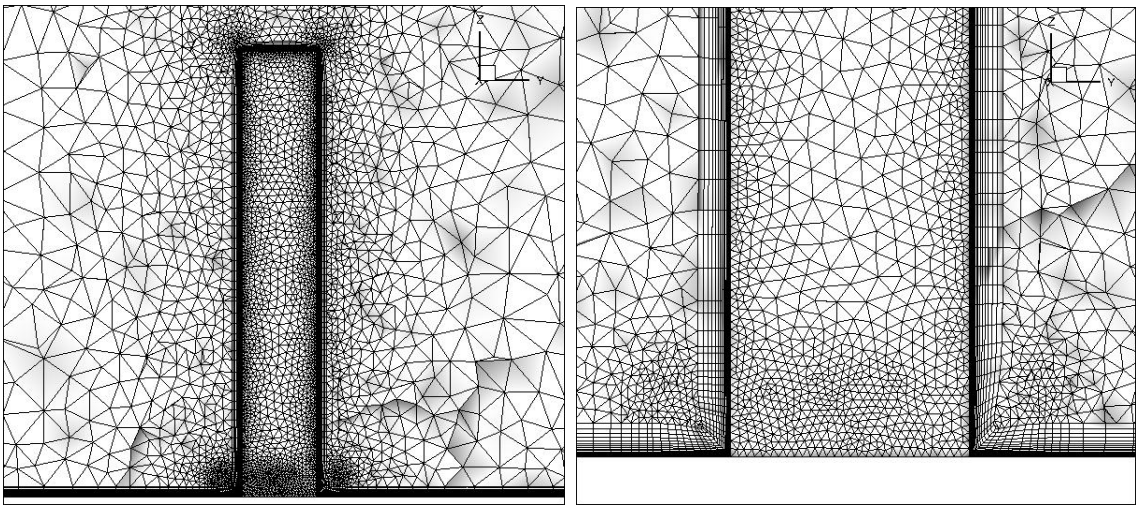


Figure 5.67 – Computational mesh around the CAARC tall building geometry (left) and detail of inflation layers at the base of the building (right).

In order to simulate wind behavior at such a low height (*183m*), the wind at the inlet had to be specially defined. As it is proposed by Huang et. al [Hua07] there are two kinds of expressions to define the velocity profile at the atmospheric boundary layer; one using a power law and another utilizing a logarithmic law. In this case the first was implemented, where the inlet velocity at each height was calculated using the following formula.

$$\frac{|\vec{c}|}{c_H} = \left(\frac{z}{z_H}\right)^{a} \tag{5.2}$$

where $c_H = 12.7\ m/sec$ is the value of air velocity at a height equal to the building height, z_H is the building height and α is a coefficient that in this case is equal to *0.3*. The inlet velocity is calculated on each node of the computational domain and is implicitly imposed as boundary condition through the characteristics based boundary conditions scheme. Another important aspect of the wind inflow is the turbulence intensity profile and its significance for the accuracy of the simulation is often mentioned in the open literature [Hua07] [Bra09] [Dag09]. Turbulence intensity can be described as the level of turbulence and is defined as:

$$I = \frac{\sqrt{\overline{U}^2}}{U} \tag{5.3}$$

where $\sqrt{\overline{U}^2}$ is the root mean square of the turbulent velocity fluctuations and U is the mean velocity. The turbulent kinetic energy can be calculated from the turbulence intensity as [Hua07]:

$$k = \frac{3}{2}(U \cdot I)^2 \tag{5.4}$$

while the dissipation rate ω can be calculated as a function of k as [FLU6.3]:

$$\omega = \frac{\sqrt{k}}{\sqrt{C_\mu} \cdot l} \tag{5.5}$$

where C_μ is an empirical constant specified in the turbulence model, approximately equal to *0.09* and l is the turbulence integral length scale, which in this case is equal to *0.58m* as it was measured at the model height H in the wind tunnel tests performed by Huang et al. [Hua07]. While for the velocity profile equation 5.2 was implemented in the code, the velocity intensity profile values at the inlet were interpolated from the wind tunnel test data reported by the same team [Hua07]. In Figure 5.68 the velocity profile at the inlet, as well as the turbulence intensity profile are presented. At the top, left and right side of the model free-slip boundary conditions were imposed.

The Reynolds number for this case was equal to *380,000* and was measured based on the building height (H) and the velocity at the same position (U_H). The non-dimensional variables used in this code were defined with the use of these values. The artificial compressibility parameter was set equal to *10.0*, while the CFL number was equal to *0.5*. Computation was performed on a workstation with an AMD FX™ 8150 Eight-Core processor at 3.62 GHz, with a partitioning of the original domain into six smaller sub-domains for parallel processing. Due to the small size of the computational mesh it was deemed unnecessary to use the multigrid method as well. The unsteady simulation was performed for 1300 real time iterations in order to obtain the time averaged results. The real time step was equal to $\Delta t=0.05$ and in each time step 50 pseudo-time iterations were performed, except for the initial real time iteration, where totally 3000 pseudo-time iterations were needed for the simulation to achieve a good initial condition. A summary of the simulation parameters is presented in Table 5.14.

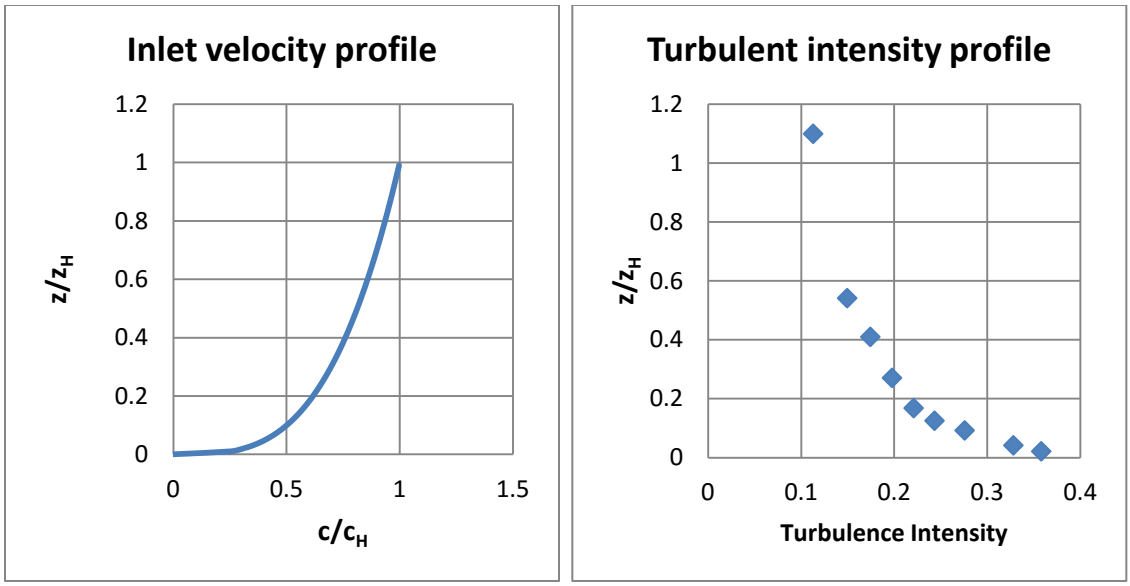


Figure 5.68 – Velocity profile (left) and turbulence intensity profile (right) at the inlet of the CAARC tall building model.

Table 5.14 – Simulation parameters for the unsteady turbulent flow around the CAARC tall building.

Parameters	
Type of flow	Viscous Turbulent, Unsteady
Reynolds number	380,000
Angle of attack	0^o
Grid density	434,082 nodes 623,801 tetrahedrons 640,685 prisms
Artificial compressibility parameter *β*.	10.0
Number of partitions	6
CFL	0.5
Real time step	0.05
Computer system	Workstation with AMD FX™ 8150 Eight-Core processor at 3.62 Ghz

The mean pressure coefficient on the building wall at the height of *z=2/3H* is presented in Figure 5.69 and is compared to corresponding results found in [Hua07] [Dag09]. The pressure coefficient diagram can be separated into three regions. Values at *x\L = [0, 1.5]* represent the windward side of the building, values at *x\L = [1.5, 2.5]* represent the parallel to the flow side of the building and values at *x\L = [2.5, 4]* represent the leeward side of the building. As it is obvious, the pressure distribution at the front wall of the building agrees very well with the experimental results, as well as the numerical ones. At the side walls there is good agreement with the numerical results by Huang et al. [Hua07] while it differs from the numerical results by Dagnew et al. [Dag09] mainly in magnitude. However there is a significant discrepancy compared to the experimental results by both researchers. These discrepancies are observed in

their reports as well and are attributed to the selection of the turbulence model and on differences at the turbulence conditions set at the inflow in each case. At the leeward side of the building the pressure distribution falls at the same range as the reference results. Slight discrepancies can be observed in the magnitude of the pressure distribution; however results from the wind tunnel experiments performed by Huang et al. are in very good agreement with the results of the current code.

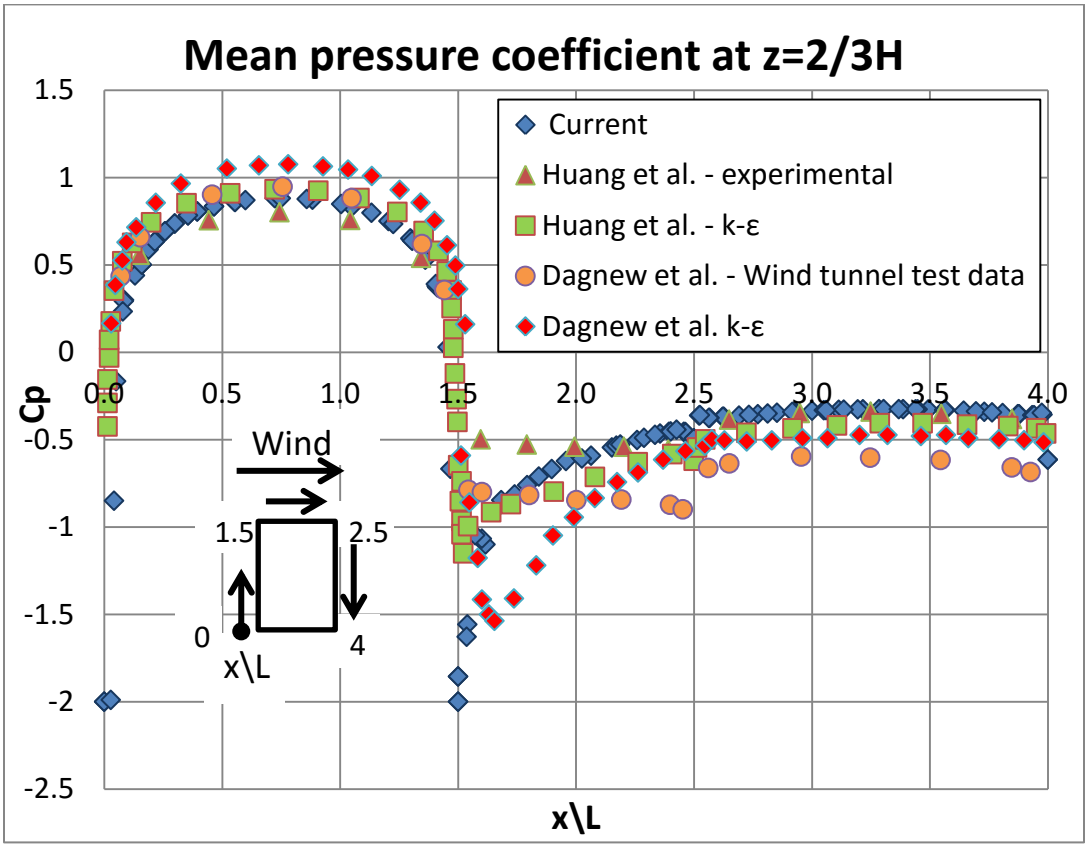


Figure 5.69 – Mean pressure coefficient distribution at z=2/3H of the CAARC standard tall building.

In figures 5.70 and 5.71 the pressure coefficient contours on the windward and leeward wall of the CAARC building are presented, respectively. On the windward side it is obvious that maximum pressure effect is achieved near the top of the building, where the velocity profile reaches a maximum, while at the base of the building an increase in the pressure coefficient along with its decrease at the sides of the building due to flow separation designates the development of a horseshoe vortex as is typical with similar flows around blunt bodies with sharp edges [Hus96] [Iac03]. At the sides of the building separation of the flow leads to low values of the pressure coefficient, while at the leeward side of the building similar values designate the area where the major vortices appear.

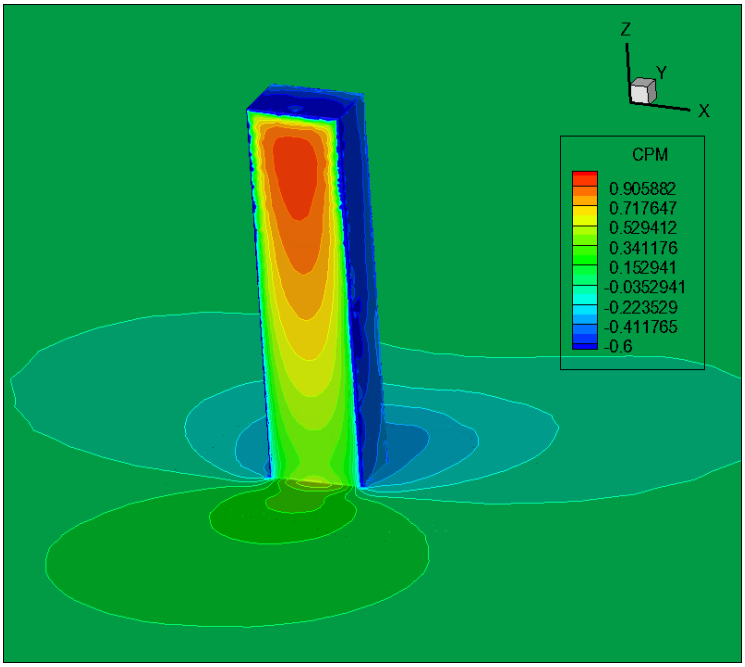


Figure 5.70 – Mean pressure coefficient contours on the windward side of the building.

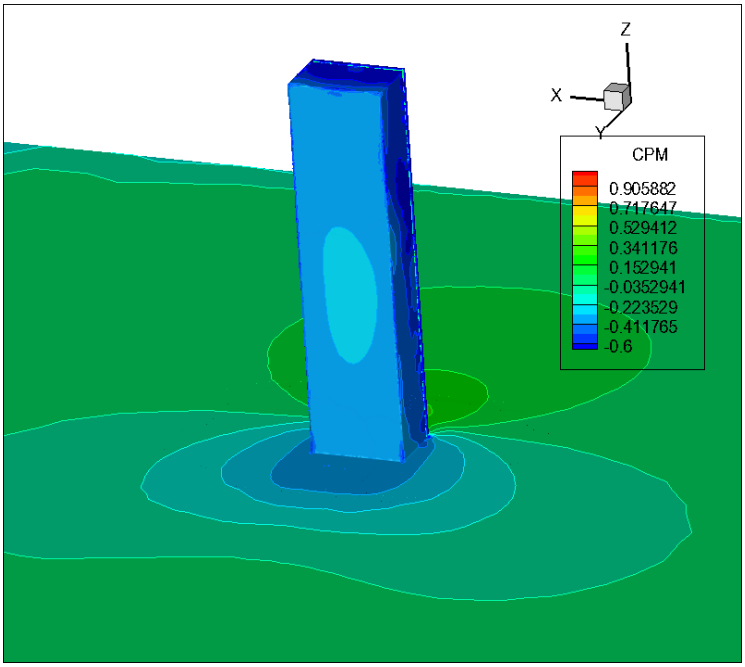


Figure 5.71 – Mean pressure coefficient contours on the leeward side of the building.

In figure 5.72 the pressure and velocity contours at the mid-plane of the CAARC standard tall building are presented. As it was expected, high pressure appears at the windward side near the top of the building, where wind velocity reaches its maximum value, while there is underpressure due to the building geometry at its leeward side. Furthermore, the velocity contours clearly reveal the variations induced by the velocity profile at the inlet, while at the leeward side of the building the streamwise velocity has negative values, due to the recirculation phenomena developed at that

area. It can be observed that no flow obstacle effect is developed at the outlet, justifying the chosen length of the computational domain.

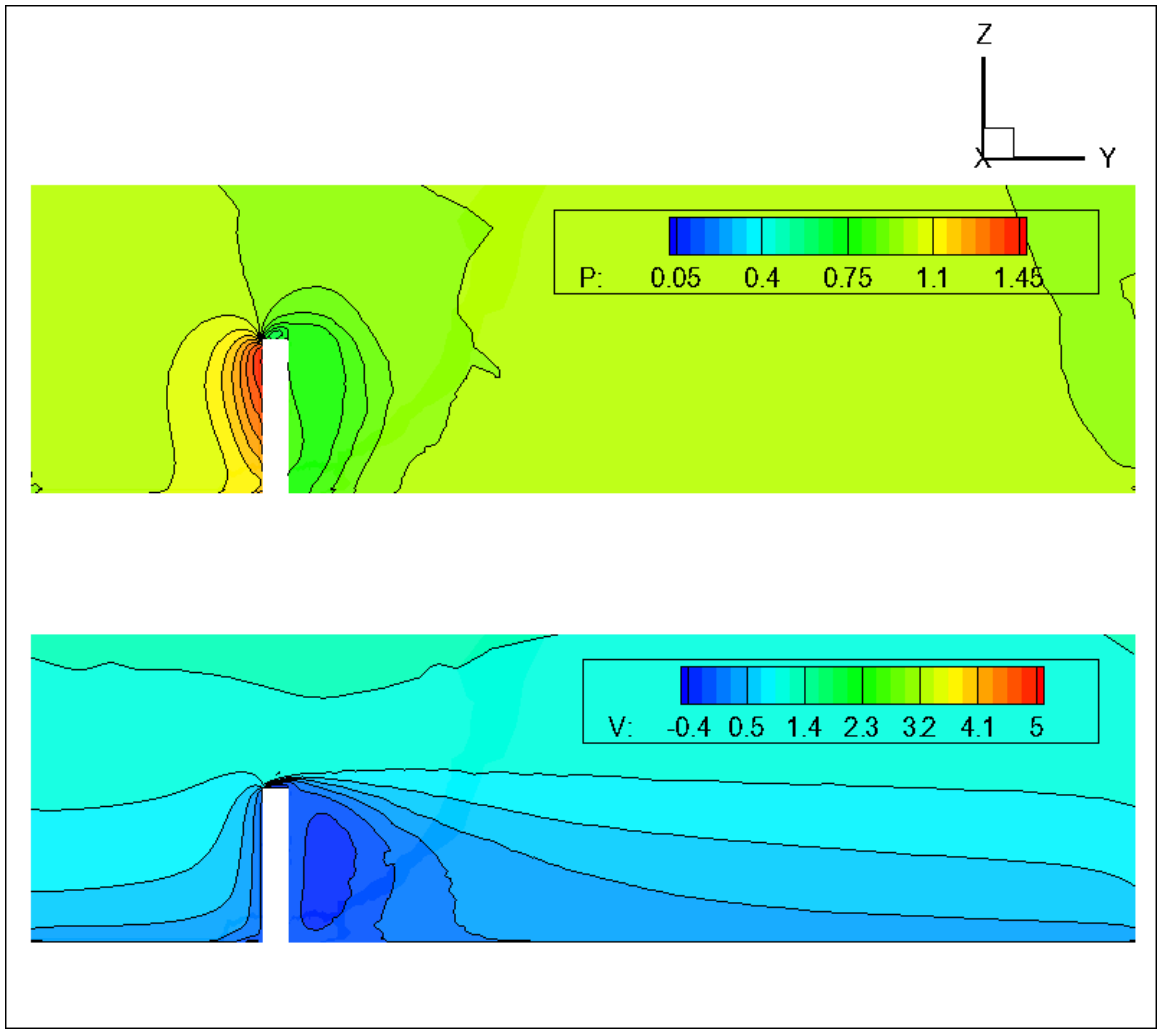


Figure 5.72 – Pressure contours (up) and streamwise velocity contours (down) at the mid-plane of the CAARC tall building case.

In figure 5.73 the pressure contours along with the velocity streamlines around the CAARC building are presented, where several features of the flow can be observed. At the base of the building at the windward side the horseshoe vortex that was described above is obvious. This vortex "hugs" the building and eventually dissipates at the leeward side due to viscosity. At the Y-Z plane a major vortex is developed near the top of the building with a secondary one at the base of it. These vortices are developed at the Y-X plane as a pair of twin symmetrical vortices at the leeward side of the building. Finally, at the walls of the building that are parallel to the flow, smaller vortices develop due to separation of the flow that is caused by the model's geometry.

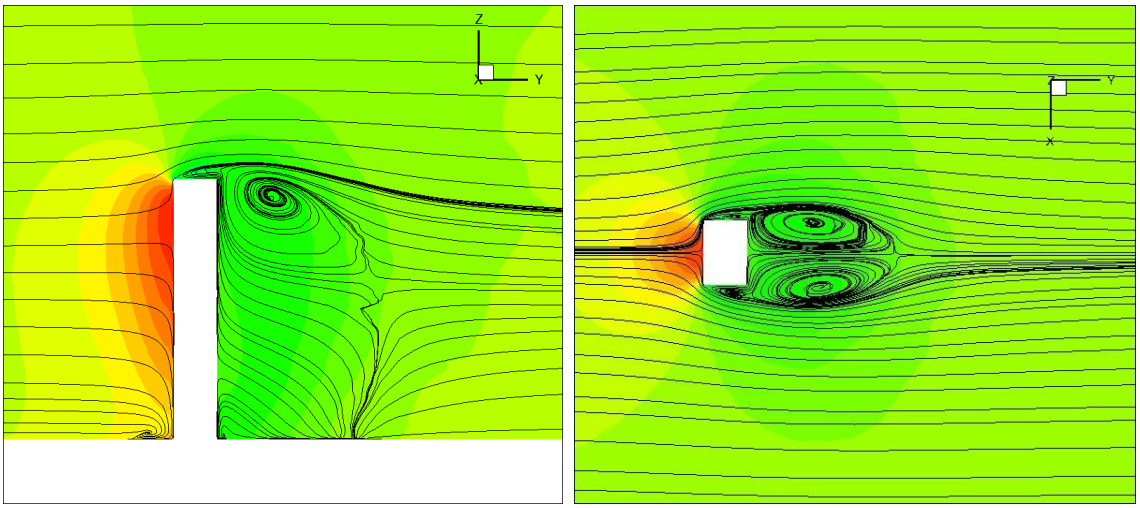


Figure 5.73 – Pressure contours and velocity streamlines at the mid-plane (left) and at height z=2/3H of the CAARC standard tall building.

5.2.4 Turbulent flow around the DLR-F11 model at 7^{o} angle of attack.

The final test case that was considered with the *Galatea-I* code concerns the flow around an aircraft geometry with a high lift configuration, as it would have been during landing. While the air is a compressible fluid, flows in such conditions are of low-Mach numbers (below *0.3*) and thus the flow is considered incompressible. This specific test case was designed by NASA for the purposes of the 2nd High Lift Prediction Workshop (HiLiftPW-2) that was held in June 2013 in San Diego, California [Cav14] [Chi14] [Del14] [Eli14] [Hof14] [Han14] [Lee15] [Mav15] [Mur15] [Rud14] [Rum14]. The geometry consists of the DLR-F11 model; a generic semi-span wing consisted of three elements (leading edge slat, wing, and trailing edge Fowler flap) with a body pod [Rum14]. The aircraft geometry is presented in Figure 5.74, while in Figure 5.75 the different elements of the aircraft model are described. The untwisted swept wing of the model had a semi-span equal to *1.4m* and an aspect ratio of *9.353*, while the aerodynamic chord was equal to *0.34709m*. The wing components were set in one particular landing configuration; the slat set at 26.5^{o} and the flap at 32^{o}, while both slat and flap were mounted on the wing with the appropriate slat tracks and flap tracks, respectively, collectively referred at as "support brackets" or "brackets". The model was tested both at low and high Reynolds numbers test wind tunnels, providing results on the forces, moments, and surface pressures for various angles of attack [Rud12].

The high Reynolds number flow conditions were *Re=15.1E+6, Mach=0.175,* T_{ref} *=114.0 K,* p_{ref} *= 295 kPa*, while the low Reynolds number flow conditions were *Re=1.35E+6, Mach=0.175,* T_{ref} *=298.6 K,* p_{ref} *= 100.7 kPa*. The Reynolds numbers were calculated with respect to the aerodynamic chord length. For the HiLiftPW-2 the geometry of three configurations of the DLR-F11 model were provided; Config 2 consisted of the wing-body-high lift system plus a side-of-body flap seal, Config 4 consisted of Config 2 along with the slat and flap support brackets, and Config 5 consisted of Config 4 as well as some slat pressure tube bundles. For each configuration

a set of coarse, medium, and fine grids were provided by NASA, while the participants were allowed to generate their own meshes.

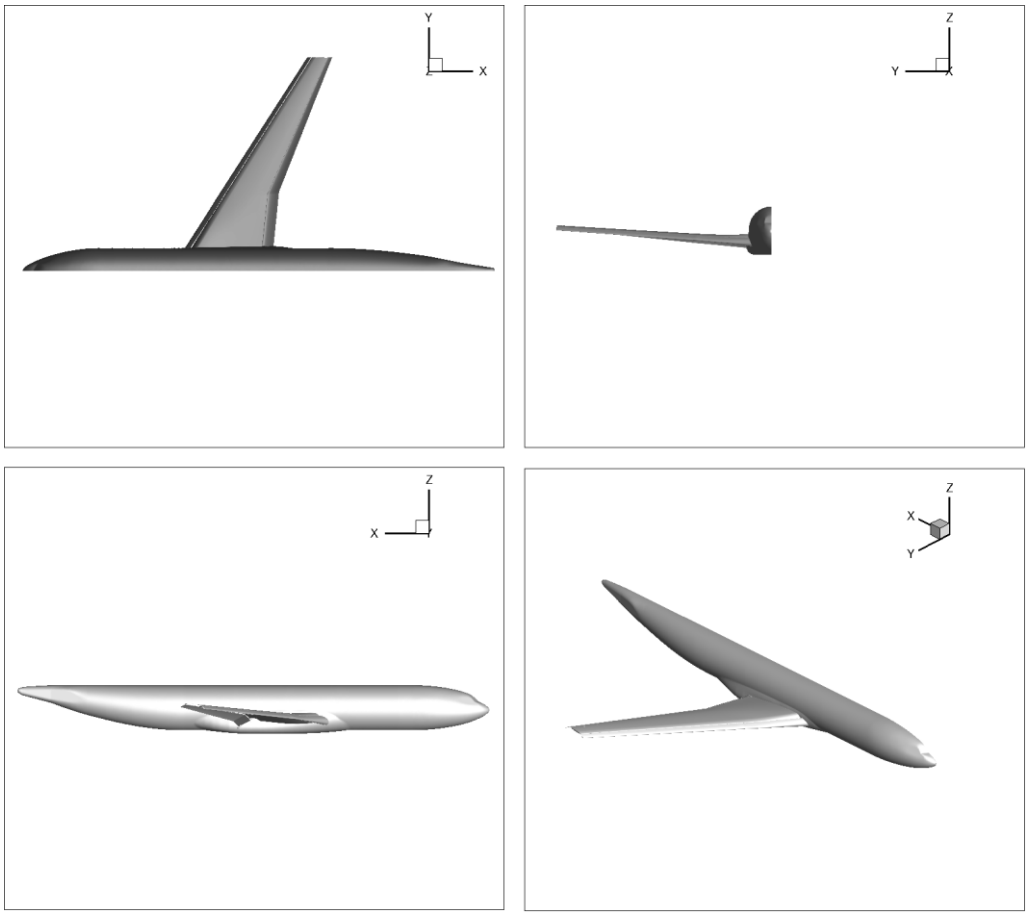


Figure 5.74 – Geometry of the DLR-F11 aircraft model.

Due to the size of the provided meshes and restrictions in computational resources, simulation was carried out on the more simple geometry; Config 2 was selected, where the wing consisted of the three components, without the support brackets, or the pressure tube bundles. Although as it was demonstrated by Chitale et al. [Chi14], these brackets alter the flow behavior, especially under the wing, simulation of the flow around the simpler configuration provided adequate results, capable of demonstrating the current code's capabilities. The simulation was performed for angles of attack equal to 7^o and 12^o. In this section the results for the 7^o angle of attack are presented. The utilized computational mesh was characterized as "medium" in density and consisted of *30,767,679* nodes, *58,300,006* tetrahedrons, *40,864,715* prisms and *275,227* pyramids. In Figure 5.76 the utilized computational grid is presented in whole, as well as a detail of the surface mesh at the aircraft region. Simulation was carried out on a Dell™ R815 Poweredge server with four AMD Opteron™ 6380 sixteen-core processors at 2.50GHz. The initial domain was divided into 32 sub-domains for parallel processing. Integration in pseudo-time was achieved with a CFL number equal to *0.25*, while the artificial compressibility

parameter was equal to *10.0*. Due to unsteadiness that occurred during the initial run of this test case, the flow was subsequently treated as unsteady in nature; the dimensionless real time step was set equal to *0.5* and the simulation was carried out for 43 real time iterations. A summary of the case parameters can be found in Table 5.15.

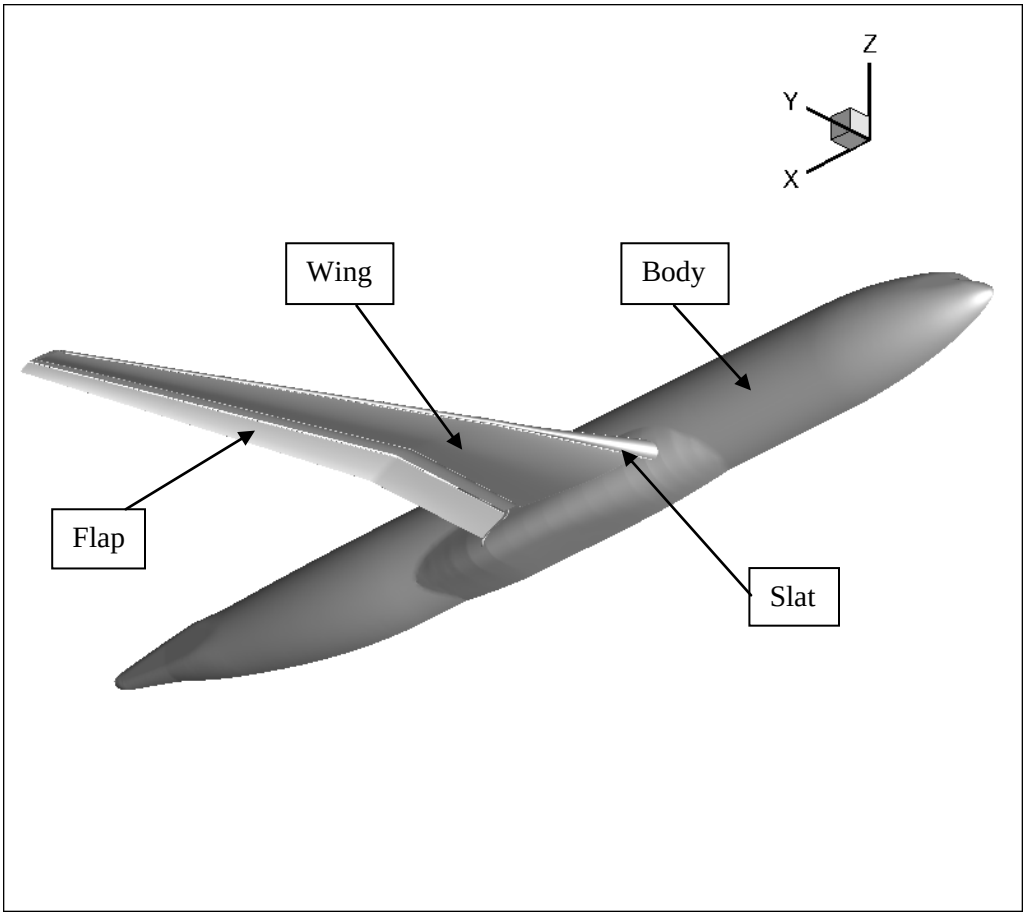


Figure 5.75 – Detailed view of the elements consisting the DLR F-11 aircraft model.

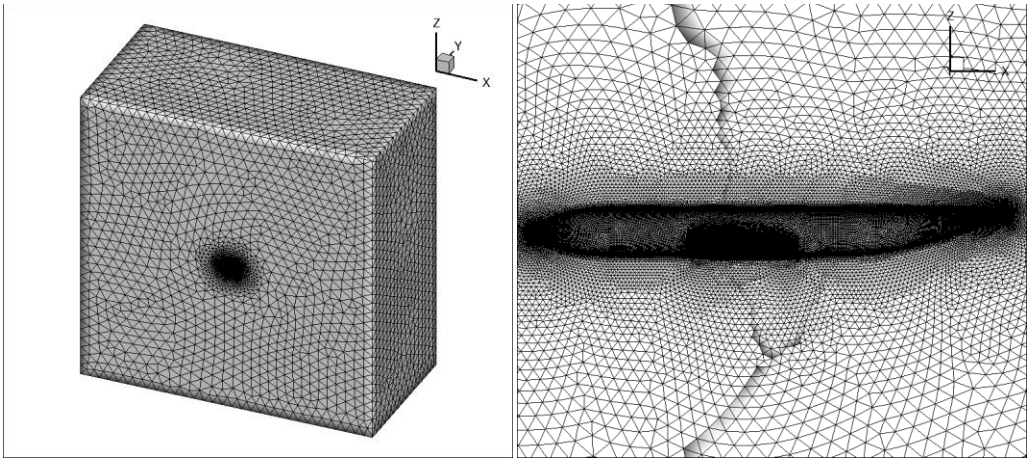


Figure 5.76 – Computational mesh for the simulation of the turbulent flow around the DLR-F11 model.

Table 5.15 – Simulation parameters for the turbulent flow around the DLR-F11 model (7° angle of attack).

Parameters	
Type of flow	Viscous Turbulent, Unsteady
Reynolds number	15.1E+6
Angle of attack	7°
Grid density	30,767,679 nodes 58,300,006 tetrahedrons 40,864,715 prisms 275,227 pyramids
Artificial compressibility parameter *β*.	10.0
Number of partitions	32
CFL	0.25
Real time step	0.5
Computer system	Dell™ R815 Poweredge server with four AMD Opteron™ 6380 sixteen-core processors at 2.50GHz

Quantitative evaluation of the test case results was performed by comparing with the experimental data of the pressure coefficient along several cross-sections of the wing, which were provided by the workshop organizers [Rud12]. Corresponding results from the current solver were extracted as the time-average results from the 43 real time iterations that have been performed. Figures 5.77 to 5.86 illustrate such comparisons. The pressure coefficient distribution on each figure is divided into three parts; the leftmost area represents the leading edge slat, the middle part is the wing area and the rightmost part is the pressure distribution around the trailing edge flap. Good agreement is found between current numerical and experimental data on most cross-sections. Some discrepancies can be observed at 89.1% and 96.5% of the wing span, which are located near the tip of the wing. These disagreements are especially located at the flap area, where flow separation phenomena are expected to occur, on a very small area. Refinement of the mesh in that specific area could produce better results, however the current discrepancies are considered of minor impact.

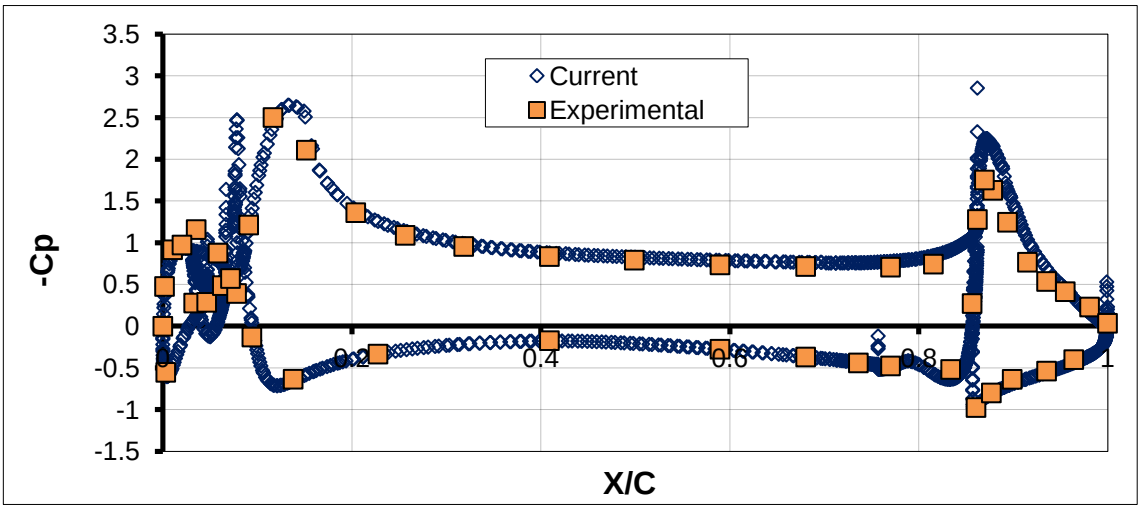


Figure 5.77 – Pressure coefficient distribution around the DLR-F11 wing at 15% wing span (7° attack angle).

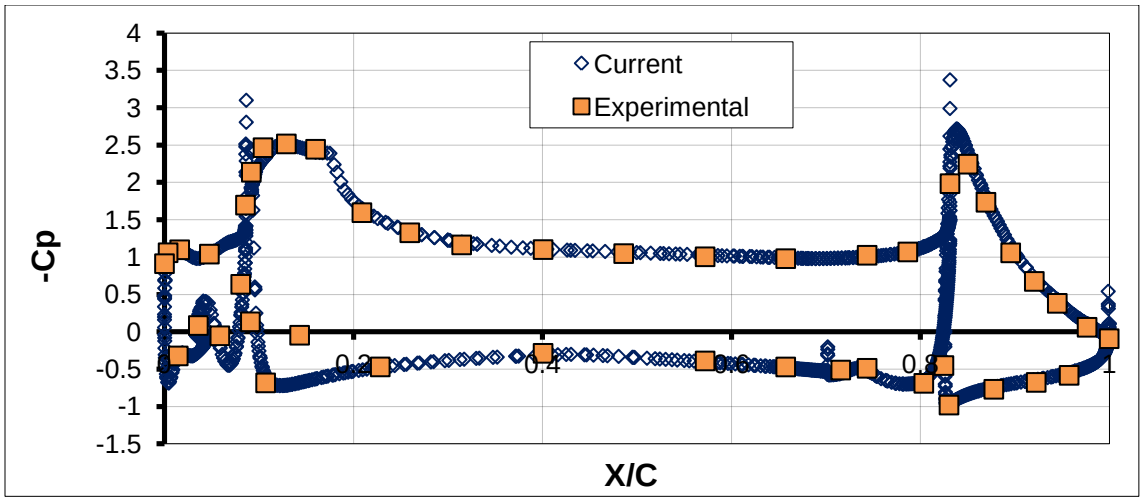


Figure 5.78 – Pressure coefficient distribution around the DLR-F11 wing at 28.8% wing span (7° attack angle).

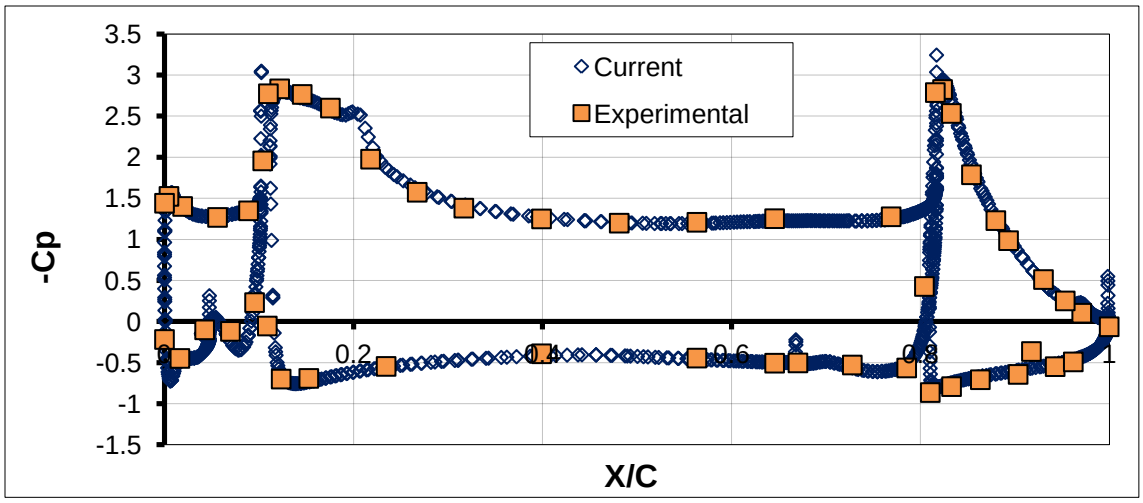


Figure 5.79 – Pressure coefficient distribution around the DLR-F11 wing at 44.9% wing span (7° attack angle).

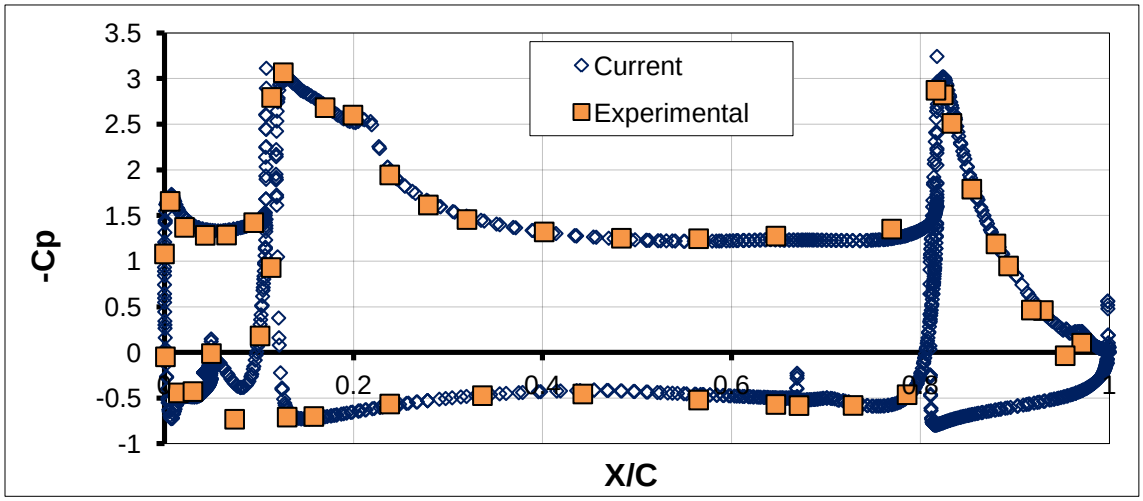


Figure 5.80 – Pressure coefficient distribution around the DLR-F11 wing at 54.3% wing span (7° attack angle).

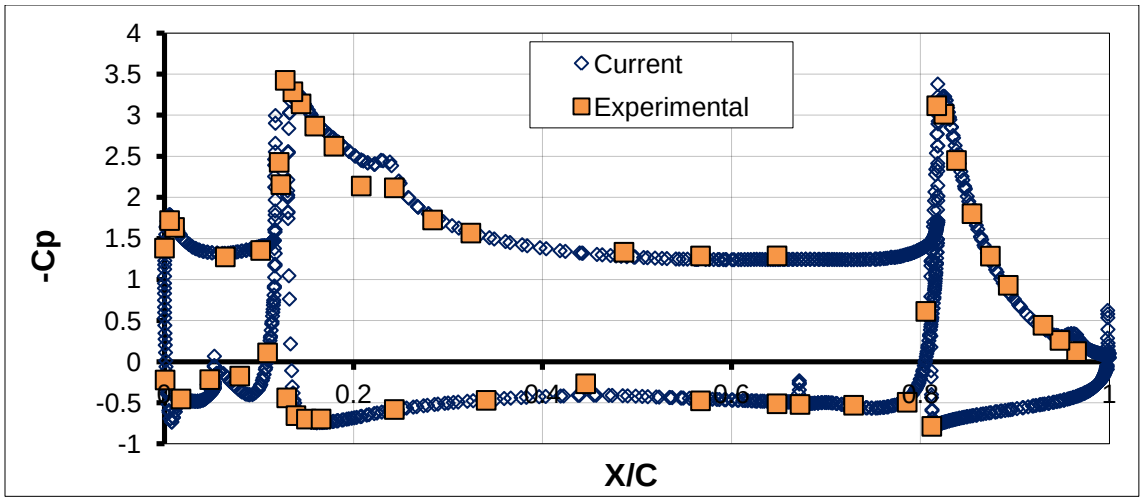


Figure 5.81 – Pressure coefficient distribution around the DLR-F11 wing at 68.1% wing span (7° attack angle).

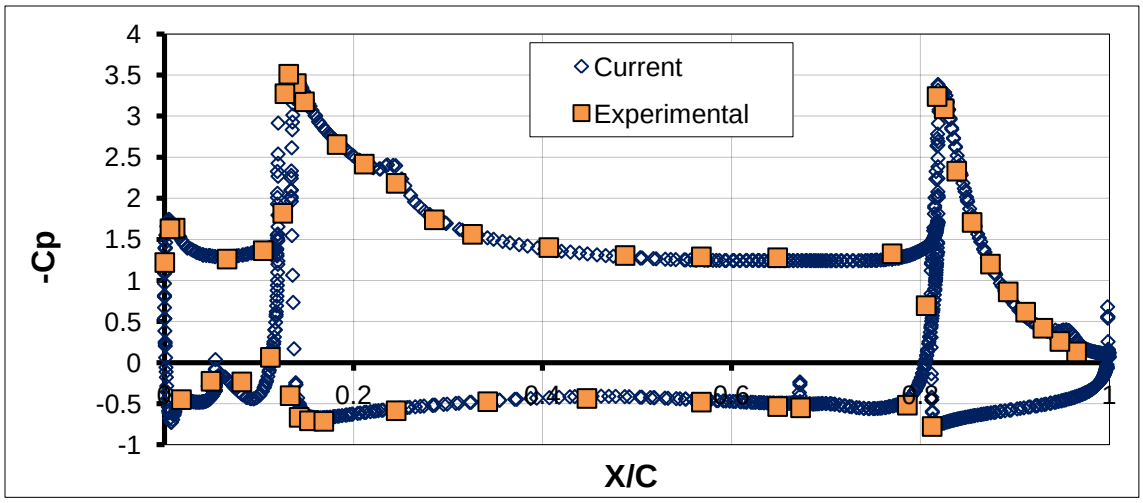


Figure 5.82 – Pressure coefficient distribution around the DLR-F11 wing at 71.5% wing span (7° attack angle).

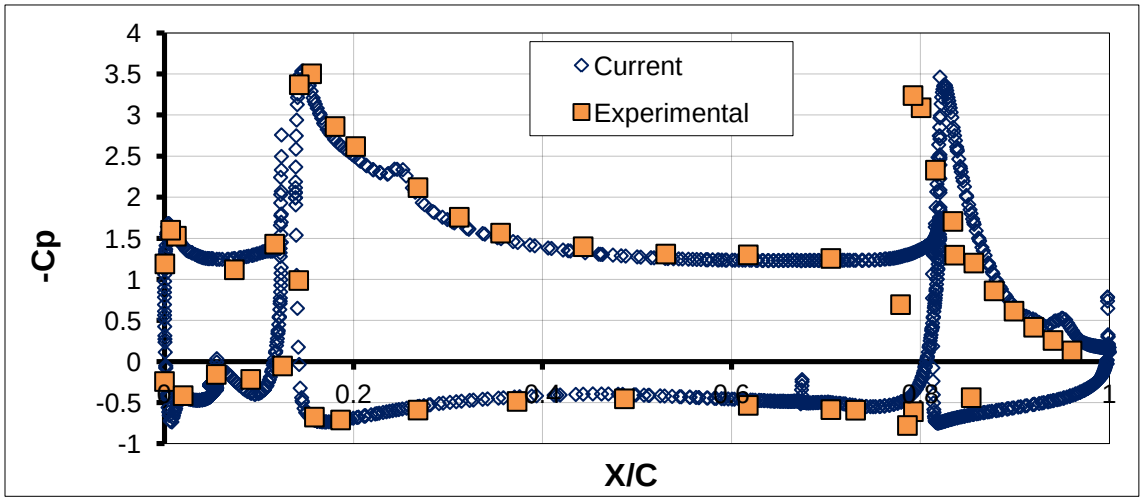


Figure 5.83 – Pressure coefficient distribution around the DLR-F11 wing at 75.1% wing span (7° attack angle).

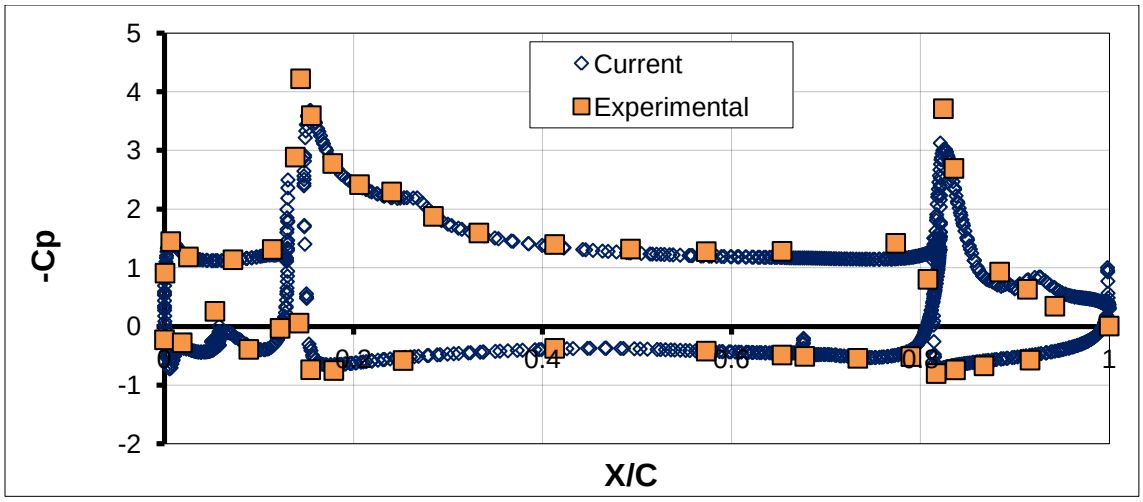


Figure 5.84 – Pressure coefficient distribution around the DLR-F11 wing at 81.8% wing span (7° attack angle).

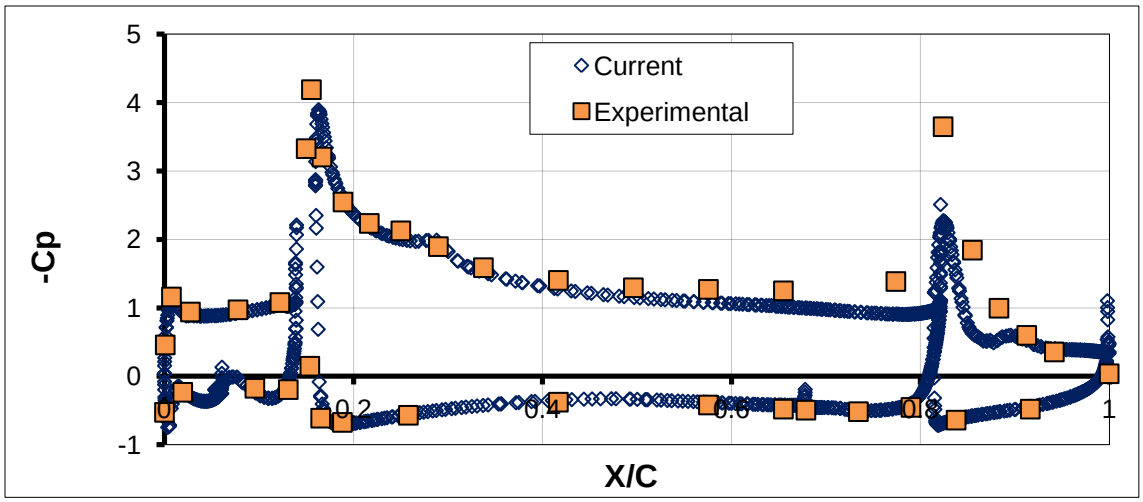


Figure 5.85 – Pressure coefficient distribution around the DLR-F11 wing at 89.1% wing span (7° attack angle).

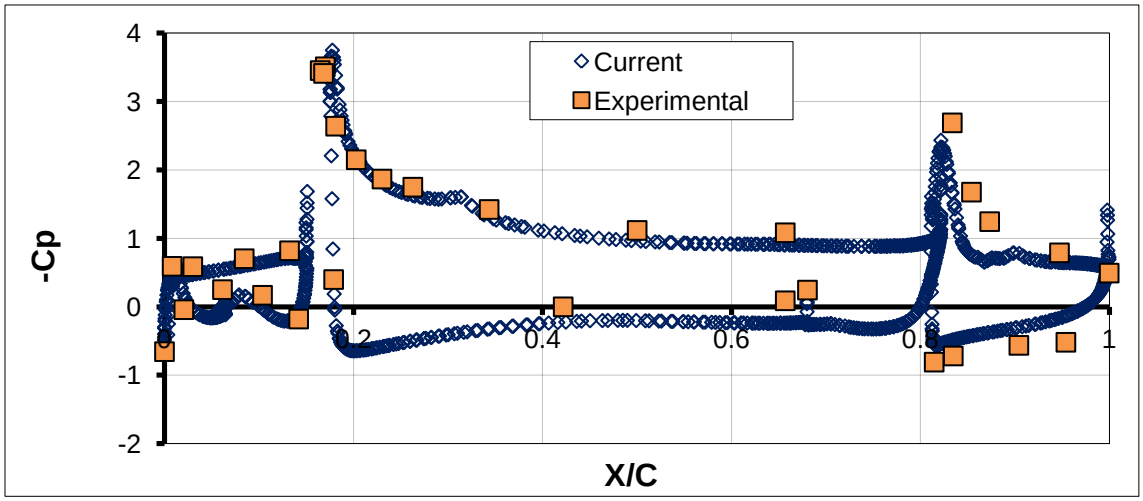


Figure 5.86 – Pressure coefficient distribution around the DLR-F11 wing at 96.4% wing span (7° attack angle).

In Figures 5.87 to 5.89 the pressure contours along with the streamlines around the DLR-F11 wing at various wing spans are illustrated. Various flow phenomena can be observed in those Figures. The flow comes at the wing at an attack angle of 7^{o}. The stagnation point at the leading edge slat is obvious in all three figures. Due to the geometry of the slat, the flow separates at its lower point and forms a vortex between the slat and the wing. The vortex is more visible near the aircraft body (15% wing span) and is carried along the pocket that is formed between the slat and the wing as it is obvious in all the figures. A similar vortex is formed at the rear of the wing, at the area between the wing and the trailing edge flap. Both vortices are the reason for the unsteadiness that was observed during the simulation. The streamlines envelop all three components of the wing and a thick boundary layer is formed at the leeward side of the wing, as it can be observed in Figure 5.90, where the turbulent kinematic viscosity contours at 15% wing span are presented. Finally, there can be observed a streamline curvature at the leeward side of the flap area, due to the structure of the geometry. However, this curvature is not visible at 96.4% wing span, where the flow is separated at the flap region, due to proximity to the wing tip. An overall picture of the pressure contours on the DLR-F11 model surfaces is presented in Figure 5.91.

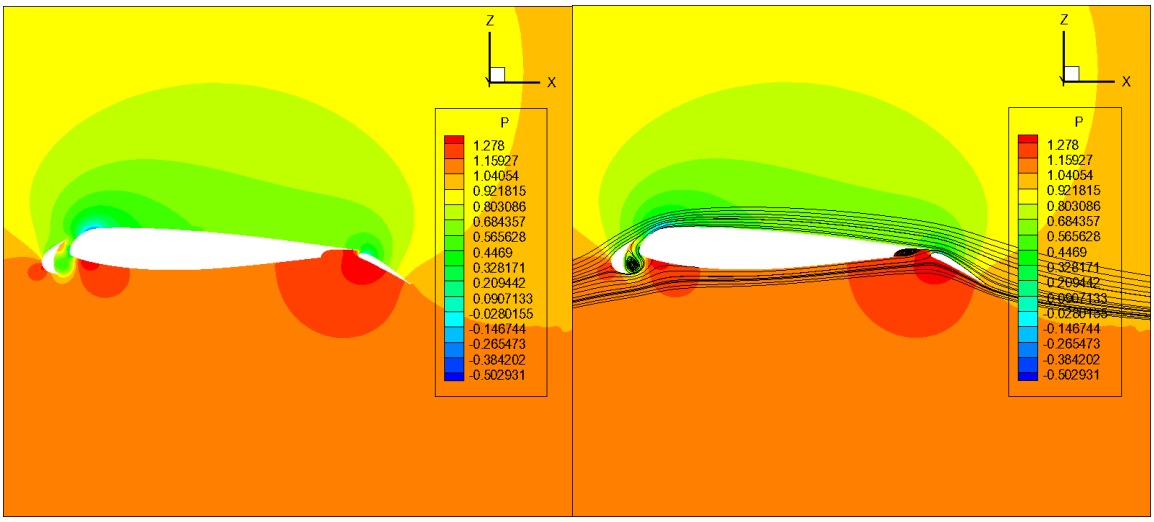


Figure 5.87 –Pressure contours (left) and velocity streamlines (right) around DLR-F11 wing at 15% wing span.

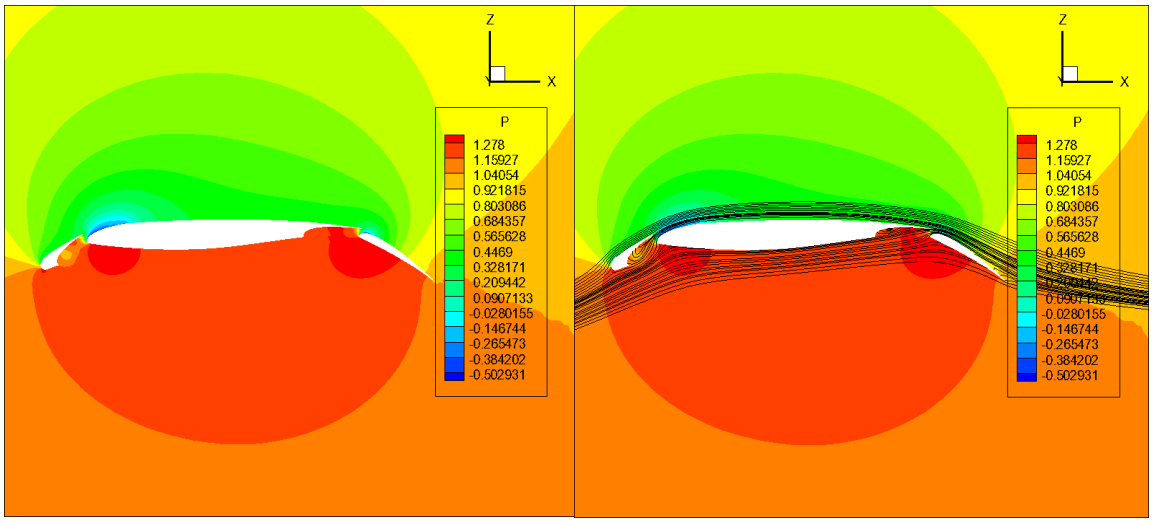


Figure 5.88 –Pressure contours (left) and velocity streamlines (right) around DLR-F11 wing at 54.3% wing span.

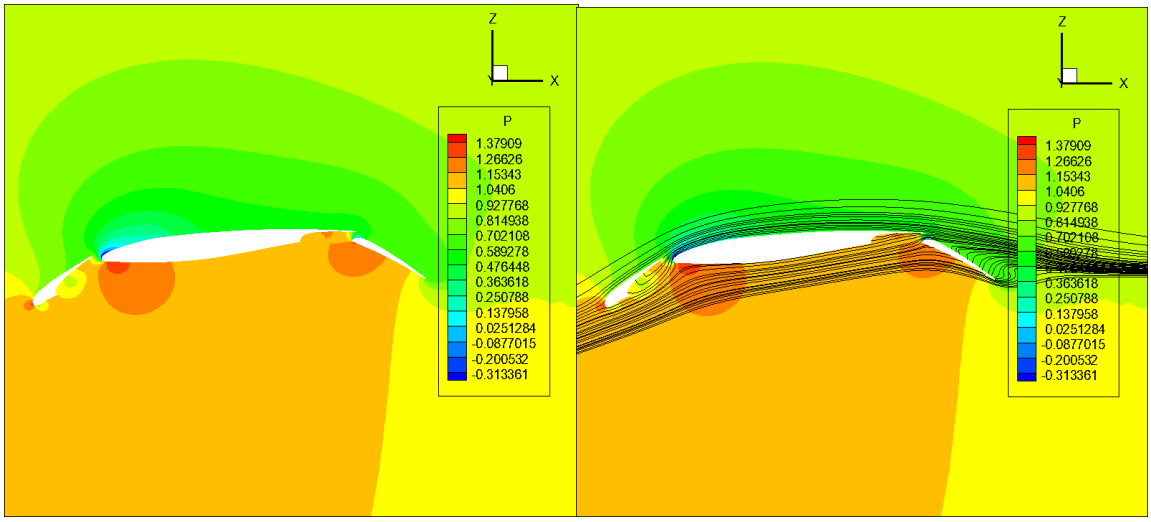


Figure 5.89 –Pressure contours (left) and velocity streamlines (right) around DLR-F11 wing at 96.4% wing span.

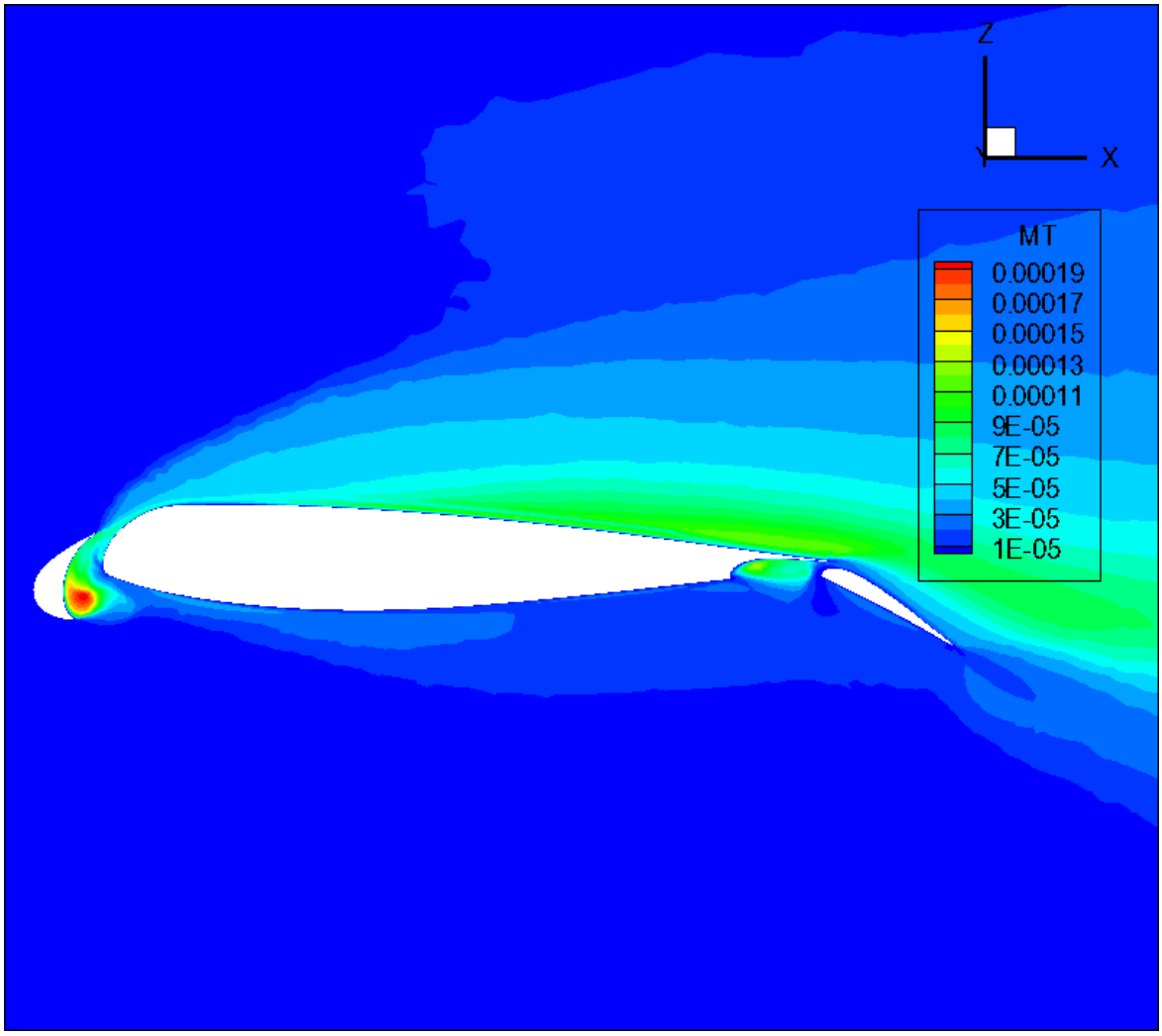


Figure 5.90 –Turbulent kinematic viscosity contours around DLR-F11 wing at 15% wing span (7^o attack angle).

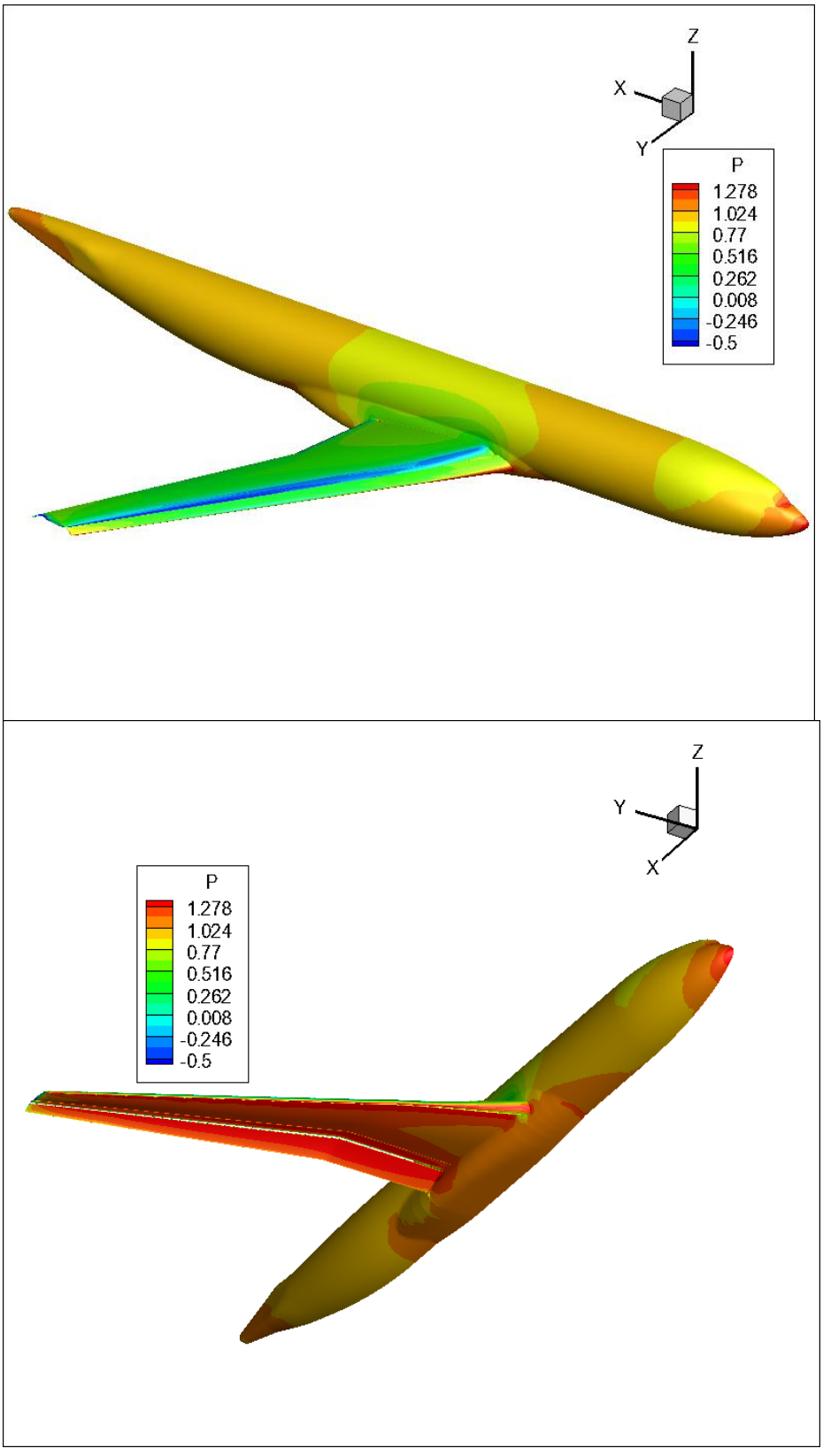


Figure 5.91 – Pressure contours on the DLR-F11 aircraft surfaces (7^{o} angle of attack).

In Figure 5.92 the vorticity contours at the wake of the DLR-F11 wing were sketched on successive X-Y parallel planes, in order to picture the wingtip vortex that is generated in that area. The computed results are compared qualitatively to those from Chitale et al. [Chi14], who published similar illustrations of the vorticity contours at the same region. As it can be observed, the vorticity contours predicted with the *Galatea-I* software are of slightly better quality than the ones produced with the Medium mesh (approx. *17.34* million nodes) of Chitale et al., while they are worse than the ones produced with their Fine mesh (approx. *112.96* million nodes). A view of the same vorticity contours observed from top-rear side of the aircraft is illustrated in Figure 5.93.

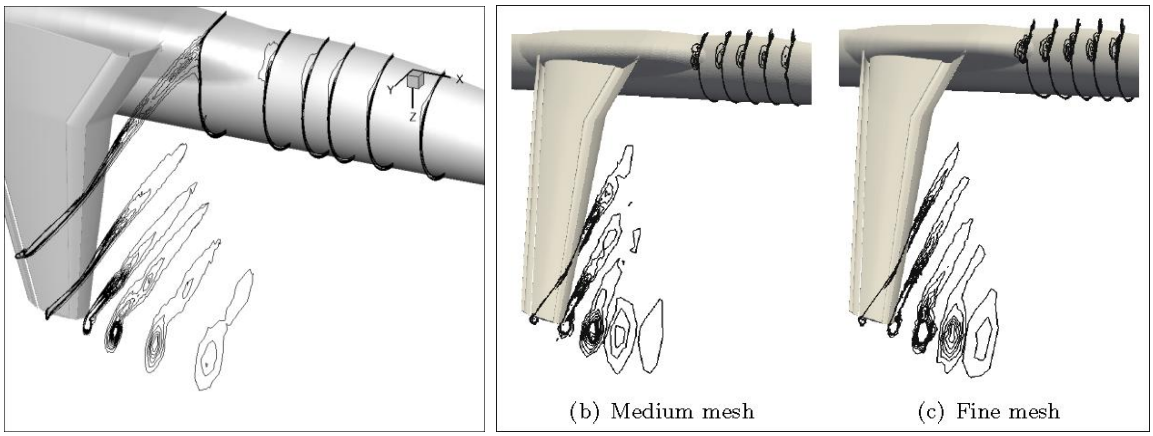


Figure 5.92 – Vorticity contours on Y-Z planes at the wake of the DLR-F11 wing (left) and qualitative comparison with reference numerical data (right) [Chi4].

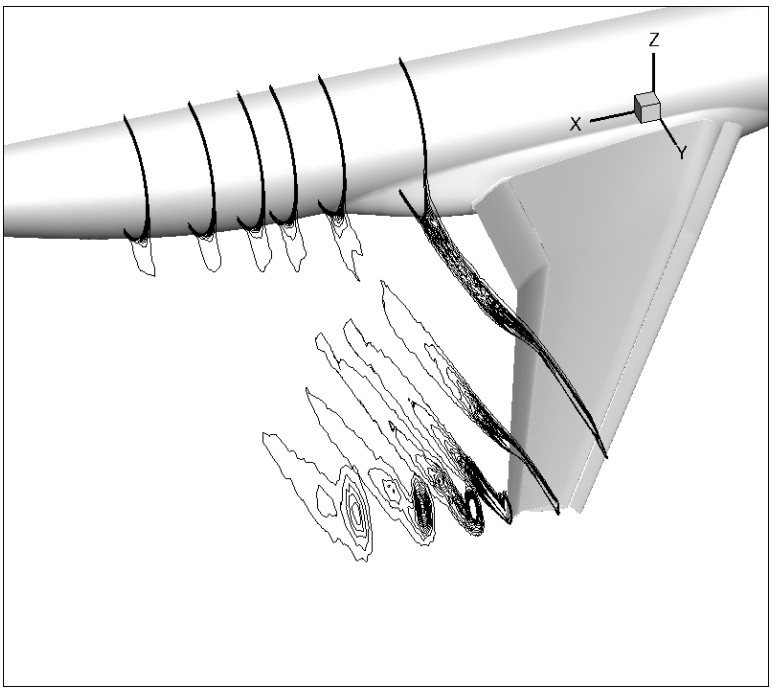


Figure 5.93 – Vorticity contours on Y-Z planes at the wake of the DLR-F11 wing observed from the top-rear side of the aircraft.

5.2.5 Turbulent flow around the DLR-F11 model at 12° angle of attack.

With the results from the previous section as initial state of the flow the mesh for the DLR-F11 model was used for the simulation of turbulent flow around it with a 12° angle of attack. The flow conditions were the same as with the case with 7° angle of attack, as well as the simulation parameters. The case was run on a Dell™ R815 Poweredge server with four AMD Opteron™ 6380 sixteen-core processors at 2.50GHz. The comparable results were computed as the time-average of 43 real time iterations. The simulation parameters for this case are presented in Table 5.16.

Table 5.16 – Simulation parameters for the unsteady flow around the DLR-F11 model (12° angle of attack).

Parameters	
Type of flow	Viscous Turbulent, Unsteady
Reynolds number	15.1E+6
Angle of attack	12°
Grid density	30,767,679 nodes 40,864,715 prisms 58,300,006 tetrahedrons 275,227 pyramids
Artificial compressibility parameter *β*.	10.0
Number of partitions	32
CFL	0.25
Real time step	0.5
Computer system	Dell™ R815 Poweredge server with four AMD Opteron™ 6380 sixteen-core processors at 2.50GHz

For the quantitative evaluation of the simulation the pressure coefficient around the wing of the DLR-F11 model was calculated at various wing spans. The results are compared to the experimental ones provided for the HiLiftPW-2 [Rud12] and presented in Figures 5.94 to 5.103. As it can be observed, the overall quality of the results is very good, as there is a satisfactory agreement with the experimental data at all the wing locations. Some discrepancies can be found in Figures 5.102 and 5.103, which are located near the wing tip. These disagreements were also observed in the case with 7° angle of attack and can be attributed to the flow separation phenomena occurring in that region, especially at the area of the flap.

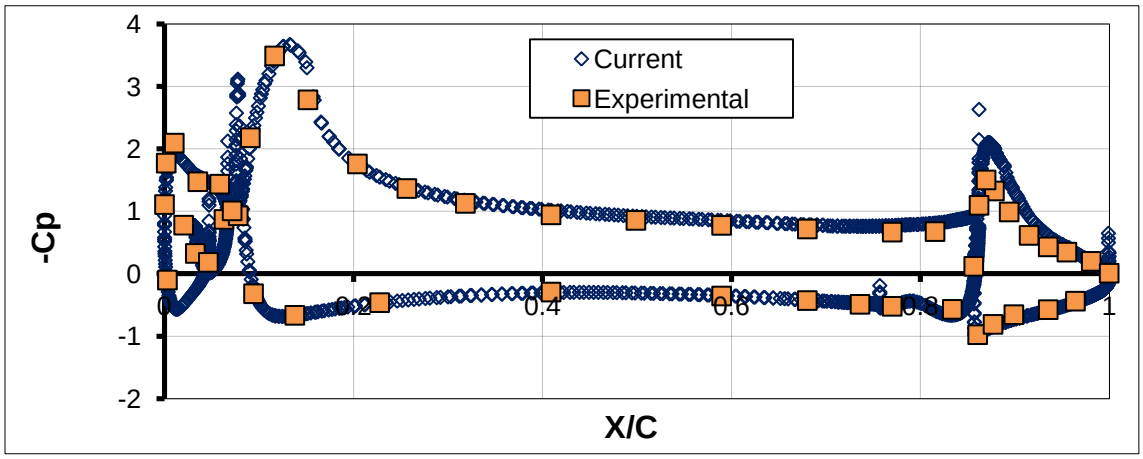


Figure 5.94 – Pressure coefficient distribution around the DLR-F11 wing at 15% wing span (12° attack angle).

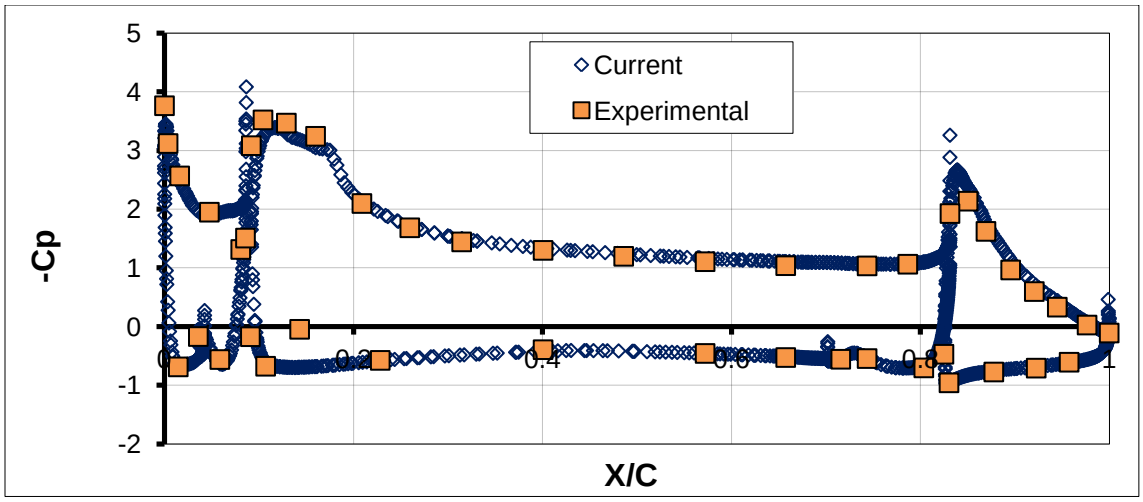


Figure 5.95 – Pressure coefficient distribution around the DLR-F11 wing at 28.8% wing span (12º attack angle).

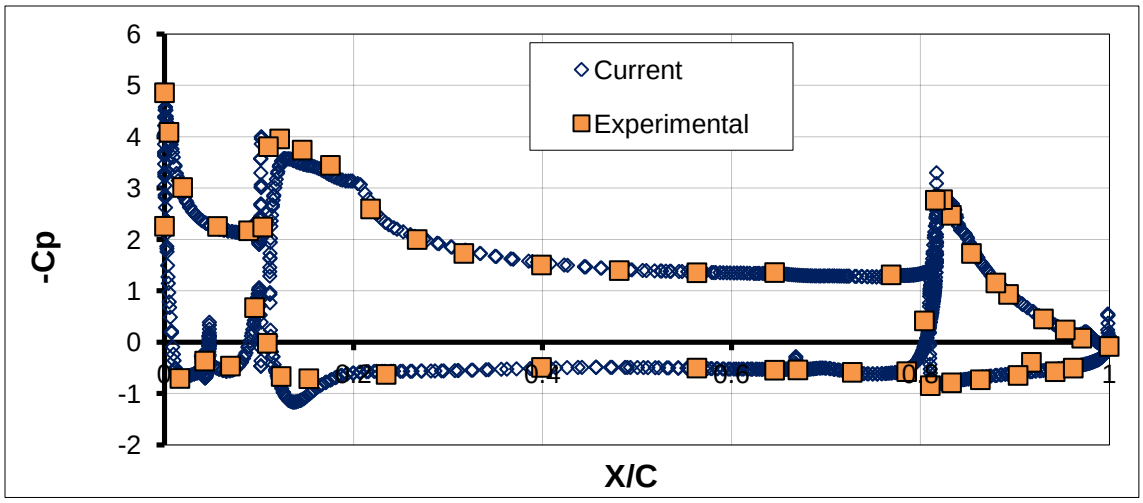


Figure 5.96 – Pressure coefficient distribution around the DLR-F11 wing at 44.9% wing span (12º attack angle).

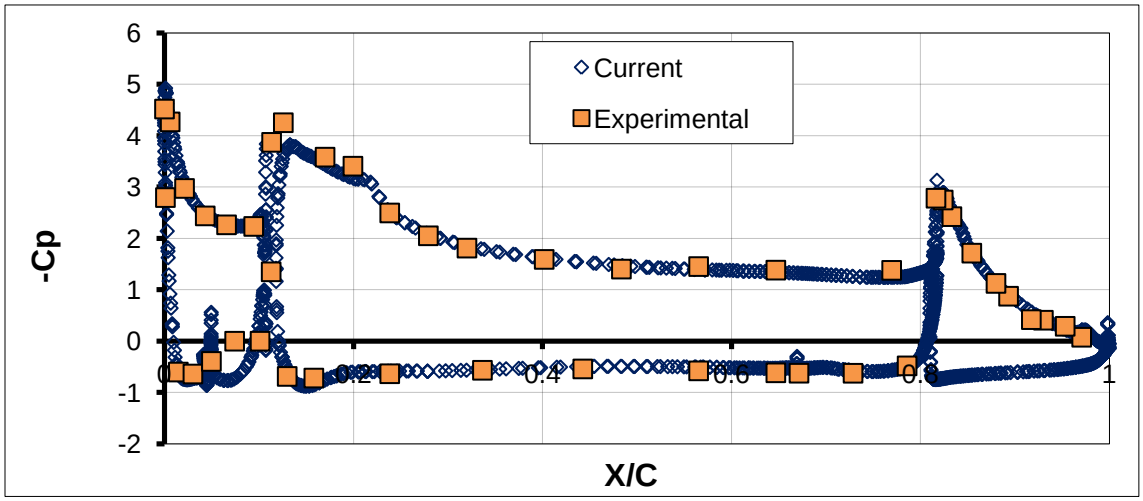


Figure 5.97 – Pressure coefficient distribution around the DLR-F11 wing at 54.3% wing span (12º attack angle).

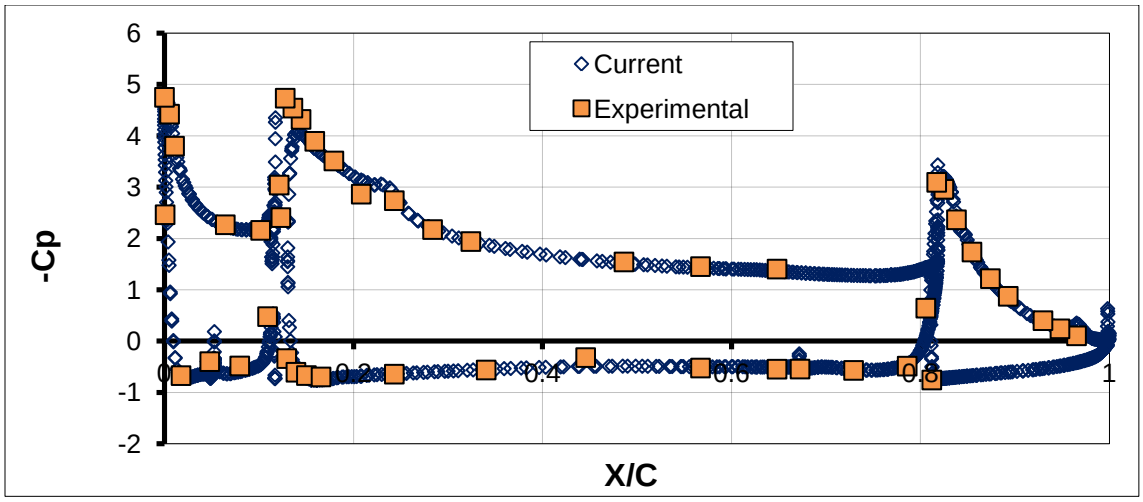


Figure 5.98 – Pressure coefficient distribution around the DLR-F11 wing at 68.1% wing span (12° attack angle).

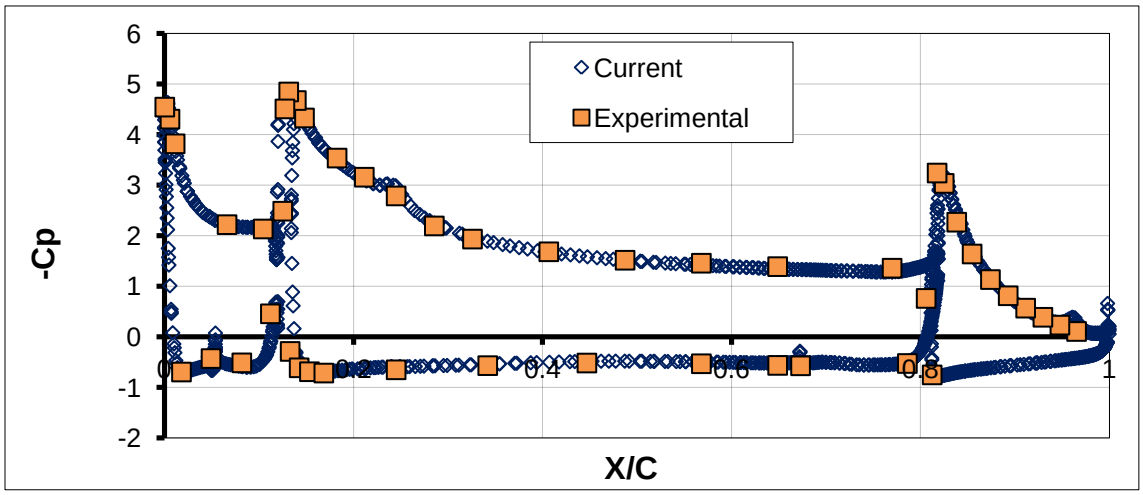


Figure 5.99 – Pressure coefficient distribution around the DLR-F11 wing at 71.5% wing span (12° attack angle).

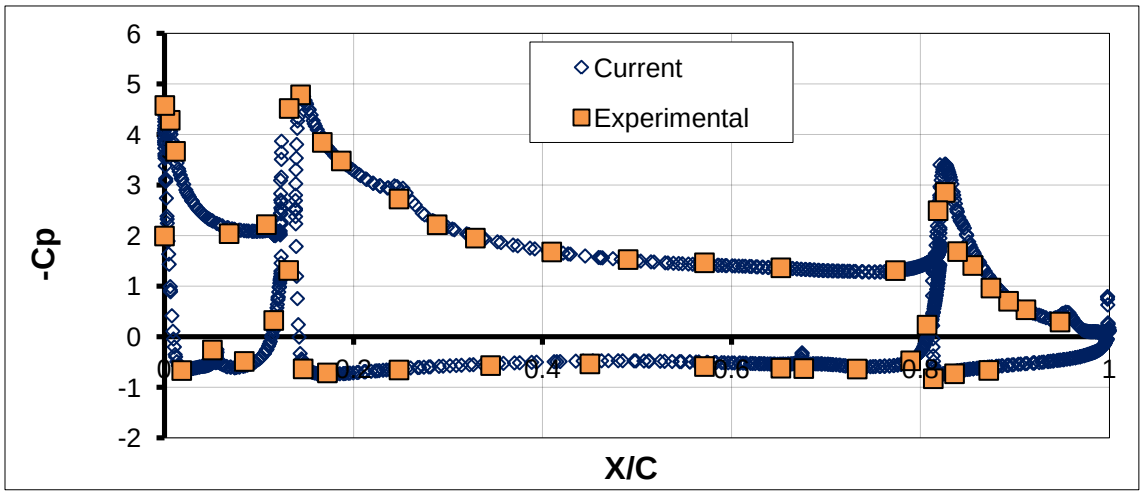


Figure 5.100 – Pressure coefficient distribution around the DLR-F11 wing at 75.1% span (12° attack angle).

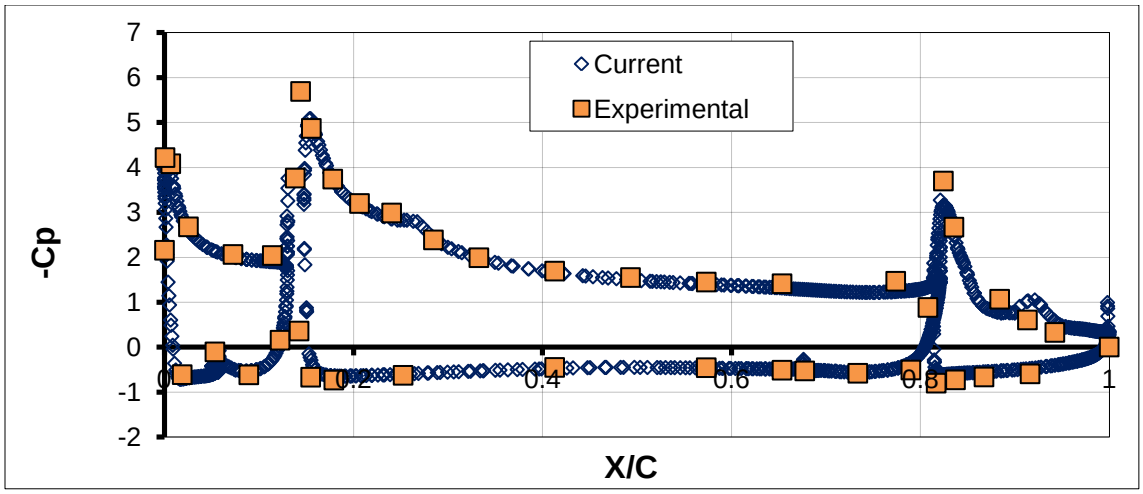


Figure 5.101 – Pressure coefficient distribution around the DLR-F11 wing at 81.8% span (12° attack angle).

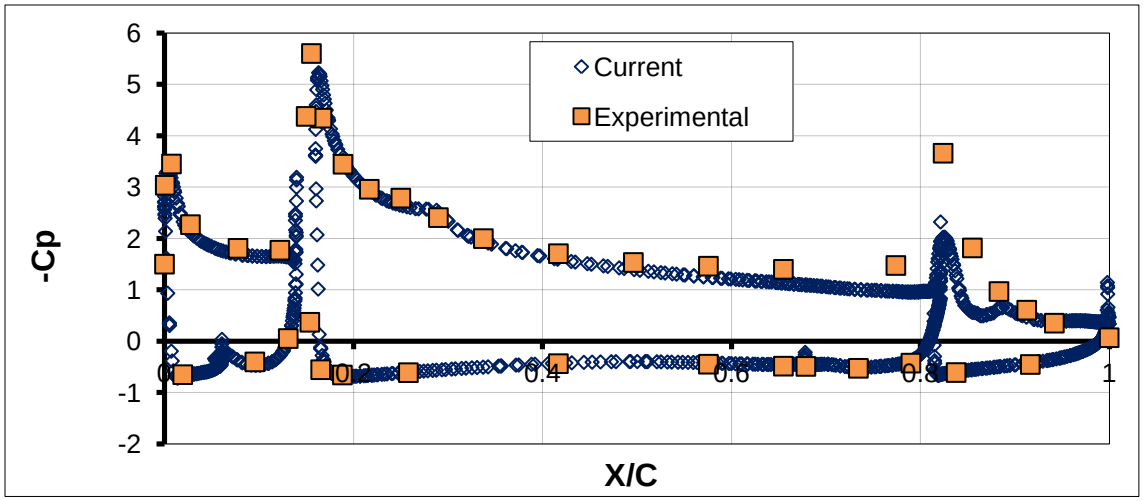


Figure 5.102 – Pressure coefficient distribution around the DLR-F11 wing at 89.1% span (12° attack angle).

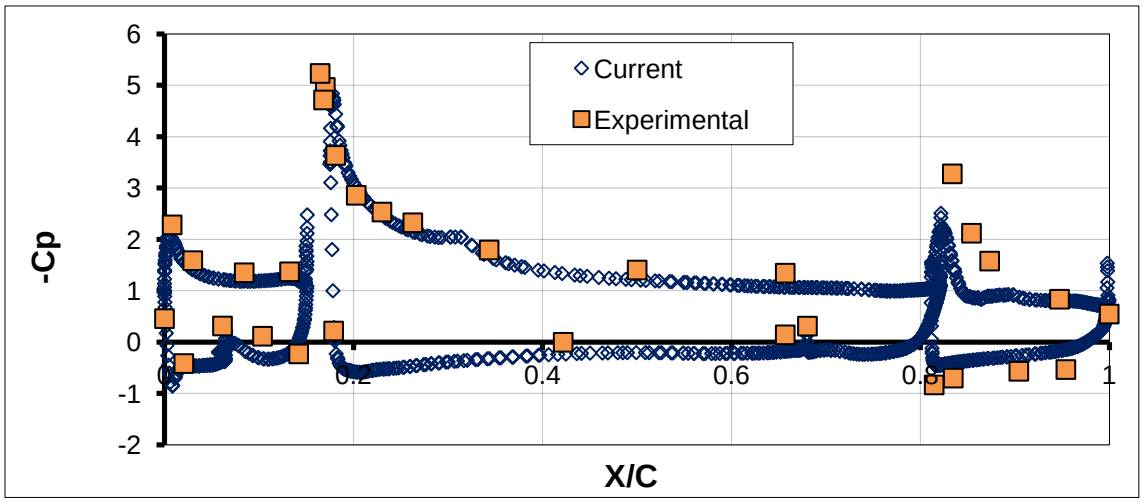


Figure 5.103 – Pressure coefficient distribution around the DLR-F11 wing at 96.4% span (12° attack angle).

In Figures 5.104, 5.105 and 5.106 the pressure contours around the wing at three wing spans are presented, along with the flow streamlines. Several flow phenomena can be observed in those Figures, especially from the behavior of the streamlines. The flow at the rear of the slat area, as well as at the area between the main wing and the flap, presents some recirculation, which seems to be transferred towards the tip of the wing and slowly dissipates. The stagnation point at the three components of the wing is obvious, with high pressure values and the colliding of the streamlines with the wing geometry. Due to the curvature of the wing, set up for landing, the streamlines are curved over the flap area. This phenomenon is carried out at the greater percentage of the wing, while at the tip separation of the flow occurs, as it is obvious in Figure 5.106. This can also justify the poor agreement with the experimental results near the tip of the wing at the flap area. In Figure 5.107 pressure contours on the surface of the DLR-F11 mode are presented.

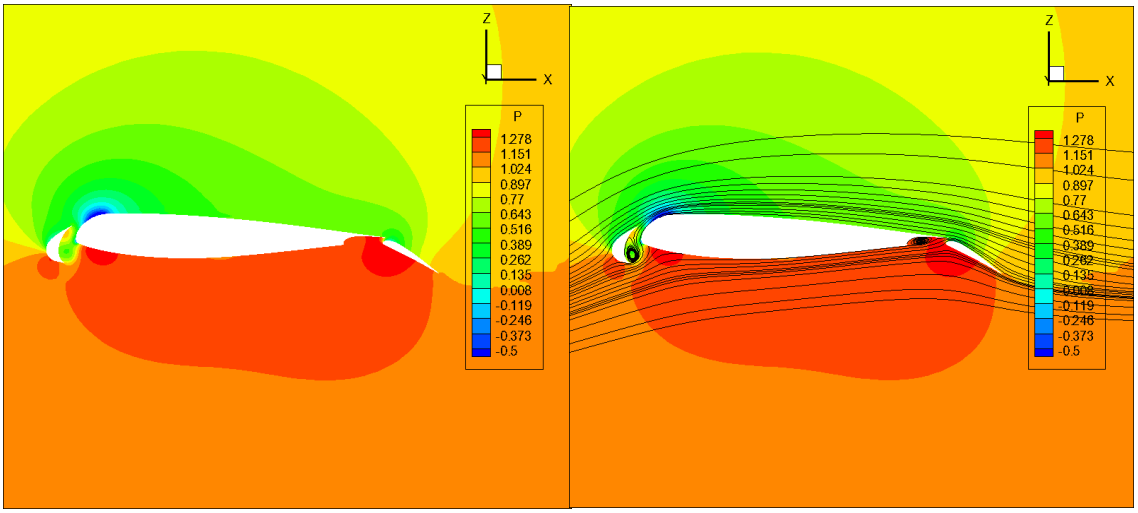


Figure 5.104 –Pressure contours (left) and velocity streamlines (right) around DLR-F11 wing at 15% wing span (12^{o} angle of attack).

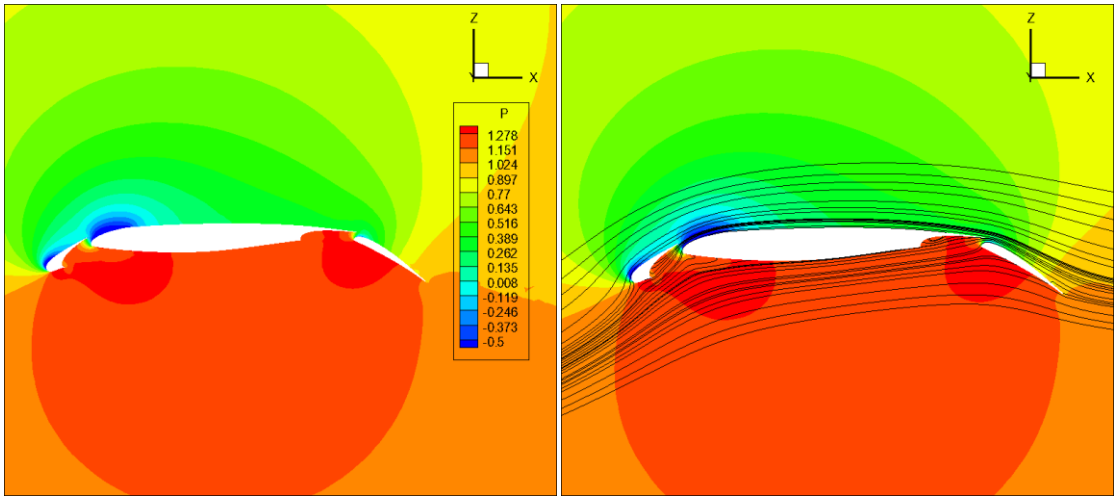


Figure 5.105 –Pressure contours (left) and velocity streamlines (right) around DLR-F11 wing at 54.3% wing span (12^{o} angle of attack).

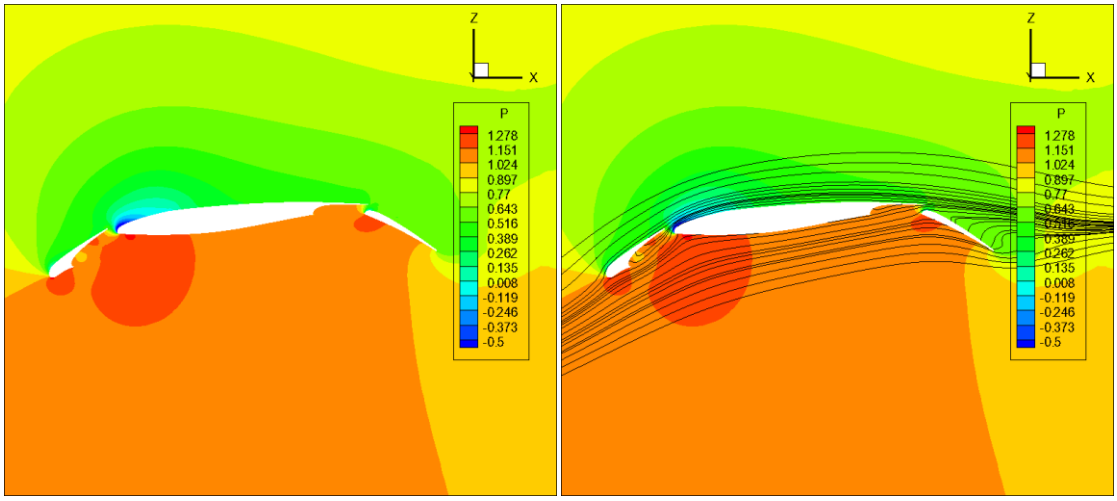


Figure 5.106 –Pressure contours (left) and velocity streamlines (right) around DLR-F11 wing at 96.4% wing span (12^o angle of attack).

In Figure 5.108 a comparison between the surface streamlines at the upper wing area for the cases with 7^o and 12^o angles of attack is illustrated. In both cases the effect of the aircraft body to the flow at the base of the wing is obvious, as there is significant curvature of the flow, especially on the main wing area. Due to the change in the angle of attack there is a bending of the streamlines around the aircraft body, especially in the lower figure, near the trailing edge of the wing. At the flap region there is significant flow separation in both cases, especially towards the wing tip, although the phenomenon is more intense for the case with the higher angle of attack. The flow separation regions can be easily identified as the regions where the streamlines coincide and form a single line. The vorticity contours at the wake of the wing of the DLR-F11 model are illustrated in Figure 5.109. As it was expected, the tip vortex phenomenon occurs in this case as well, with significant intensity, while vortices seem to occur near the lower part of the aircraft body. These vortices represented actually the bending of the streamlines around the aircraft body that was observed in Figure 5.108.

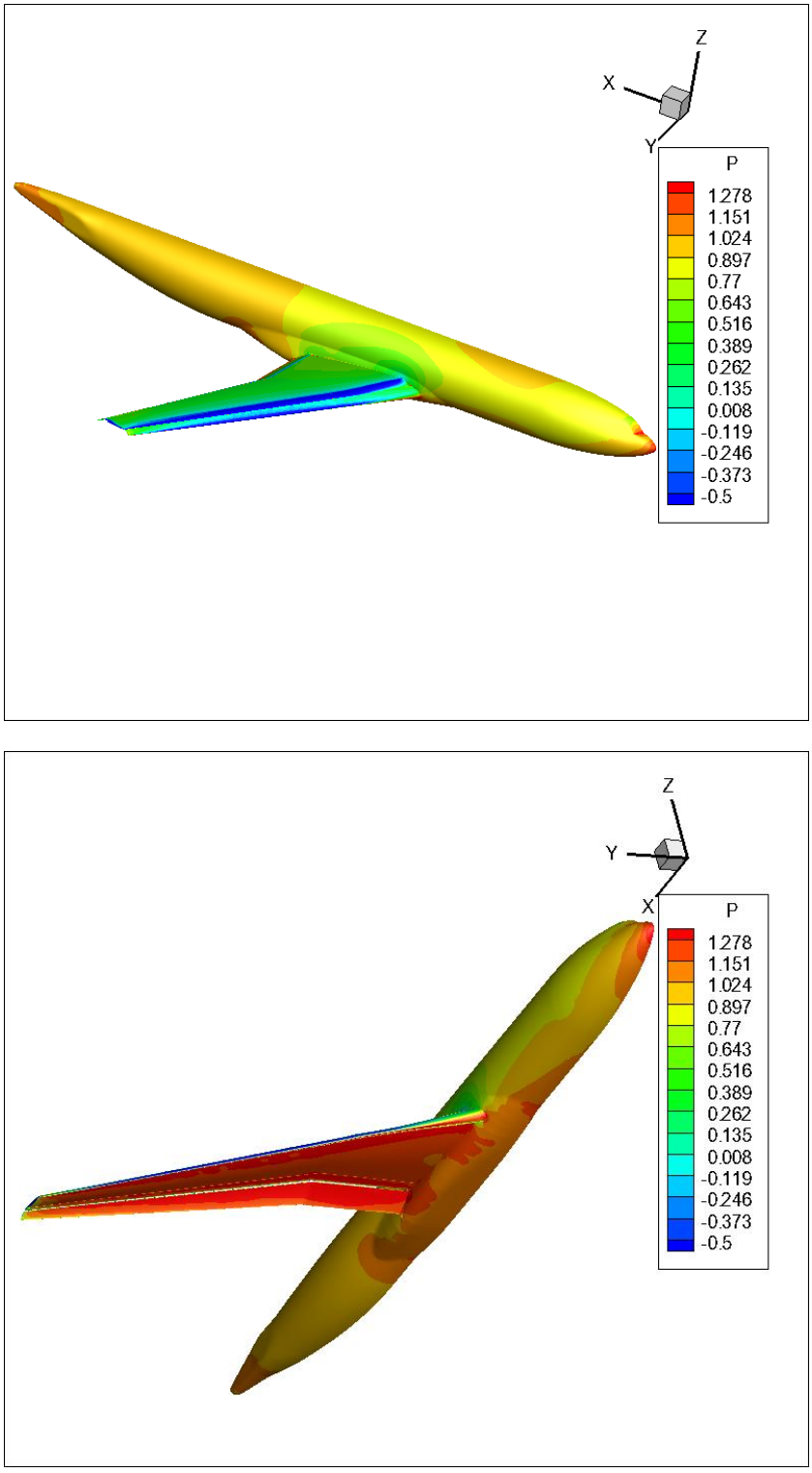


Figure 5.107 – Pressure contours on the DLR-F11 aircraft surfaces (12^{o} angle of attack).

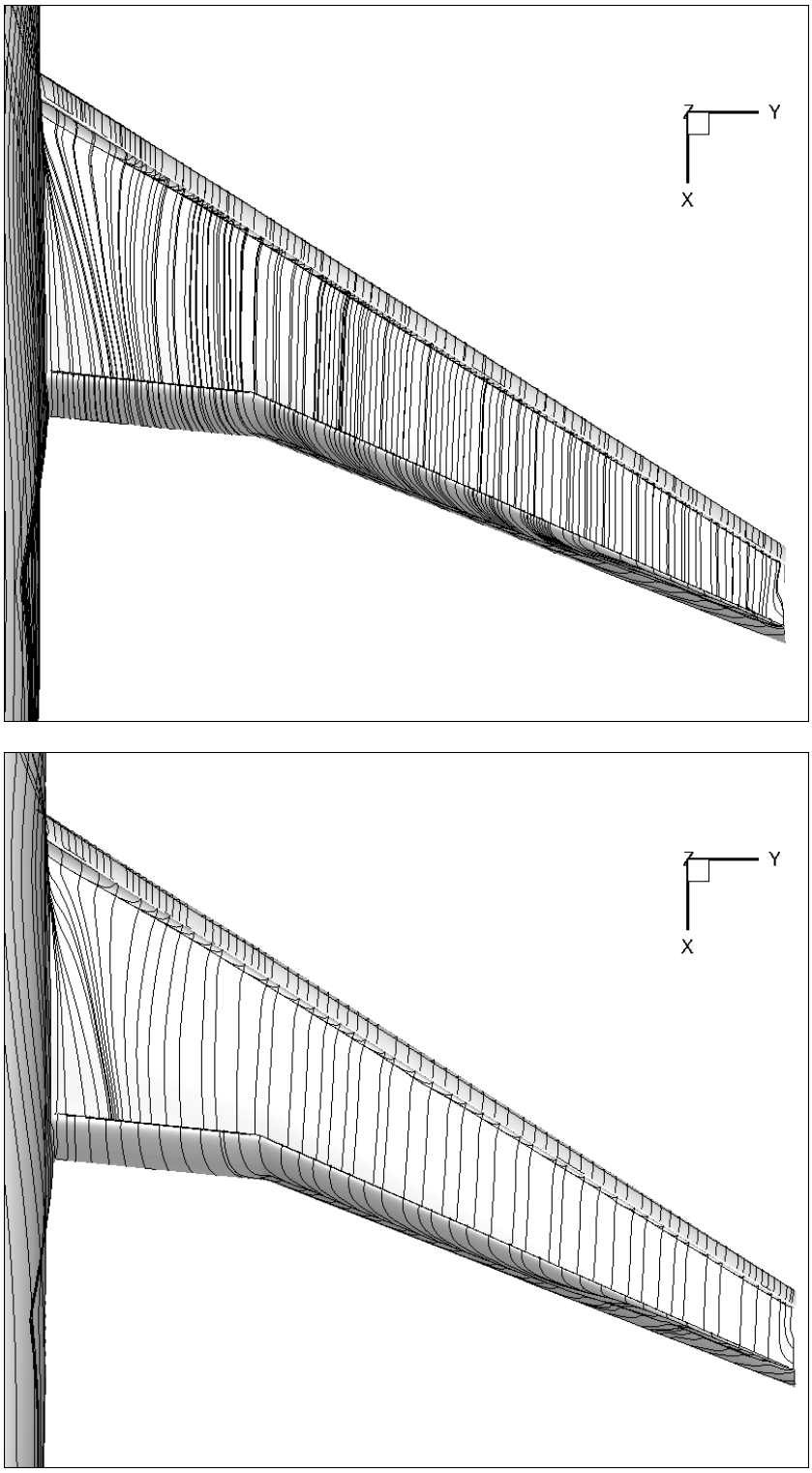


Figure 5.108 – Surface streamlines on the upper wing area with flow at 7^{o} angle of attack (top) and 12^{o} angle of attack (bottom).

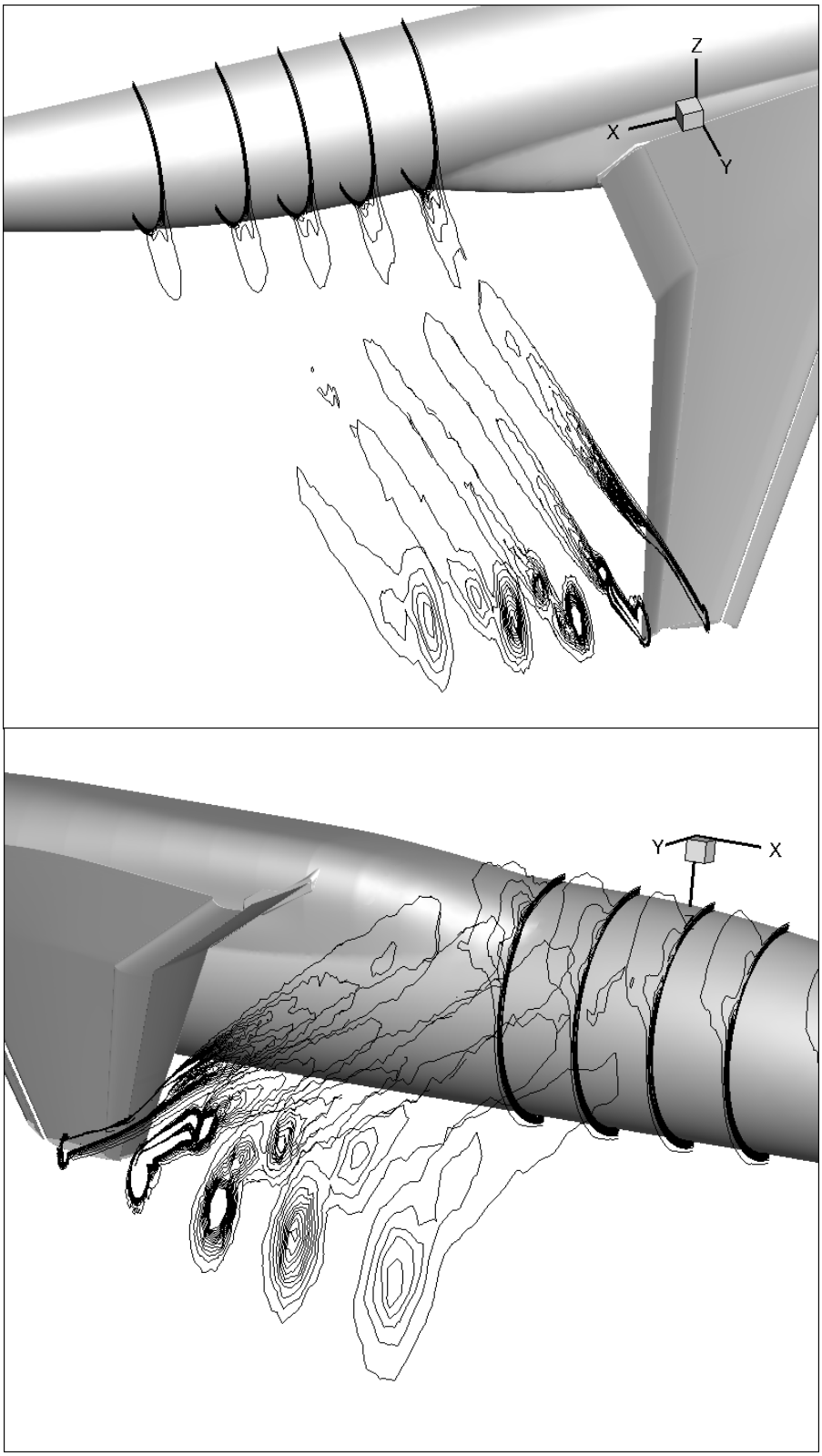


Figure 5.109 – Vorticity contours at the wake of the wing of the DLR-F11 model (12^{o} angle of attack)

5.3 Evaluation of the multigrid scheme

For the assessment of the multigrid scheme, several simulations of some of the aforementioned test cases were performed, using different multigrid techniques and compared to non-multigrid test runs, providing insight on the achieved acceleration of the iterative procedure.

5.3.1 Inviscid flow over a rectangular wing with a NACA0012 airfoil

For this test case three coarser meshes were generated with the isotropic agglomeration method. Figure 5.110 illustrates a far view of the initial fine grid and the three coarser generated ones, while in Figure 5.111 a closer view of the area around the rectangular wing is presented. For the purposes of comparison only FAS was implemented in this case. The derived history of residual convergence per number of iterations and (wall-clock) time are presented in Figure 5.112, respectively. The speed-up of convergence of the multigrid run compared to a non-multigrid one was equal to approximately *18.9* regarding the number of iterations, and equal to approximately *15.6* regarding the elapsed time.

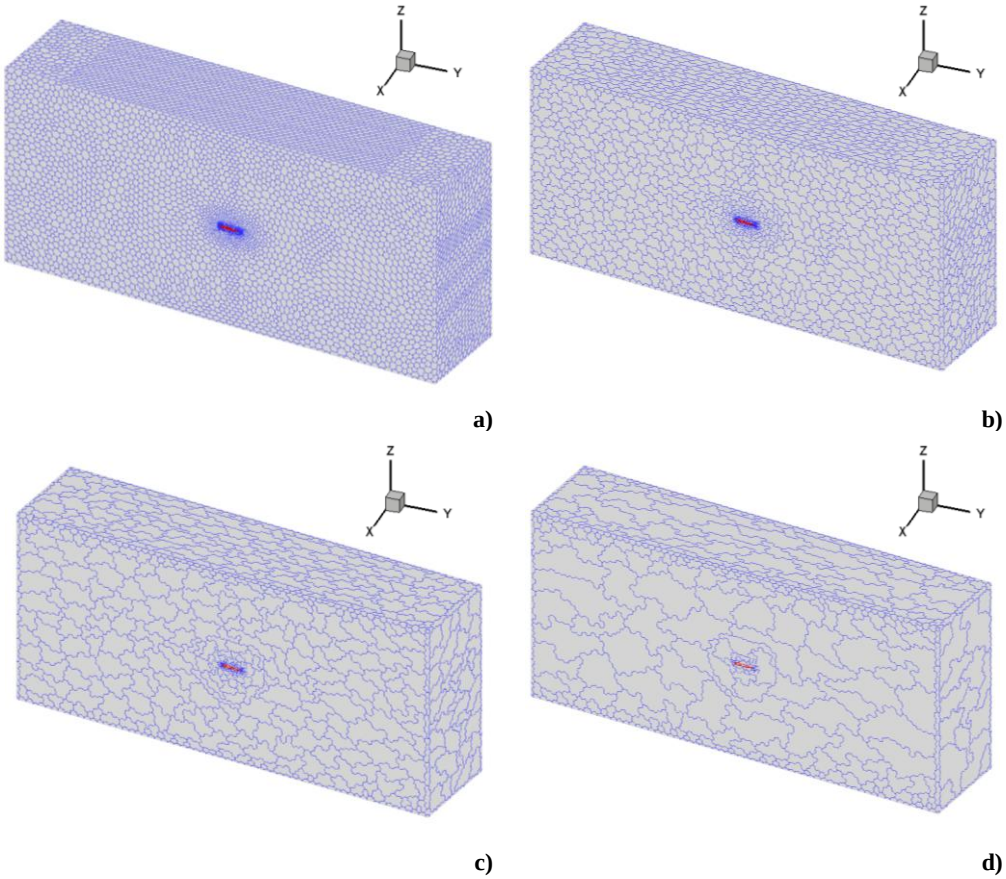


Figure 5.110 – Initial and three coarser grids generated via isotropic agglomeration for use with the multigrid scheme of the complete domain of the rectangular wing with NACA0012 airfoil flow field.

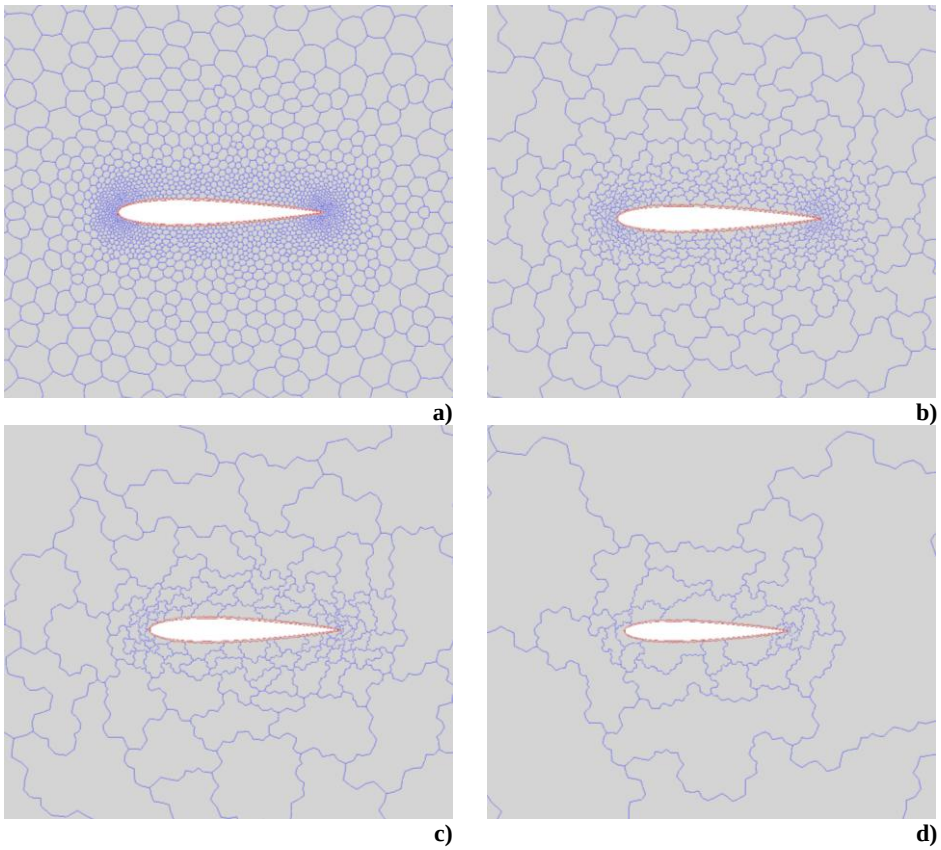


Figure 5.111 – Initial and three coarser grids generated via isotropic agglomeration for use with the multigrid scheme; close view around the NACA0012 airfoil.

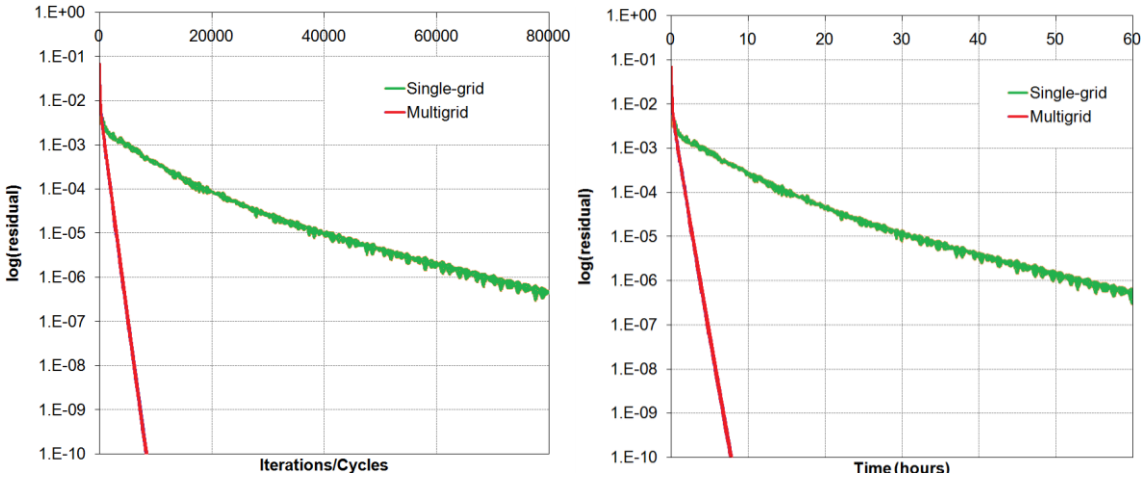


Figure 5.112 – Convergence history for the case of inviscid flow around NACA0012 rectangular wing, per iterations (left) and per elapsed time (right).

5.3.2 Three dimensional cavity flow (*Re=400*)

For this test case two coarser multigrid meshes were generated with two methodologies; one with the isotropic agglomeration method and one with the full-coarsening directional agglomeration method. For the former, the number of control volumes was reduced approximately eight times, while for the latter the generated control volume number was five times smaller than with the initial fine grid. In Figure 5.113 the initial control volume arrangement, along with the two coarser generated ones with the directional agglomeration method are presented.

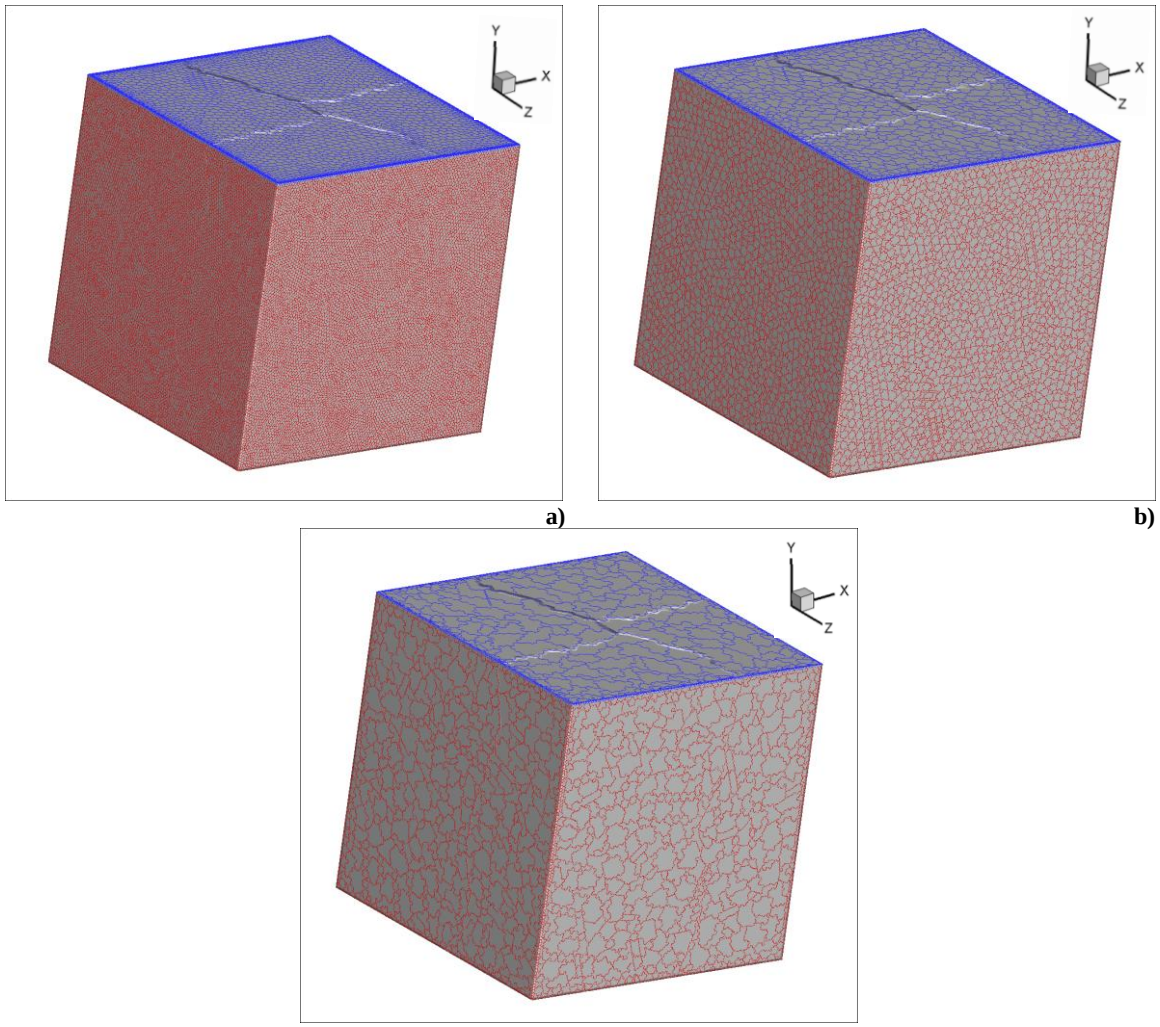


Figure 5.113– Initial control volume mesh and two coarser generated mesh via directional agglomeration scheme for the lid-driven cavity flow test case.

The convergence history per iterations and per elapsed time of both types of agglomeration schemes, along with a run with no multigrid implementation are presented in Figure 5.114. For the multigrid runs the combined FMG-FAS process was implemented. As it was expected, better acceleration was achieved, both in terms of number of iterations and elapsed time, with the full coarsening directional agglomeration method; a speed-up of approximately *6.3* was achieved per iterations and approximately *5.1* per elapsed time. With the isotropic agglomeration the respective speed-up factors were *~5.0* and *~3.6*. All simulations were allowed to converge up to a final

residual of *1.0E-9*. The advantage of the directional agglomeration strategy resides in its capability to preserve the initial topology of the hybrid mesh.

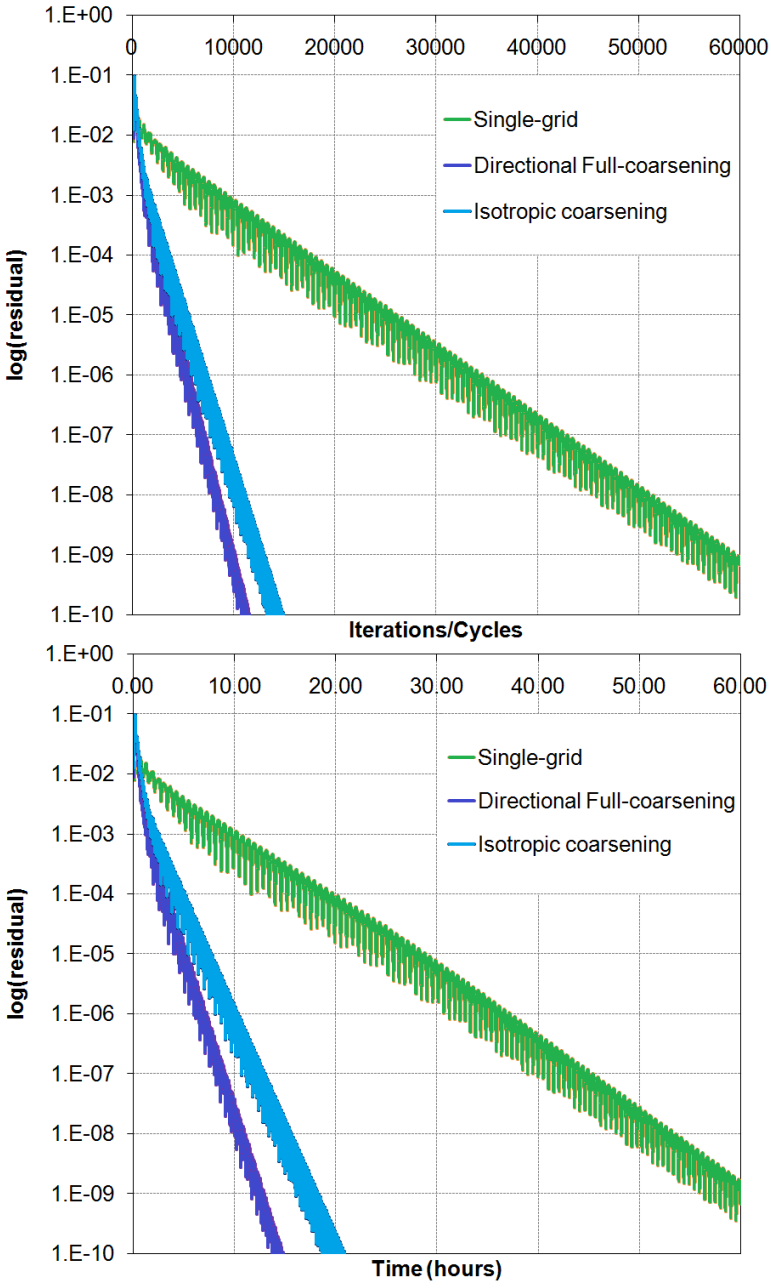


Figure 5.114 – Convergence history of pressure per number of iterations/cycles (top) and time (bottom) for the 3D lid-driven cavity flow case at *Re=400*.

5.3.3 Steady turbulent flow around an axisymmetric submarine hull at 0° angle of attack (*Re=12.0E+6*).

For the case of the turbulent flow around the DARPA SUBOFF hull geometry at 0^o angle of attack three coarser grids were generated with a full-coarsening directional strategy. The initial grid along with the generated coarser ones is presented in Figure 5.115. The number of control volumes was reduced approximately five times on each agglomeration level. In Figures 5.116 and 5.117 detailed views of the fused control volumes that make up the three coarser meshes at the bow and stern area of the submarine, respectively, are presented.

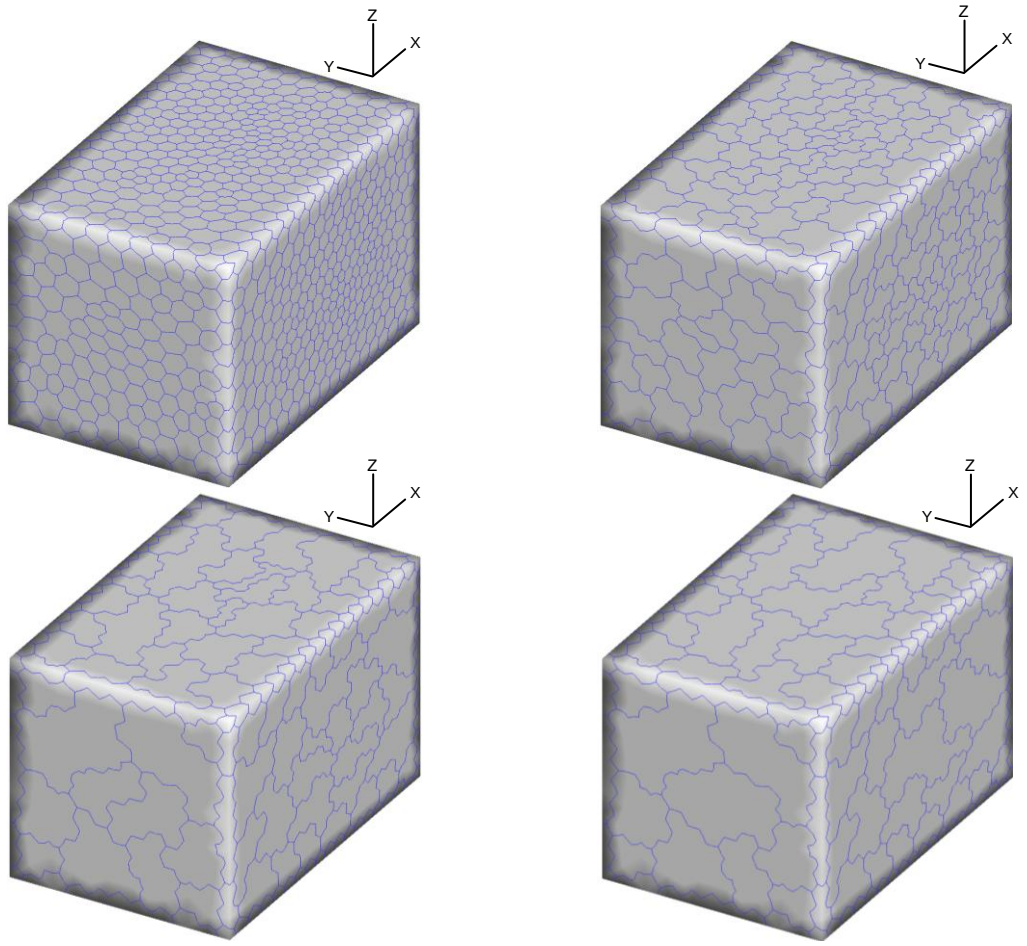


Figure 5.115 – Initial fine grid and three coarser grids generated with the full-coarsening directional agglomeration methodology.

Both FMG-FAS and only-FAS multigrid strategies were implemented for this case in order to examine the improvement to convergence that they can provide, compared to the convergence of a non-multigrid simulation run. In Figures 5.118 and 5.119 the convergence history per iterations and per time of pressure and of the turbulent kinetic energy, respectively, are presented for all multigrid and non-multigrid simulations. The FMG-FAS scheme achieved a speed-up of approximately 5.0 in terms of elapsed time, compared to the single grid (~*3.0E-4* residual of pressure). The corresponding speed-up factor for the FAS scheme was approximately *3.0*. The superiority of the FMG-FAS scheme is obvious in this case, as despite the oscillatory nature of

the convergence it managed to achieve at the same iterations\elapsed time as with simple FAS a residual of more than an order of magnitude lower. The superiority of the multigrid methods against that of the non-multigrid runs is more obvious in Figure 5.119, where the irregular bumps that can be seen at the convergence of turbulent kinetic energy for the non-multigrid run, are not visible at the convergence history of the multigrid ones.

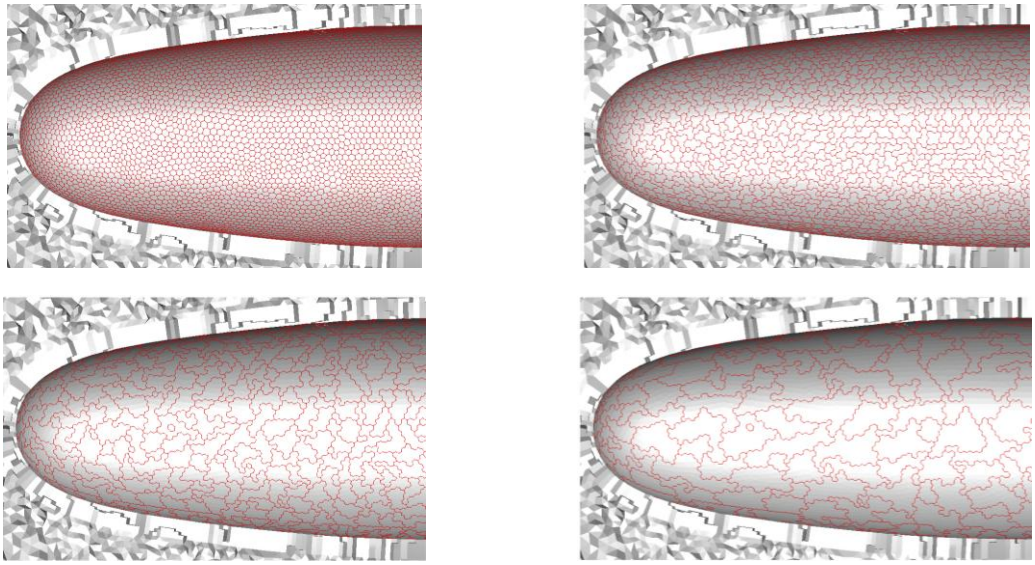


Figure 5.116 – Close-up view of the fused control volumes at the bow area of the submarine hull.

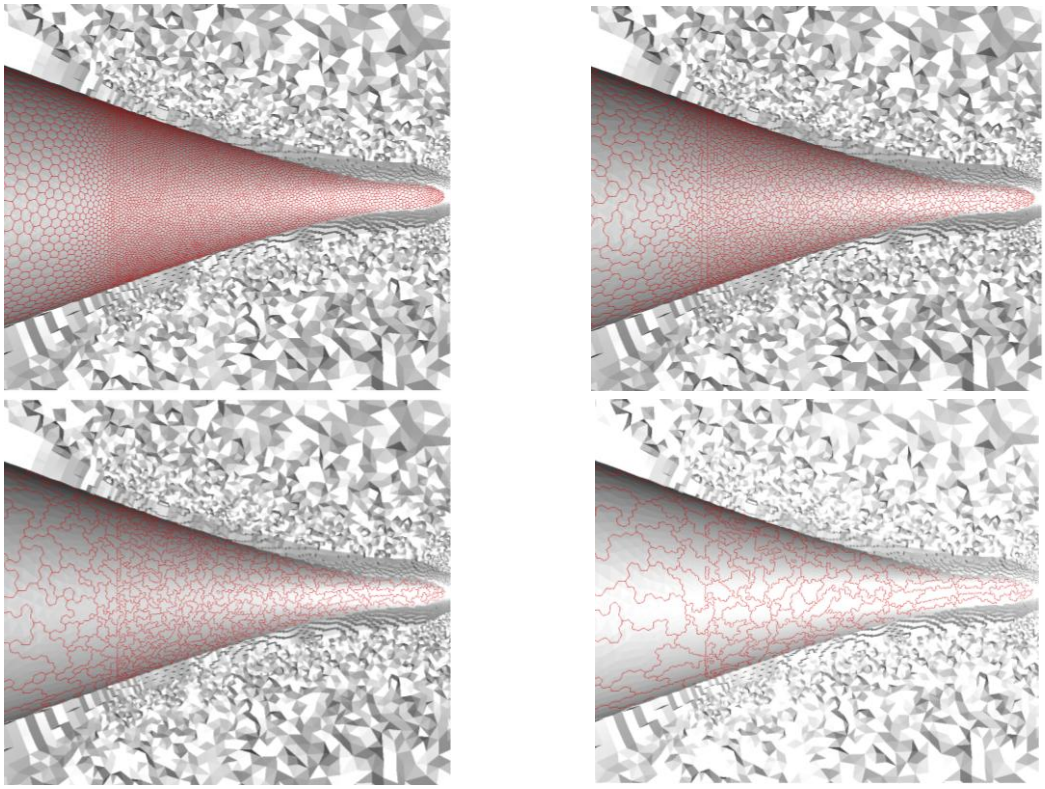


Figure 5.117 – Close-up view of the fused control volumes at the stern area of the submarine hull.

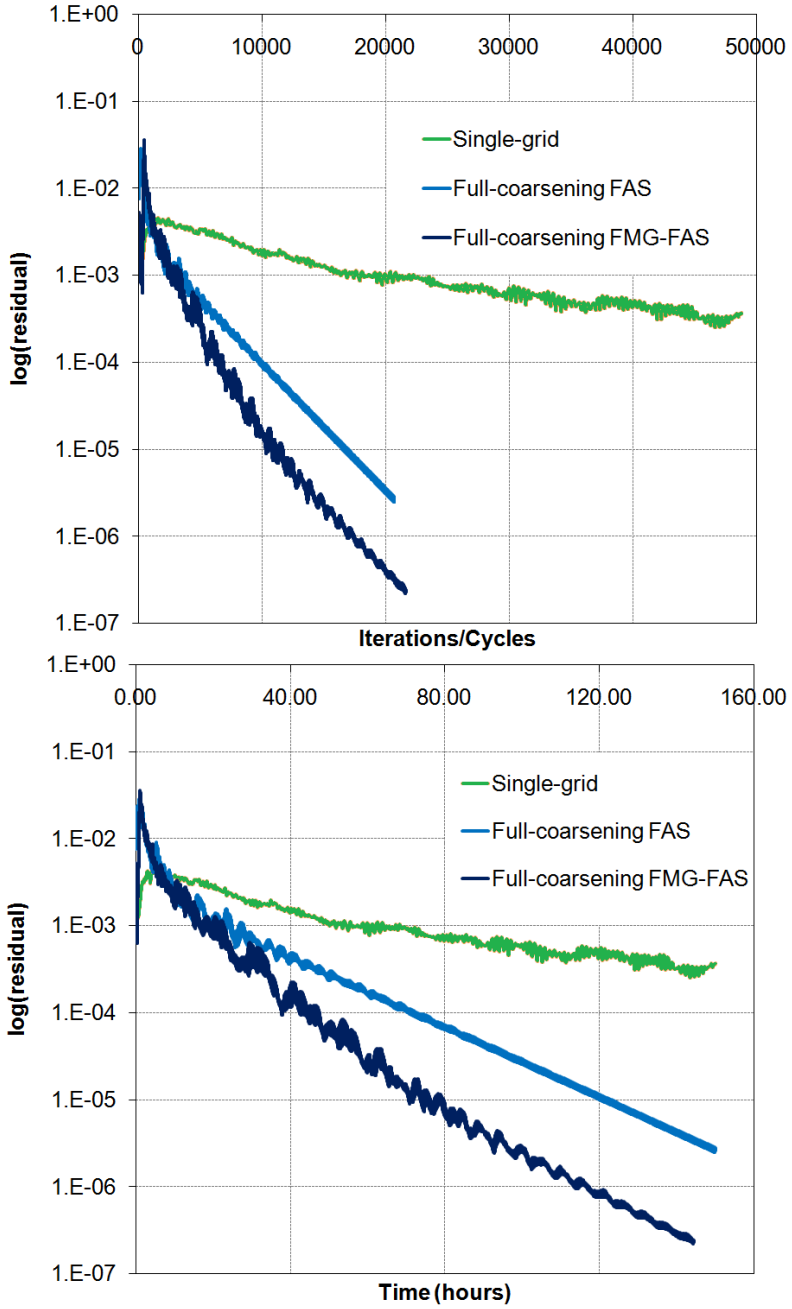


Figure 5.118 – Convergence history of pressure for multigrid and single grid runs of the turbulent flow around the SUBOFF hull geometry.

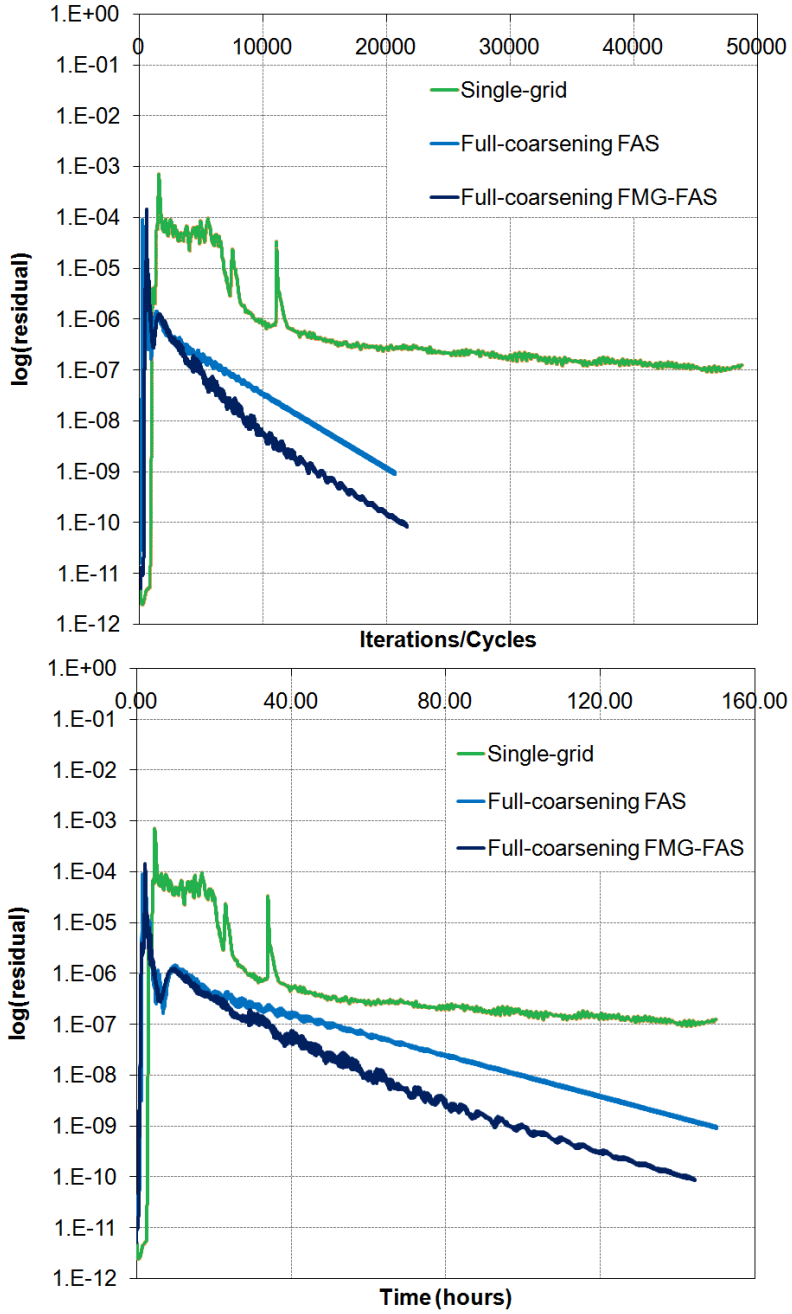


Figure 5.119 – Convergence history of turbulent kinetic energy for multigrid and single grid runs of the turbulent flow around the SUBOFF hull geometry.

CHAPTER 6

CONCLUSIONS

6.1 Summary

In this thesis the development of a code that enables the simulation of incompressible fluid flows, by solving the Reynolds Averaged Navier-Stokes (RANS) equations, enhanced with the artificial compressibility method, was reported. As the Reynolds averaging scheme, along with the Boussinesq assumption [Bla01] was used in this work, evaluation of the kinetic energy was succeeded with the SST model. The governing equations are discretized in space over hybrid unstructured three-dimensional computational grids, composed by tetrahedrons, prisms and pyramids, with a node-centered finite-volume scheme [Lyg14a] [Sar14] [Lyg16]. Temporal discretization was performed with an explicit four-stage Runge-Kutta scheme (RK(4)), while for the simulation of time accurate flows, a dual time stepping approach was adopted. At the boundaries of the computational domain the typical no-slip or free-slip conditions were applied on wall boundaries for viscous or inviscid flows, respectively, while at the inlet and outlet regions of the mesh a locally one-dimensional characteristic type of boundary condition was formulated [Sar14]. Acceleration of the aforementioned techniques was performed in two ways; via an agglomeration multigrid scheme [Mav99] [Car00] [Nish13] [Lyg14a] [Lyg14b] [Lyg14c], and a domain decomposition parallelization approach [Ven95] [Lan96] [Sar15]. With the former approach, several successively coarser grids are generated and relaxation of the PDEs is performed on the coarser ones and prolonged to the finer meshes, thus eliminating low frequency errors. With the latter method, the initial grid is divided into smaller ones, and the governing equations are solved simultaneously on all sub-domains, utilizing the architecture of multi-core processors.

The described methodologies were validated against standard benchmark test cases, as well as more complex ones. The obtained results were in satisfactory agreement with the experimental data and with numerical results from reference solvers, thus demonstrating the performance of the proposed methodology for such simulations. More specifically, in the case of steady-state solutions, the current code's capability to predict laminar viscous flows was assessed with standard test cases, such as the flow inside a closed cavity [Tai05] [Vra12], the flow around a circular cylinder at low Reynolds number [Cou77] [Vra12], or the flow around a sphere [Tan56] [LeC70] [Lee00] [Wan08]. In the first case the velocity profiles at the mid-plane of the cavity was compared to corresponding numerical results found in open literature, and were found in good agreement. In the second and third case the geometric characteristics of the vortices generated at the wake of the simulated geometries agreed well with the corresponding data from reference solvers. In all three cases recirculation phenomena were adequately predicted, thus verifying the proposed methodologies' capabilities to predict such phenomena. Of great interest was the adoption of the DARPA SUBOFF test cases [Gro89] [Hua89] [Gor90] [Liu98] for the evaluation of the current code's capabilities to predict turbulent phenomena in (pseudo-) steady state. The selected test cases of the SUBOFF family concerned the flow around an axisymmetric

submarine hull at three different angles of attack, namely at 0^o, 18^o and 30^o, and different Reynolds numbers, while additionally the flow around the same axisymmetric hull, with a fairwater configuration attached to it was simulated at a zero angle of attack and yaw. In the former set of test cases the results comprised the pressure coefficient distribution over the hull geometry, as well as the velocity profiles near the stern. The simulation results were compared to corresponding numerical results from reference solvers, as well as experimental data [Tox08] [Gro11] and exhibited satisfactory agreement. For the latter case of the SUBOFF group the pressure coefficient distributions at several points of the fairwater surface was computed and compared to reference data [Gor90]. With the DARPA SUBOFF test case, the demonstration of the *Galatea-I* solver's capabilities to predict complex flow phenomena, such as flow separation, recirculation vortices and tip vortex phenomena, was made possible.

Unsteady flows were simulated with the *Galatea-I* solver, with the use of the dual time stepping scheme. For the evaluation of this methodology the laminar and turbulent viscous flow around a circular cylinder was initially simulated. The code was able in both cases to produce the so called Von Karman vortex street. The characteristics of the flow, namely the Strouhal number for the vortex shedding phenomenon and the lift and drag coefficient variation on the cylinder were in good agreement with reference results [Rog90] [Bel95] [LiuC98] [Cha99] [Tai03] [Vra12], especially in the case of laminar flow. Discrepancies in the drag coefficient variation for the turbulent flow case were found; however, this was attributed to the rather coarse mesh that was used in this case. The turbulent flow around the CAARC Standard Tall Building test case [Mel80] [Syk83] [Hua07] [Bra09] [Dag09] [Zha15] was realized with the unsteady flow solver. This case, initially established as a standard for the simulation of natural wind characteristics, was used for the demonstration of the *Galatea-I* capabilities to produce turbulent flow results of unsteady nature. Finally, the flow around the DLR-F11 aircraft model in high lift configuration [Cav14] [Chi14] [Del14] [Eli14] [Hof14] [Han14] [Lee15] [Mav15] [Mur15] [Rud14] [Rum14] was successfully simulated with the use of the proposed methodology. The flow around an aircraft, although compressible in most cases, is of incompressible nature during the phases of take-off, or landing; the Mach in these cases is sufficiently low for the flow to be characterized as incompressible. The configuration was simulated in two angles of attack and the solver was able to successfully produce very satisfactory results in the form of the pressure coefficient distribution around the wing elements at various wing spans; the current numerical results were in very good agreement with experimental ones, although minor discrepancies were found at the tip of the wing due to insufficient density of the computational mesh in this region. This kind of flow is generally of unsteady nature; the simulated tip vortices and the flow separation across the wing flap area attest to this statement. The comparable results were obtained as the mean state of the unsteady flow.

Significant contribution to the efficiency of the described simulations was obtained through the acceleration techniques that were incorporated to the *Galatea-I* solver. In all test cases presented in this study the multi-core capabilities of the available computing systems were exploited via the parallel processing methodology. A comparison between the various agglomeration strategies was implemented on three test cases; the inviscid flow around a NACA0012 rectangular wing, the laminar viscous flow inside a closed cavity and the turbulent viscous flow around the DARPA SUBOFF axisymmetric hull model at zero degrees angle of attack. For the inviscid flow test case

three coarser meshes were generated via the isotropic agglomeration methodology. Compared to the single grid simulation a maximum speed-up coefficient of approximately *15.6* regarding the elapsed time was observed. For the cavity flow test case, besides the comparison between multigrid and non-multigrid simulations, the directional and isotropic methodologies of agglomeration were also tested; two coarser meshes were generated with each agglomeration method. The multigrid simulations provided a speed-up coefficient over the single grid solution of approximately *3.6* and *5.1* for the isotropic and directional agglomeration strategies, respectively, with the latter being clearly superior, as it was expected. For the turbulent flow test case no isotropic agglomeration grids were generated; three coarser grids were generated with the directional agglomeration scheme. In this case comparison between the simple Full Coarsening FAS multigrid scheme and the Full Coarsening FMG-FAS scheme was realized, with the latter proving superior to the former; a speedup factor of approximately *5.0* was observed, compared to the *3.0* speedup coefficient that was achieved with the simple FAS methodology. The superior acceleration results of the FMG-FAS methodology is clearly attributed to its nature; just by solving the governing equations on the coarser grids a crude approximation of the flow field can be achieved, which the finer grids can quickly enhance.

6.2 Contributions

The main contributions of this work are listed below:

1. A computational methodology and the corresponding software tool was developed (*Galatea-I*) for the simulation of the 3D, steady and unsteady, inviscid, laminar, and turbulent flows, based on the artificial compressibility concept. The methodology uses unstructured hybrid grids, is parallelized using the domain decomposition concept, while a considerable acceleration is achieved through the use of the multi-grid technique. This development is not a trivial task, as the materialization and the successful combination of such complex computational procedures is a challenge.
2. The adoption of the artificial compressibility concept for the solution of the incompressible flow equations, and the combination of this concept with the multigrid technique and with parallel processing, which render the artificial compressibility concept very attractive and competitive to pressure-correction techniques.
3. The development of boundary conditions for 3D flow fields, based on characteristics, which proved to be effective and accurate.
4. The development of the parallelization strategy, based on the domain decomposition concept and MPI protocol. Moreover, for the implementation of this strategy, several supporting methodologies (and the corresponding software) have been developed as well, such as: a) the algorithm for the correct renumbering of the sub-domains of a partitioned initial grid, b) the algorithm for the generation of the overlapping region between adjacent partitions, c) the algorithm for the correct transfer of data between domains, incorporating collective communication MPI functions. This last algorithm ensured the correct communication of adjacent partitions and the balanced synchronous execution of all parallel instances of the same code.
5. The incorporation and adaption of a (previously developed) multigrid methodology in the *Galatea-I* code. The extensive evaluation of this methodology to various test cases demonstrated its ability to considerably accelerate the convergence rate.

6. The use of a very wide suite of different test cases (steady and unsteady) for the extensive validation of the proposed methodology and the developed software. This extensive evaluation demonstrated the potential of the proposed methodologies in simulating complicated flow fields with very satisfactory accuracy and with comparable or better results from those of alternative methodologies. Moreover, proved the equipotential of the artificial compressibility concept for accurately simulating such complicated flow phenomena.

6.3 Ongoing – future work

The *Galatea-I* code has been already incorporated in a computational methodology for Fluid-Structure Interaction (FSI), serving as the fluid flow simulator. With this methodology the computed forces on solid structures that interact with the fluid flow are transferred via a partitioned FSI coupling procedure to an open-source Computational Structural Mechanics (CSM) code [Calculix] that computes the resulting deformations, and rates of deformation. The spatial coupling of the structural and flow meshes, is realized with an interpolation scheme, based on Radial Basis Functions (RBFs), which employs the Partition of Unity approach [Ren09]. Mesh deformation is applied with an RBF interpolation methodology that is accelerated with a surface point reduction technique, based on the agglomeration of adjacent boundary nodes [Str15]. The static form of the methodology has been already tested with the CAARC Standard Tall Building model test case and has provided very satisfactory results. For the realization of a dynamic FSI methodology an Arbitrary Lagrangian-Eulerian (ALE) algorithm is under development, to allow for the simulation of flows with moving meshes and boundaries.

Additionally to the RANS based SST turbulence model, several models that incorporate the Large Eddy Simulation approach (LES) have been already applied to the *Galatea-I* solver; the Smagorinsky model [Sma63], the WALE (Wall Adapted Local Eddy-viscosity) model [Nic99], the dynamic Germano-Lilly model [Ger91] [Lil92] and the dynamic Kinetic Energy model [Dav97]. All four methodologies provide the required by the Boussinesq assumption turbulent kinematic viscosity and thus can be used by the same flow equations as the ones used in the *Galatea-I* code.

Future extensions to the current presented work include the following:

- Incorporation of the energy equation in order to simulate heat transfer problems. The incompressible solver could then be coupled with the already developed radiative heat transfer algorithm developed by Lygidakis et al. [Lyg12].
- Development of a procedure for the simulation of free-surface boundaries, such as the Volume of Fluid (VOF) or Moment of Fluid (MOF) methods. With this addition the incompressible flow solver could be used for the simulation of multi-fluid flows with a free surface.
- Further examination of the MPI libraries. A non-geometric grouping of the adjacent partitions could provide further acceleration, as data transfer would only occur inside carefully designed groups.
- Modularization of the *Galatea-I* solver to include other numerical methods. A modular program, developed with Object Oriented programming, could perform geometric based

schemes, such as the domain decomposition approach or the agglomeration method separately from any numerical method. Subsequently, any numerical method would be applied to the core code as a module and the coupling of different methods would be more reasonable.

6.4 Publications

1. G. N. Lygidakis, S. S. Sarakinos, I. K. Nikolos, “Comparison of different agglomeration multigrid schemes for compressible and incompressible flow simulations”, Advances in Engineering Software, Available online 13 January 2016, ISSN 0965-9978, http://dx.doi.org/10.1016/j.advengsoft.2015.12.004
2. S. S. Sarakinos, G. N. Lygidakis, I. K. Nikolos, “Flow Analysis of the DLR-F11 High Lift Model Using a Time-Accurate Unsteady Incompressible Solver Based on the Artificial Compressibility Method”, Journal of Aircraft, 2016 (Submitted).
3. S. S. Sarakinos, G. N. Lygidakis, “Evaluation of a Parallel Agglomeration Multigrid Finite-Volume Algorithm, Named Galatea-I, for the Simulation of Incompressible Flows on 3D Hybrid Unstructured Grids”, ASME 2014 International Mechanical Engineering Congress and Exposition, Montreal, Quebec, Canada, November 14-20, 2014.
4. G.N. Lygidakis, S.S. Sarakinos, I.K. Nikolos, "A Parallel Agglomeration Multigrid Method for Incompressible Flow Simulations", in P. Iványi, B.H.V. Topping, (Editors), Proceedings of the Ninth International Conference on Engineering Computational Technology, Civil-Comp Press, Stirlingshire, UK, Paper 27, 2014.
5. S. S. Sarakinos, G. N. Lygidakis, I. K. Nikolos, “Assessment of the Academic CFD code GALATEA – I with the DARPA SUBOFF Test Case”, ASME 2015 International Mechanical Engineering Congress and Exposition, Houston, Texas, November 13 – 19, 2015.
6. S.S. Sarakinos, G.N. Lygidakis, I.K. Nikolos, “Simulating Unsteady Incompressible Flows using Galatea-I, a Parallel Multigrid Finite-Volume Solver”, Proceedings of the 8th GRACM International Congress on Computational Mechanics, Volos, Greece, 12 – 15 July 2015.
7. S.S. Sarakinos, G.N. Lygidakis, I.K. Nikolos, "Acceleration Strategies for Simulating Compressible and Incompressible Flows", in P. Iványi, B.H.V. Topping, (Editors), Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, Stirlingshire, UK, Paper 47, 2015.
8. G. A. Strofylas, G. I. Mazanakis, S. S. Sarakinos, G. N. Lygidakis, I. K. Nikolos, “Using Improved Radial Basis Functions Methods for Fluid-Structure Coupling and Mesh Deformation”, ECCOMMAS Congress 2016, Crete Island, Greece, 5–10 June 2016 (Accepted).
9. S. S. Sarakinos, G. N. Lygidakis, I. K. Nikolos, “Assessment of the academic CFD code Galatea-I with the DLR-F11 model in High Lift configuration”, ASME 2016 International Mechanical Engineering Congress and Exposition, Phoenix Convention Center, 11-17 November, 2016 (Submitted).

10. G. N. Lygidakis, S. S. Sarakinos, I. K. Nikolos, “Comparison of different LES turbulence models for the simulation of flow over the CAARC standard tall building”, ASME 2016 International Mechanical Engineering Congress and Exposition, Phoenix Convention Center, 11-17 November, 2016 (Submitted).
11. G. A. Strofylas, G. I. Mazanakis, S. S. Sarakinos, G. N. Lygidakis, I. K. Nikolos, “On the use of Improved Radial Basis Functions methods in Fluid Structure Interaction Simulations”, ASME 2016 International Mechanical Engineering Congress and Exposition, Phoenix Convention Center, 11-17 November, 2016 (Submitted).

REFERENCES

[Ach89] S. Acharya, and F. H. Moukalled, "Improvements to incompressible flow calculation on a nonstaggered curvilinear grid", *Numerical Heat Transfer, Part B Fundamentals*, 15(2), pp. 131-152, 1989.

[Ahn06] H.T. Ahn, and Y. Kallinderis, "Strongly Coupled Flow/Structure Interactions with a Geometrically Conservative ALE Scheme on General Hybrid Meshes", *Journal of Computational Physics*, 219, pp. 671-696, 2006.

[Amd67] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the Spring Joint Computer Conference*, pp. 483–485, April 18-20, 1967.

[And94] W.K. Anderson, and D.L. Bonhaus, "An Implicit Algorithm for Computing Turbulent Flows on Unstructured Grids", *Computers & Fluids*, 23, pp. 1-21, 1994.

[And96] W. K. Anderson, R. D. Rausch, and D. L. Bonhaus, "Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids", *Journal of Computational Physics*, 128(2), pp. 391–408, 1996.

[ANSYS06] ANSYS CFX-Solver Theory Guide, ANSYS CFX Release 11.0, December 2006.

[Bar92] T.J. Barth, "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations", in *Proceedings of the AGARD-FDP-VKI special course at VKI*, AGARD-R-787, pp. 6.1-6.61, Rhode-Saint-Genese, 2-6 March, 1992.

[Bea76] R. M. Beam and R. F. Warming, "An implicit finite-difference algorithm for hyperbolic systems in conservation-law form," *Journal of Computational Physics*, 22(1), pp. 87–110, 1976.

[Bel95] A. Belov, L. Martinelli, and A. Jameson, "A new implicit algorithm with multigrid for unsteady incompressible flow calculations", in *Proceedings of the 33rd Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, 1995. AIAA 95-0049.

[Bla01] J. Blazek, *Computational Fluid Dynamics: Principles and Applications*, Elsevier, 2001.

[Bla85] P. A. Blackmore, "A comparison of experimental methods for estimating dynamic response of buildings", *Journal of Wind Engineering and Industrial Aerodynamics*, 18(2), pp. 197-212, 1985.

[Boo54] G. Boole, *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*, Macmillan. Reprinted with corrections, Dover Publications, New York, NY, 1958. (Reissued by Cambridge University Press, 2009; ISBN 978-1-108-00153-3), 1854.

[Bra09] A. L. Braun, and A. M. Awruch, “Aerodynamic and aeroelastic analyses on the CAARC standard tall building model using numerical simulation”, *Computers & Structures*, 87(9), pp. 564-581, 2009.

[Bra96] M.E. Braaten, and S.D. Connell, “Three-Dimensional Unstructured Adaptive Multigrid Scheme for the Navier-Stokes Equations”, *AIAA*, (34), pp. 281-290, 1996.

[Bui93] T. N. Bui, and C. Jones, “A heuristic for reducing fill-in in sparse matrix factorization”, in *Proceedings of Society for Industrial and Applied Mathematics (SIAM) Conference on Parallel Processing for Scientific Computing*, Norfolk, VA (United States), 21-24 Mar 1993.

[Bun85] P. G. Buningt, “A 3-D chimera grid embedding technique”, in *Proceedings of the 7th Computational Physics Conference*, Cincinnati, OH, USA, 1985.

[Calculix] G. Dhondt, CalculiX CrunchiX User’s Manual, version 2.7, 2014.

[Car00] G. Carre, L. Fournier, and S. Lanteri, “Parallel linear multigrid algorithms for the acceleration of compressible flow calculations”, *Computional Methods in Applied Mechanics and Engineering*, 184, pp. 427-448, 2000.

[Car73] L. S. Caretto, A. D. Gosman, S. V. Patankar, and D. B. Spalding, “Two calculation procedures for steady, three-dimensional flows with recirculation,” in *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, pp. 60–68, 1973.

[Cat03] P. Catalano, M. Wang, G. Iaccarino, and P. Moin, “Numerical simulation of the flow around a circular cylinder at high Reynolds numbers”, *International Journal of Heat and Fluid Flow*, 24, pp. 463–469, 2003.

[Cav14] Cavallo, Peter A. “CRUNCH CFD Calculations for HiLiftPW-2 with Discretization Error Predictions”, *52nd Aerospace Sciences Meeting, AIAA SciTech*, (AIAA paper: 2014-0916). 2014.

[Cha99] C.T. Chan, and K. Anastasiou, “Solution of incompressible flows with or without a free surface using the finite volume method on unstructured triangular meshes”, *International Journal for Numerical Methods in Fluids*, 29, pp. 35–57, 1999.

[Che04] Chen, L., and C. W. Letchford, “Parametric study on the along-wind response of the CAARC building to downbursts in the time domain", *Journal of Wind Engineering and Industrial Aerodynamics*, 92(9), pp. 703-724, 2004.

[Che90] G. Chesshire, and W. D. Henshaw, “Composite overlapping meshes for the solution of partial differential equations,” *Journal of Computational Physics*, 90(1), pp. 1–64, 1990.

[Chi14] K. C. Chitale, M. Rasquin, J. Martin, and K. E. Jansen, "Finite Element Flow Simulations of the EUROLIFT DLR-F11 High Lift Configuration", *arXiv preprint arXiv:1402.6759*, 2014.

[Cho67a] A. J. Chorin, “A numerical method for solving incompressible viscous flow problems,” *Journal of Computational Physics*, 2(1), pp. 12–26, 1967.

[Cho67b] A. J. Chorin, “The numerical solution of the Navier-Stokes equations for an incompressible fluid,” *Bulletin of the American Mathematical Society,* 73(6), pp. 928–931, 1967.

[Cou77] M. Coutanceau, R. Bouard, “Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady Flow”, *Journal of Fluid Mechanics,* 79(2), pp. 231-256, 1977.

[Dag09] A. K. Dagnew, G. T. Bitsuamalk, and R. Merrick, “Computational evaluation of wind pressures on tall buildings”, in *Proceedings of the 11th American Conference on Wind Engineering*, San Juan, Puerto Rico, 2009.

[Dag10] A. K. Dagnew, and G. T. Bitsuamlak, “LES evaluation of wind pressures on a standard tall building with and without a neighboring building", in *Proceedings of the 5th International Symposium on Computational Wind Engineering*, Chapel Hill, North Carolina, USA, 2010.

[Dan13] S. J. Daniels, I. P. Castro, and Z-T. Xie, “Peak loading and surface pressure fluctuations of a tall model building”, *Journal of Wind Engineering and Industrial Aerodynamics,* 120, pp. 19-28, 2013,

[Dar88] F. Darema, D. A. George, V. A. Norton, and G. F. Pfister, “A single-program-multiple-data computational model for EPEX/FORTRAN,” *Parallel Computing*, 7(1), pp. 11–24, 1988.

[Dar06] M.S. Darwish, T. Saad and Z. Hamdan, “A High Scalability Parallel Algebraic Multigrid Solver”, in *Proceedings of the European Conference on Computational Fluid Dynamics (ECCOMAS CFD 2006)*, Egmond aan Zee, The Netherlands, 5-8 September, 2006.

[Dav96] L. Davidson, “A pressure correction method for unstructured meshes with arbitrary control volumes”, *International Journal for Numerical Methods in Fluids*, 22(4), pp. 265–281, 1996.

[Dav97] L. Davidson, “LES of recirculating flow without any homogenous direction: A dynamic one-equation subgrid model,” in *Proceedings of the 2nd International Symposium on Turbulence Heat and Mass Transfer*, Delft, Netherlands, pp. 481-490, 1997.

[Del14] T. Deloze, and E. Laurendeau. “NSMB contribution to the 2nd High Lift Prediction Workshop”, *52nd Aerospace Sciences Meeting, AIAA SciTech,* (AIAA 2014-0913), 2014.

[Den79] S. C. R. Dennis, D. B. Ingham, and R. N. Cook, “Finite-difference methods for calculating steady incompressible flows in three dimensions,” *Journal of Computational Physics*, 33(3), pp. 325–339, 1979.

[Din95] P. C. Diniz, S. Plimpton, B. Hendrickson, and R. W. Leland, “Parallel Algorithms for Dynamically Partitioning Unstructured Grids”, in *Proceedings of Society for Industrial and Applied Mathematics (SIAM) Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, USA,15-17 Feb 1995.

[Doo84] J. P. Van Doormaal and G. D. Raithby, “Enhancements of the SIMPLE method for predicting incompressible fluid flows,” *Numerical Heat Transfer,* 7(2), pp. 147–163, 1984.

[Dwy86] H. S. Dwyer, M. Soliman, and M. Hafez, “Time accurate solutions of the Navier-Stokes equations for reacting flows”, in *Proceedings of the Tenth International Conference on Numerical Methods in Fluid Dynamics*, Beijing, China, June 23–27, 1986.

[Dwy89] H. A. Dwyer, and S. Ibrani, “Time accurate solutions of the incompressible and three-dimensional Navier-Stokes equations”, *Computer Methods in Applied Mechanics and Engineering*, 75(1–3), pp. 333–341, 1989.

[Eli14] P. Eliasson, and S-H. Peng, "Results from the Second AIAA CFD High-Lift Prediction Workshop Using Edge", *Journal of Aircraft*, 52(4), pp. 1042-1050, 2014.

[Far93] C. Farhat, and M. Lesoinne, “Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics,” *International Journal for Numerical Methods in Engineering*, 36(5), pp. 745–764, 1993.

[Fer02] J.H. Ferziger, and M. Peric, *Computational Methods for Fluid Dynamics - 3rd Edition*, Springer, 2002.

[Fey65] R. P. Feynman, R. B. Leighton, and M. Sands, *Feyman Lectures on Physics: Quantum Mechanics*, Narosa Publishing House, 1965.

[Fid82] C. M. Fiduccia and R. M. Mattheyses, “A linear-time heuristic for improving network partitions”, in *Proceedings of the 19th Conference on Design Automation*, pp. 175–181, 1982.

[FLU6.3] FLUENT, ANSYS. 6.3. Theory Manual. 2005. Fluent Inc. Central Source Park, 10 Cavendish Court, Lebanon, NH 03766, USA.

[Fly66] M. J. Flynn, “Very high-speed computing systems”, *Proceedings of the IEEE*, 54(12), pp. 1901-1909, 1966.

[Fly72] M. J. Flynn, “Some computer organizations and their effectiveness,” *Transactions on Computers, IEEE*, 100(9), pp. 948–960, 1972.

[Fra90] R. Franke, W. Rodi, B. Schonung, “Numerical calculation of laminar vortex shedding flow past cylinders”, *Journal of Wind Engineering and Industrial Aerodynamics*, 35, pp. 237-257, 1990.

[Ger91] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, “A dynamic subgrid-scale eddy viscosity model”, *Physics of Fluids*, 3(7), pp. 1760–1765, 1991.

[God09] D. Goddeke, S. H. Buijssen, H. Wobker, and S. Turek, “GPU acceleration of an unmodified parallel finite element Navier-Stokes solver,” in *Proceedings of the HPCS’09 International Conference on High Performance Computing & Simulation*, pp. 12–21, 2009.

[Gor90] J.J. Gorski, R.M. Coleman, H.J. Haussling, “Computation of incompressible flow around the DARPA SUBOFF bodies”, *Ft. Belvoir: Defence Technical Information Center*, July 1990. http://www.dtic.mil/get-tr-doc.

[Gre98] P. M. Gresho, and R. L. Sani, *Incompressible flow and the finite element method. Volume 1: Advection-diffusion and isothermal laminar flow*, John Wiley and Sons, Inc., New York, 1998.

[Greg70] Gregory, N., and C. L. OReilly, *Low-speed aerodynamic characteristics of NACA 0012 aerofoil section, including the effects of upper-surface roughness simulating hoar frost*, Technical Report, Her Majesty's Stationery Office, London, 1970.

[Gro89] N.C. Groves, T.T. Huang and M.S. Chang, "Geometric characteristics of DARPA (Defense Advanced Research Projects Agency) SUBOFF models (DTRC model numbers 5470 and 5471)", *Ft. Belvoir: Defence Technical Information Center*, March 1989, http://www.dtic.mil/get-tr-doc.

[Gro11] Gross, Andreas, A. Kremheller, and Hermann Fasel. "Simulation of Flow over Suboff Bare Hull Model", in *Proceedings of the 49th AIAA aerospace sciences meeting*, 2011.

[Gro92] W. D. Gropp, "Parallel computing and domain decomposition," in *Proceedings of the Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, PA, 1992.

[Gro96] W. D. Gropp, and A. Skjellum, "MPICH model MPI implementation reference manual", *Technical report, Argonne National Laboratory*, Argonne, 1996.

[Gro99] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface, vol. 1*, MIT press, 1999.

[Haf89] M. Hafez, J. Dacles, and M. Soliman, "A velocity/vorticity method for viscous incompressible flow calculations," in *Proceedings of the 11th International Conference on Numerical Methods in Fluid Dynamics*, pp. 288–296, 1989.

[Hal70] D. S. Halacy, *Charles Babbage, father of the computer*, Crowell-Collier Press, 1970.

[Han02] V. Hannemann, "Structured Multigrid Agglomeration on a Data Structure for Unstructured Meshes", *International Journal for Numerical Methods in Fluids*, 40, pp. 361-368, 2002.

[Han14] J. Hanke, P. Shankara, and D. Snyder, "Numerical Simulation of DLR-F11 High Lift Configuration from HiLiftPW-2 Using STAR-CCM+", 52nd Aerospace Sciences Meeting, AIAA SciTech, (AIAA 2014-0914), 2014.

[Har65] F. Harlow and J. Welsh, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface," *The Physics of Fluids*, 8(10), pp. 2182–2189, 1965.

[Hen95] B. Hendrickson and R. Leland, "A multi-level algorithm for partitioning graphs", in *Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, p. 28, San Diego, California, 3-6 December 1995.

[Hof14] J. Hoffman, J. Jansson, N. Jansson, and R. V. de Abreu, “Time-resolved adaptive FEM simulation of the DLR-F11 aircraft model at high Reynolds number”, In *Proceeding of 52nd Aerospace Sciences Meeting*, AIAA, 2014.

[Hua07] S. Huang, Q. S. Li, and S. Xu. "Numerical evaluation of wind effects on a tall steel building by CFD", *Journal of Constructional Steel Research*, 63(5), pp. 612-627, 2007.

[Hua89] T. T. Huang, H. L. Liu, and N.C. Groves, “Experiments of the DARPA SUBOFF program”, *Ft. Belvoir: Defence Technical Information Center*, December 1989, http://www.dtic.mil/get-tr-doc.

[Hus96] H. J. Hussein, and R. J. Martinuzzi. "Energy balance for turbulent flow around a surface mounted cube placed in a channel", *Physics of Fluids*, 8(3), pp. 764-780, 1996.

[Iac03] G. Iaccarino, A. Ooi, P. A. Durbin, and M. Behnia, “Reynolds averaged simulation of unsteady separated flow”, *International Journal of Heat and Fluid Flow*, 24(2), pp. 147-156, 2003.

[Imp04] J. Impagliazzo and J. A. N. Lee, “History of computing in education”, *in Proceedings of the IFIP 18th World Computer Congress, TC3/TC9, 51st Conference on the history of computing in education*, Toulouse, France, Springer Science & Business Media, 22-27 August 2004.

[Jan86] D. S. Jang, R. Jetli, and S. Acharya. "Comparison of the PISO, SIMPLER, and SIMPLEC algorithms for the treatment of the pressure-velocity coupling in steady flow problems", *Numerical Heat Transfer, Part A: Applications*, 10(3), pp. 209-228, 1986.

[Jes10] D. C. Jespersen, “Acceleration of a CFD code with a GPU,” *Scientific Programming*, 18(3–4), pp. 193–201, 2010.

[Joh99] T. A. Johnson, and V. C. Patel. "Flow past a sphere up to a Reynolds number of 300", *Journal of Fluid Mechanics*, 378, pp. 19-70, 1999.

[Kal05] Y. Kallinderis, and H.T. Ahn, “Incompressible Navier-Stokes Method with General Hybrid Meshes”, *Journal of Computational Physics*, 210, pp. 75-108, 2005.

[Kal96] Y. Kallinderis, “A 3-D Finite Volume Method for the Navier Stokes Equations with Adaptive Hybrid Grids”, *Applied Numerical Mathematics*, 20, pp. 387-406, 1996.

[Kar95] G. Karypis and V. Kumar, “Analysis of multilevel graph partitioning,” in *Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, 1999.

[Kar98] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” *SIAM Journal on scientific Computing*, 20(1), pp. 359–392, 1998.

[Kat09] A. Katz and A. Jameson,”Multicloud: Multigrid Convergence with a Meshless Operator”, *Journal of Computational Physics*, 228, pp. 5237-5250, 2009.

[Kim03] K. Kim, *Three-Dimensional Hybrid Grid Generator and Unstructured Flow Solver for Compressors and Turbines*, PhD thesis, Texas A&M University, 2003.

[Kim85] J. Kim and P. Moin, “Application of a fractional-step method to incompressible Navier-Stokes equations,” *Journal of Computational Physics*, 59(2), pp. 308–323, 1985.

[Kir01] C. Kiris and D. Kwak, “Numerical solution of incompressible Navier–Stokes equations using a fractional-step approach,” *Computers & Fluids*, 30(7), pp. 829–851, 2001.

[Koo00] B. Koobus, C. Farhat and H. Tran, “Computation of Unsteady Viscous Flows around Moving Bodies Using the k-ε Turbulence Model on Unstructured Dynamic Grids”, *Computer Methods in Applied Mechanics and Engineering*, 190, pp. 1441-1466, 2000.

[Kou03] D.G. Koubogiannis, A.N. Athanasiadis and K.C. Giannakoglou, “One- and Two-Equation Turbulence Models for the Prediction of Complex Cascade Flows Using Unstructured Grids”, *Computers and Fluids*, 32, pp. 403-430, 2003.

[Kue93] H. Kuerten and B. Geurts, “Compressible turbulent flow simulation with a multigrid multiblock method”, in *NASA CONFERENCE PUBLICATION*, pp. 305–305, 1993.

[Kwa11] D. Kwak and C. C. Kiris, *Computation of viscous incompressible flows*, Springer Science & Business Media, 2011.

[Kwa86] D. Kwak, J. L. C. Chang, S. P. Shanks, and S. R. Chakravarthy, “A three-dimensional incompressible Navier-Stokes flow solver using primitive variables,” *AIAA Journal*, 24(3), pp. 390–396, 1986.

[Lal88] M.H. Lallemand, “Etude de Schemas Runge-Kutta a 4 pas pour la Resolution Multigrille des Equations d’ Euler 2D”, Raport de Recherche, *INRIA*, 1988.

[Lam04] N.K. Lambropoulos, D.G. Koubogiannis and K.C. Giannakoglou, “Acceleration of a Navier-Stokes equation solver for unstructured grids using agglomeration multigrid and parallel processing”, *Computer Methods in Applied Mechanics and Engineering*, 193, pp. 781-803, 2004.

[Lan96] S. Lanteri, “Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies”, *Parallel Computing*, 22(7), pp. 943–968, 1996.

[Lan98] C.B. Laney, *Computational Gasdynamics*, Cambridge University Press, 1998.

[Lar91] B. Larrouturou, “How to Preserve Mass Fractions when Computing Compressible Multi-component Flows”, *Journal of Computational Physics*, 95, pp. 59-84, 1991.

[Lav75] S. H. Lavington, *A history of Manchester computers*, NCC Publications, 1975.

[LeC70] B. P. Le Clair, A. E. Hamielec, and H. R. Pruppacher. "A numerical study of the drag on a sphere at low and intermediate Reynolds numbers", *Journal of the Atmospheric Sciences*, 27(2), pp. 308-315, 1970.

[Lee81] K. D. Lee, and P. E. Rubbert, “Transonic flow computations using grid systems with block structure,” in *Proceedings of the Seventh International Conference on Numerical Methods in Fluid Dynamics*, pp. 266–271, 1981.

[Lee00] S. Lee, “A numerical study of the unsteady wake behind a sphere in a uniform flow at moderate Reynolds numbers”, *Computers & Fluids* 29(6), pp. 639-667, 2000.

[Lee10] E.M. Lee-Rausch, D.P. Hammond, E.J. Nielsen, S.Z. Pirzadeh and C.L. Rumsey, “Application of the FUN3D unstructured-grid Navier-Stokes solver to the 4th AIAA drag prediction workshop cases”, in *Proceedings of 28th AIAA Applied Aerodynamics Conference*, Chicago, Illinois, AIAA 2010-4551, 28 June - 1 July 2010.

[Lee15] E. M. Lee-Rausch, C. L. Rumsey, and M. A. Park, “Grid-Adapted FUN3D Computations for the Second High Lift Prediction Workshop”, *Journal of Aircraft*, 52(4), pp. 1098-1111, 2015.

[Lev10] D.W. Levy, K.R. Laflin, E.N. Tinoco, J.C. Vassberg, M. Mani, B. Rider, C. Rumsey, R.A. Wahls, J.H. Morrison, O.P. Brodersen, S. Crippa, D.J. Mavriplis and M. Murayama, “Summary of data from the fifth AIAA CFD drag prediction workshop”, in *Proceedings of 51st AIAA Aerospace Sciences Meeting*, Grapevine, Texas, AIAA 2013-0046, 7 -10 January, 2013.

[Lil92] D. K. Lilly, “A proposed modification of the Germano subgrid-scale closure method”, *Physics of Fluids A, Fluid Dynamics*, 4(3), pp. 633–635, 1992.

[Liu98] H.L. Liu, and T.T. Huang, “Summary of DARPA SUBOFF experimental program data”, *Ft. Belvoir: Defense Technical Information Center*, June 1998, http://handle.dtic.mil/100.2/ADA359226.

[LiuC98] C. Liu, X. Zheng, and C.H. Sung, “Preconditioned multigrid methods for unsteady incompressible flows”, *Journal of Computational Physics*, 139, pp. 35–57, 1998.

[Lyg12] G.N. Lygidakis and I.K. Nikolos, “Using the Finite-Volume Method and Hybrid Unstructured Meshes to Compute Radiative Heat Transfer in 3-D Geometries”, *Numerical Heat Transfer Part B: Fundamentals*, 62, pp. 289-314, 2012.

[Lyg13] G.N. Lygidakis and I.K. Nikolos, “Using a High-Order Spatial/Temporal Scheme and Grid Adaptation with a Finite-Volume Method for Radiative Heat Transfer”, *Numerical Heat Transfer Part B: Fundamentals*, 64, pp. 89-117, 2013.

[Lyg14a] G.N. Lygidakis and I.K. Nikolos, “Using a parallel spatial/angular agglomeration multigrid scheme to accelerate the FVM radiative heat transfer computation - part I: methodology”, *Numerical Heat Transfer B*, 66, 471-497, 2014.

[Lyg14b] G.N. Lygidakis, S.S. Sarakinos and I.K. Nikolos, “A parallel agglomeration multigrid method for incompressible flow simulations”, in *Proceedings of 9th International Conference on Engineering Technology*, Naples, Italy, 2-5 September, 2014, Civil-Comp Press, paper 27.

[Lyg14c] G.N. Lygidakis and I.K. Nikolos, “A parallel agglomeration multigrid method for the acceleration of compressible flow computations on 3D hybrid unstructured grid”, in *Proceedings of 11th World Congress on Computational Mechanics (WCCM XI), 5th European Conference on Computational Mechanics (ECCM V), 6th European Conference on Computational Fluid Dynamics (ECFD VI), IACM-ECCOMAS*, Barcelona, Spain, 20-25 July, 2014.

[Lyg14d] G.N. Lygidakis and I.K. Nikolos, “Using the DLR-F6 Aircraft Model for the Evaluation of the Academic CFD Code "Galatea"”, in *Proceedings of the International Mechanical Engineering Congress and Exposition, ASME-IMECE2014*, Montreal, Canada, 14-20 November 2014, IMECE2014-39756.

[Lyg15] G.N. Lygidakis, *On the numerical solution of compressible fluid flow and radiative heat transfer problems*, Ph.D. Thesis, Technical University of Crete, 2015.

[Lyg16] G. N. Lygidakis, S. S. Sarakinos, I. K. Nikolos, “Comparison of different agglomeration multigrid schemes for compressible and incompressible flow simulations”, *Advances in Engineering Software*, *Advances in Engineering Software*, Available online 13 January 2016, ISSN 0965-9978, http://dx.doi.org/10.1016/j.advengsoft.2015.12.00.

[Mag61] R. H. Magarvey, and R. L. Bishop. "Transition ranges for three-dimensional wakes", *Canadian Journal of Physics*, 39(10), pp. 1418-1422, 1961.

[Mav96] D.J. Mavriplis and V. Venkatakrishnan, “A 3D Agglomeration Multigrid Solver for the Reynolds-Averaged Navier-Stokes Equations on Unstructured Meshes”, *International Journal for Numerical Methods in Fluids*, 23, pp. 527-544, 1996.

[Mav97] D.J. Mavriplis, “Directional Coarsening and Smoothing for Anisotropic Navier-Stokes Problems”, *Electronic Transactions on Numerical Analysis*, 6, pp. 182-197, 1997.

[Mav98] D.J. Mavriplis, “Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes”, *Journal of Computational Physics*, 145, pp. 141-165, 1998.

[Mav99] D.J. Mavriplis and S. Pirzadeh, “Large-Scale parallel unstructured mesh computations for 3D high-lift analysis”, *NASA ICASE Report, No. 99-9*, 1999.

[Mav10] D.J. Mavriplis and M. Long, “NSU3D results for the fourth AIAA drag prediction workshop”, in *Proceedings of 28th AIAA Applied Aerodynamics Conference*, Chicago, Illinois, AIAA 2010-4550, 28 June - 1 July 2010.

[Mav15] D. Mavriplis, "NSU3D Results for the Second AIAA High Lift Prediction Workshop." *Journal of Aircraft*, 52(4), pp. 1063-1081, 2015.

[McC99] S. McCartney, *ENIAC: The triumphs and tragedies of the world’s first computer*, Walker & Company, 1999.

[McD71] P. W. McDonald, “The computation of transonic flow through two-dimensional gas turbine cascades,” in *Proceedings of ASME 1971 International Gas Turbine Conference and Products Show*, pp. V001T01A089–V001T01A089, 1971.

[Mel80] Melbourne, W. H. “Comparison of measurements on the CAARC standard tall building model in simulated model wind flows”, *Journal of Wind Engineering and Industrial Aerodynamics*, 6(1), pp. 73-88, 1980.

[Men92] F. R. Menter, “Influence of free stream values on k-ω turbulence model predictions”. *AIAA Journal*, 30(6), pp. 1657-1659, 1992.

[Men03] F. Menter, F.J. Carregal, T. Esch and B. Konno, “The SST turbulence model with improved wall treatment for heat transfer predictions in gas turbines”, *Proceedings of the International Gas Turbine Congress*, IGTC 2003-TS-059, Tokyo, Japan, 2 - 7 November, 2003.

[Moo65] G. Moore, “Cramming More Components Onto Integrated Circuits”, *Electronics*, (38) 8, pp. 114, 1965.

[Mou11] V. Moureau, P. Domingo, and L. Vervisch, “Design of a massively parallel CFD code for complex geometries,” *Comptes Rendus Mécanique*, vol. 339(2), pp. 141–148, 2011.

[MPI93] The MPI Forum, *MPI: A Message Passing Interface*, 1993.

[Mun90] B. R. Munson, D. F. Young, and T. H. Okiishi, *Fundamentals of fluid mechanics*, New York, 1990.

[Mur15] M. Murayama, K. Yamamoto, Y. Ito, T. Hirai, and K. Tanaka, “Japan Aerospace Exploration Agency Studies for the Second High-Lift Prediction Workshop”, *Journal of Aircraft*, 52(4), 1026-1041, 2015.

[Mur98] S. Murakami, "Overview of turbulence models applied in CWE–1997", *Journal of Wind Engineering and Industrial Aerodynamics*, 74, pp. 1-24, 1998.

[Nak87] K. Nakahashi and S. Obayashi, “Viscous flow computations using a composite grid,” in *Proceedings of 8th Computational Fluid Dynamics Conference*, Honolulu, HI, pp. 303–312, 1987.

[NASA] NASA/Langley Research Center, www.larc.nasa.gov.

[Nee97] R.E. Neel, *Advances in Computational Fluid Dynamics: Turbulent separated flows and transonic potential flows*, PhD Thesis, Blacksburg, Virginia, *USA*, 1997.

[Neu45] J. von Neumann, “First draft of a report on the EDVAC”, *1945*, Reprinted in *Papers of John von Neumann on Computing and Computer Theory*, W. Aspray and A. Burks, eds. MIT Press, Cambridge, MA, 1987.

[Nic99] F. Nicoud and F. Ducros, “Subgrid-scale stress modelling based on the square of the velocity gradient tensor,” *Flow, Turbulence and Combustion*, 62, pp. 183–200, 1999.

[Nis10] H. Nishikawa, B. Diskin and J.L. Thomas, “Critical Study of Agglomerated Multigrid Methods for Diffusion”, *AIAA Journal*, 48, pp. 839-847, 2010.

[Nis11] H. Nishikawa and B. Diskin, “Development and Application of Parallel Agglomerated Multigrid Methods for Complex Geometries”, in *Proceedings of the 20th AIAA Computational Fluid Dynamics Conference*, Honolulu, Hawaii, USA, 27 - 30 June, 2011, AIAA 2011-3232.

[Nish13] H. Nishikawa, B. Diskin, J.L. Thomas and D.P. Hammond, “Recent advances in agglomerated multigrid”, in *Proceedings of 51st AIAA Aerospace Sciences Meeting*, Grapevine, Texas, USA, 7 - 10 January, 2013, AIAA 2013-0863.

[Nou87] B. Nour-Omid, A. Raefsky, and G. Lyzenga, “Solving finite element equations on concurrent computers,” in *Proceedings of Parallel computations and their impact on mechanics*, Boston, MA, USA, 13-18 December, 1987.

[Oba92] E. D. Obasaju, "Measurement of forces and base overturning moments on the CAARC tall building model in a simulated atmospheric boundary layer", *Journal of Wind Engineering and Industrial Aerodynamics*, 40(2), pp. 103-126, 1992.

[Ors86] S. A. Orszag, M. Israeli, and M. O. Deville, “Boundary conditions for incompressible flows”, *Journal of Scientific Computing*, 1(1), pp. 75–111, 1986.

[Pan89] D. Pan and S. R. Chakravarthy, “Unified formulation for incompressible flows,” in *Proceedings of the 27th AIAA Aerospace Sciences Meeting*, 1, 1989.

[Pat72] S. V. Patankar, and D. B. Spalding, “A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows”, *International Journal of Heat and Mass Transfer*, 15(10), pp. 1787-1806, 1972.

[Pat86] D. A. Paterson, and C. J. Apelt. “Computation of wind flows over three-dimensional buildings”, *Journal of Wind Engineering and Industrial Aerodynamics*, 24(3), pp. 193-213, 1986.

[Qua13] L. Quartapelle, *Numerical solution of the incompressible Navier-Stokes equations, vol. 113*, Birkhauser, 2013.

[Ren09] T. C. S. Rendall and C. B. Allen, “Improved radial basis function fluid-structure coupling via efficient localized implementation”, *International Journal for Numerical Methods in Engineering*, 78(10), pp. 1188–1208, 2009.

[Rod86] W. Rodi, G. Scheuerer, “Scrutinizing the k–ε turbulence model under adverse pressure gradient conditions”, *Journal of Fluids Engineering*, 108, 1986.

[Roe81] P. Roe, “Approximate Riemann Solvers, Parameter Vectors and Difference Schemes”, *Journal of Computational Physics*, 43, pp. 357-371, 1981.

[Rog89] S. E. Rogers, D. Kwak, and C. Kiris, “Numerical solution of the incompressible Navier-Stokes equations for steady-state and time-dependent problems”, *in Proceedings of the Tenth Australasian Fluid Mechanics Conference*, University of Melbourne, 11-15 Dec., 1989.

[Rog90] S. E. Rogers and D. Kwak, “Upwind differencing scheme for the time-accurate incompressible Navier-Stokes equations,” *AIAA Journal*, 28(2), pp. 253–262, 1990.

[Ros91] M. Rosenfeld, D. Kwak, and M. Vinokur, “A fractional step solution method for the unsteady incompressible navier-stokes equations in generalized coordinate systems”, *Journal of Computational Physics*, 94(1), pp. 102–137, 1991.

[Ros92] C. C. Rossow, “Efficient computation of inviscid flow fields around complex configurations using a multiblock multigrid method,” *Communications in Applied Numerical Methods*, 8(10), pp. 735–747, 1992.

[Rud12] R. Rudnik, K. Huber, and S. Melber-Wilkending, “EUROLIFT Test Case Description for the 2nd High Lift Prediction Workshop”, *30th AIAA Applied Aerodynamics Conference, Fluid Dynamics and Co-located Conferences*, (AIAA Paper 2012-2924), 2012.

[Rud14] Rudnik, R., and S. Melber-Wilkending. “DLR Contribution to the 2nd High Lift Prediction Workshop”, *52nd Aerospace Sciences Meeting, AIAA SciTech*, (AIAA 2014-0915). 2014.

[Rum14P] C. L. Rumsey, J. P. Slotnick, “Overview and Summary of the Second AIAA High Lift Prediction Workshop”, in *Proceedings of AIAA Scitech 2014*, National Harbor, Maryland, 13 - 17 January, 2014.

[Sar14] S.S. Sarakinos, G.N. Lygidakis and I.K. Nikolos, “Evaluation of a Parallel Agglomeration Multigrid Finite-Volume Algorithm, named Galatea-I, for the Simulation of Incompressible Flows on 3D Hybrid Unstructured Grids”, in *Proceedings of the International Mechanical Engineering Congress and Exposition, ASME-IMECE2014*, Montreal, Canada, 14-20 November 2014, IMECE2014-39759.

[Sar15] S.S. Sarakinos, G.N. Lygidakis, I.K. Nikolos, "Acceleration Strategies for Simulating Compressible and Incompressible Flows", in P. Iványi, B.H.V. Topping, (Editors), *Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, Civil-Comp Press, Stirlingshire, UK, Paper 47, 2015.

[Sch70] H. A. Schwarz, *Ueber einen Grenzubergang durch alternirendes Verfahren*, Zurcher u. Furrer, 1870.

[Shi93] W. C. L. Shih, C. Wang, D. Coles, A. and Roshko, “Experiments on flow past rough circular cylinders at large Reynolds numbers”, *Journal of Wind Engineering and Industrial Aerodynamics*, 49, pp. 351–368, 1993.

[Shi01] S. Shin, *Reynolds-averaged Navier-Stokes computation on tip clearance flow in a compressor cascade using an unstructured grid*, Ph.D. Thesis, Virginia Polytechnic Institute and State University, 2001.

[Sma63] J. Smagorinsky, “General circulation experiments with the primitive equations”, *Monthly Weather Review*, 91(3), pp. 99–164, 1963.

[Smi04] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge: Cambridge University Press, 2004.

[Sor03] K.A. Sorensen, O. Hassan, K. Morgan and N.P. Weatherill, “A Multigrid Accelerated Hybrid Unstructured Mesh Method for 3D Compressible Turbulent Flow”, *Computational Mechanics*, 31, pp. 101-114, 2003.

[Str15] G. A. Strofylas, G.N. Lygidakis, I.K Nikolos, “Accelarating RBF-Based Mesh Deformation by Implementing An Agglomeration Strategy”, *in Proceedings of ASME International Mechanical Engineering Congress & Exposition (IMECE 2015)*, Houston, Texas, 2015, IMECE2015-50902.

[Ste77] J. L. Steger and P. Kutler, “Implicit Finite-Difference Procedures for the Computation of Vortex Wakes,” *AIAA Journal*, 15(4), pp. 581–590, 1977.

[Stok51] G. G. Stokes, *On the effect of the internal friction of fluids on the motion of pendulums, Vol. 9*. Pitt Press, 1851.

[Syk83] D. M. Sykes, “Interference effects on the response of a tall building model”, *Journal of Wind Engineering and Industrial Aerodynamics*, 11(1), pp. 365-380, 1983.

[Tai03] C. H. Tai, and Y. Zhao, “Parallel unsteady incompressible viscous flow computations using an unstructured multigrid method,” *Journal of Computational Physics*, 192, pp. 277-311, 2003.

[Tai05] C. H. Tai, Y. Zhao, and K. M. Liew, “Parallel-multigrid computation of unsteady incompressible viscous flows using a matrix-free implicit method and high-resolution characteristics-based scheme”, *Computer Methods in Applied Mechanics and Engineering*, 194(36–38), pp. 3949–3983, 2005.

[Tan56] S. Taneda, “Experimental investigation of the wake behind a sphere at low Reynolds numbers”, *Journal of the Physical Society of Japan*, 11(10), pp. 1104-1108, 1956.

[Tan86] H. Tanaka, and N. Lawen. "Test on the CAARC standard tall building model with a length scale of 1: 1000", *Journal of Wind Engineering and Industrial Aerodynamics* 25(1), pp. 15-29, 1986.

[Tay91] L. K. Taylor and D. L. Whitfield, “Unsteady three-dimensional incompressible Euler and Navier-Stokes solver for stationary and dynamic grids”, *22nd Fluid Dynamics, Plasma Dynamics and Lasers Conference, Fluid Dynamics and Co-located Conferences*, 1991.

[Tez92] T. E. Tezduyar, S. Mittal, S. E. Ray, and R. Shih, “Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements,” *Computer Methods in Applied Mechanics and Engineering*, 95(2), pp. 221–242, 1992.

[Thi09] J. C. Thibault and I. Senocak, “CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows,” in *Proceedings of the 47th AIAA aerospace sciences meeting*, pp. 2009–758, 2009.

[Tho93] J. F. Thompson and N. P. Weatherill, “Aspects of numerical grid generation: current science and art”, *11th Applied Aerodynamics Conference, Guidance, Navigation, and Control and Co-located Conferences*, 1993.

[Top500] http://www.top500.org/

[Tox08] S. Toxopeus, "Viscous-flow calculations for bare hull DARPA SUBOFF submarine at incidence", *International Shipbuilding Progress*, 55(3), pp. 227, 2008.

[Tur36] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", *Journal of the London Mathematical Society*, pp. 230–265, 1936.

[Van84] Van, D. J., and G. D. Raithby, "Enhancement of the SIMPLE method for predicting incompressible fluid flow", *Numerical Heat Transfer*, 7(2), pp. 147-163, 1984.

[Vas10] J.C. Vassberg, E.N. Tinoco, M. Mani, B. Rider, T. Zickuhr, D.W. Levy, O.P. Brodersen, B. Eisfeld, S. Crippa, R.A. Wahls, J.H. Morrison, D.J. Mavriplis and M. Murayama, "Summary of the fourth AIAA CFD drag prediction workshop", in *Proceedings of 28th AIAA Applied Aerodynamics Conference*, Chicago, Illinois, AIAA 2010-4547, 28 June - 1 July, 2010.

[Ven92] V. Venkatakrishnan, H. D. Simon, and T. J. Barth, "A MIMD implementation of a parallel Euler solver for unstructured grids," *The Journal of Supercomputing*, 6(2), pp. 117–137, 1992.

[Ven95] V. Venkatakrishnan, "Implicit schemes and parallel computing in unstructured grid CFD", in *Proceedings of 26th Computational Fluid Dynamics Lecture Series Program*, Von Karman Institute for Fluid Dynamics, Rhode Saint-Genese, Belgium,13-17 March, 1995.

[Ver07] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*, Pearson Education, 2007.

[VKa63] T. von Karman, *Aerodynamics*, Cornell University press: Mc Graw-Hill Company, 1963.

[Vos10] J.B. Vos, and S. Sanchi, , "DPW4 results using different grids including near-field/far-field drag analysis", in *Proceedings of 28th AIAA Applied Aerodynamics Conference*, Chicago, Illinois, AIAA 2010-4552, 28 June - 1 July, 2010.

[Vra12] S. Vrahliotis, T. Pappou, S. and Tsangaris, "Artificial compressibility 3-D Navier-Stokes solver for unsteady incompressible flows with hybrid grids", *Engineering Applications of Computational Fluid Mechanics*, 6(2), pp. 248-270, 2012.

[Wan08] X. Y. Wang, K. S. Yeo, C. S. Chew, and B. C. Khoo, "A SVD-GFD scheme for computing 3D incompressible viscous fluid flows", *Computers & Fluids*, 37(6), pp. 733-746, 2008.

[Wea88] N. P. Weatherill, "On the combination of structured-unstructured meshes", in *Proceedings of Numerical grid generation in computational fluid mechanics'88*, pp. 729–739, 1988.

[Weiss] E. W. Weisstein, "Tetrahedron", from *MathWorld* -- A Wolfram Web Resource. http://mathworld.wolfram.com/Tetrahedron.html.

[Wil89] R. D. Williams, “Supersonic fluid flow in parallel with an unstructured mesh,” *Concurrency: Practice and Experience*, 1(1), pp. 51–62, 1989.

[Xu92] Y. L. Xu, K. C. S. Kwok, and B. Samali, “Control of wind-induced tall building vibration by tuned mass dampers”, *Journal of Wind Engineering and Industrial Aerodynamics*, 40(1), pp. 1-32, 1992.

[Zha00] Y. Zhao, and B. Zhang, “A high-order characteristics upwind FV method for incompressible flow and heat transfer simulation on unstructured grids”, *Computer Methods in Applied Mechanics and Engineering*, 190(5–7), pp. 733–756, 2000.

[Zha15] Y. Zhang, W. G. Habashi, and R. A. Khurram. “Predicting wind-induced vibrations of high-rise buildings using unsteady CFD and modal analysis”, *Journal of Wind Engineering and Industrial Aerodynamics*, 136, pp. 165-179, 2015.