NATIONAL TECHNICAL UNIVERSITY OF ATHENS
School of Mechanical Engineering
Fluids Sector
Laboratory of Thermal Turbomachines
Parallel CFD & Optimization Unit

Contribution to the development of primal and adjoint CFD solvers for use in aerodynamic shape optimization.

Joint Postgraduate Course
''Computational Mechanics''

Master Thesis

Danai Chatzinikolaou

Supervisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2022

Abstract

This Master Thesis deals with the development of primal and adjoint solvers for use in aerodynamic shape optimization. It is divided in two main parts; the first one was conducted in Rolls-Royce Deutschland based on the in-house CFD solver HYDRA. It is a 3D steady solver, which solves the RANS equations, coupled with the Spalart-Almaras turbulence model, on unstructured body-fitted grids according to the vertex-centered finite-volume method. It is then coupled with a discrete adjoint solver to perform aerodynamic shape optimization. The second part is conducted at the National Technical University of Athens using a 2D steady solver which solves the Navier-Stokes equations on cartesian grids using the cell-centered finite-volume method and is coupled with a continuous adjoint solver.

In the first part, a source term is added to the primal 3D equations to alleviate limit cycle oscillations in case the solution is not stable and then the discrete adjoint problem is derived based on the new primal equations. The flow in a compressor vane is simulated with two different angles of attack, in which case the adjoint computes the sensitivity map.

In the second part, the existing flow solver is modified to handle viscous flows and assessed by simulating a flow over a flat plate and around a NACA0012 airfoil. The results are compared with literature data. Afterwards, the software is parallelised and its efficiency is evaluated. At the end, the continuous adjoint problem to this solver is developed for use in aerodynamic shape optimization. The equivalent adjoint boundary conditions are derived as imposed on a cartesian grid. The software is evaluated by optimizing the shape of two airfoils, the target being to maximize lift.

Acknowledgements

As this might be the last time I am writing acknowledgments, I would like to make this section as accurate as possible. This thesis has been a long journey for me, filled with ups and downs and I would like to remember every step and how much I changed throughout it. It has been written in three countries, Greece, Italy and Germany and I'm lucky I had amazing people around me throughout the way.

Starting with my biggest fans, my parents, Mpia and Giorgos. They deserve a massive thank you for their endless support and love. Everything I have achieved is because of them.

To continue, I would like to thank my professor Kyriakos Giannakoglou. If I can summarize our collaboration, I would say that he is constantly putting obstacles on my way and at the same time helping me overcome them. I'm thankful for everything he taught me, but I couldn't be more grateful to him for giving me the biggest opportunity of my life, which brought me where I am now.

Next, I would like to thank Konstantinos Samouchos for our amazing collaboration. He is both an excellent mentor and a very good friend.

A big thank you to my next mentor Ilias Vasilopoulos, who has been a huge part of the second half of the journey of this thesis. I'm grateful for all his patience, support and guidance, and I know that without him, this thesis wouldn't have been as it is.

For the final part I left all of my dearest friends who were always there for me through thick and thin, making me the best version of myself. To begin, I would like to thank Maro, Christina, Kyriaki, Dimitris, Ilias and Dimitris, who made my years as a student memorable. Next, my beloved teammates, Zina, Filio, Danai, Chrysa, Anais, with whom I've shared endless memories of joy and happiness, and at the same time we have been there for each other through our best and worst times. To finish, I owe a big thank you to Maria, Kimon and Tasos for everything we went through during the master program, I wish only the best for them. And last but not least I would like to thank Moritz, who despite the short time we've known each other, has been an amazing friend.

Abbreviations

CFD	Computational Fluid Dynamics
LTT	Laboratory of Thermal Turbomachines
NTUA	National Technical University of Athens
PCopt	Parallel CFD & Optimization unit
RRD	Rolls-Royce Deutschland
GC	Ghost-Cells
MPI	Message Passing Interface
ABC	Adjoint Boundary Condition
FAE	Field Adjoint Equations
SD	Sensitivity Derivatives
OGV	Outlet Guide Vane
RANS	Reynold-Averaged Navier-Stokes
RMS	Root-Mean-Square
RHS	Right Hand Side

Contents

Chapter 1

Introduction

Throughout its history, fluid dynamics has been a field of continuous development. One of the major steps in the subject was in the 18th century, when Leonhard Euler introduced the equations of acceleration of a steady, irrotational flow under gravitational action [13], which set the base of modern fluid dynamics. To follow, Louis Marie Henri Navier in 1822 [27] made the first attempt to extend the equations on viscous fluids, which combined with G.G. Stokes' study in 1842-1851 [46], formulated the widely known "Navier-Stokes" equations.

In the years that followed, the Navier-Stokes equations have been studied thoroughly in order to derive analytical solutions (L. Prandlt [38], H. Blasius [8]) and account for additional physical phenomena (O. Reynolds [39]). The second breakthrough came around the middle of the 20th century with the development of high-speed computers and introduced a new scientific field, the "Computational Fluid Dynamics" (CFD). The computer, combined with the extensive development of numerical methods to solve physical problems, allowed for faster and inexpensive analysis of flow fields compared to the limited and costly experiments performed in the wind tunnel until that time.

In parallel to fluid dynamics, another field which has always attracted the interest of many scientists, the theory of optimization. Hence, the pursue of a problem's (mathematical or not) optimal solution, with respect to specific criteria. Starting with optimization of dedicated geometrical problems, such as finding the minimum distance between two points, and continuing with the development of the theory of Calculus of Variations, a field of mathematical analysis that uses variations to find minima and maxima of functions. While the beginning of the subject cannot be precisely dated, scientists like I.Newton, L.Euler, G.W. von Leibniz, A.L. Cauchy, C.Jacobi and many more showed immediate interest and developed new principles and theories within the

subject. [18] Based on the Calculus of Variations the first optimization algorithms emerged during the 19th century and, in the 20th century, thanks to the continuous development of computers, stochastic optimization methods arise and gradient-based methods prosper.

1.1 CFD-based Optimization within the Industry

Today, optimization tools combined with CFD analysis are widely used in the industry, in a variety of sectors including automotive, aerospace, chemical and medical research and have revolutionized the products' design process. Even though there are only a few dedicated studies, the introduction of CFD analysis in the industry has significantly reduced the cost of product development, the need for experimental analysis, the time of optimization activities and it has helped to achieve improved, detailed and reliable design of products. One study on the benefit of CFD application in a chemical and engineered-material company showed that in a six year period the financial benefit of using CFD has been six times greater than the amount spent on CFD activities. ([11])

Taking a look at some of today's biggest companies in the aerospace and automotive sector, CFD based optimization has become a fundamental design tool. Rolls-Royce plc has developed an in-house CFD solver, HYDRA [51], to study flows around complex geometries inside the gas turbine. The tool is handling unsteady 3D flows in compressor and turbine cascades and is mainly used to assess the aerodynamic efficiency of turbomachinery components. Combined with a gradient-based optimization algorithm it has become a powerful tool for the company's engine design. General Electric Aviation has also developed an in-house solver, TACOMA [3], to perform aerodynamic analysis and shape optimization in turbomachinery blades for jet engines. It is a 3D unsteady CFD solver using RANS equations and the $k-\omega$ turbulence model implemented in FORTRAN 90. There has been a lot of effort on making the software parallel using a tri-hybrid parallelization approach, namely, a combination of MPI, OpenMP and OpenACC. In addition to the in-house solvers, both companies utilize commercial CFD software for simulating heat transfer, aero-acoustics, aero-mechanics etc. Coming back in Europe, in 2018 Airbus announced collaboration with both the german and french aerospace research centers DLR and ONERA in order to enhance their CFD knowledge and capability on designing aerodynamically efficient aircrafts [2]. The product of the collaboration is the high fidelity second order finite volume CFD solver, CODA, developed in C++ which is aimed to simulate external aerodynamics during the aircraft's normal and complex manoeuvres [50]. Within the automotive sector, Tesla Motors has been using the commercial STAR-CCM+, CFD solver developed by Siemens, to perform aerodynamic simulations around electric vehicles. The tool

has assisted to optimize the energy consumption and the range performance of the vehicles, which is currently one of the biggest challenges in the sector [45]. However, industry has always been a competitive environment. Different companies working in the same sector are developing and selling similar products. For this reason each company has to find the balance between delivering a product fast and having the highest quality possible. Naturally, as CFD analysis has become an essential tool in today's industry, the challenge for a CFD developer is to create a competitive software to be able to fit in the market; hence, a software which produces accurate results in the shortest time possible.

1.2 Adjoint Optimization

Due to the big and complex geometries and flowfields that the industrial applications deal with, CFD analysis can be very expensive and time-consuming. This is why CFD solvers are in the majority of the times combined with gradient-based optimization algorithms rather than stochastic ones. One of the most popular gradient based optimization methods is the adjoint method. In simple terms, the adjoint method is a method of computing the objective function sensitivities with respect to the design variables by demanding the elimination of terms that are expensive to compute (i.e. the derivatives of the flow quantities w.r.t. the design variables). The method will further be discussed in the next sections.
The method was first introduced by Pironneau in 1984 [37] for aerodynamic applications and was further developed by Jameson [19] Giles and Pierce [15] [36]. Originally, it was used for problems of external aerodynamics, such as flows around airfoils and wings [6]. Later on, Yang et al. [55] applied the adjoint method for flows in 2D cascades, governed by the compressible Euler equations, and Chung et al. [10] performed an adjoint-based inverse design of the 3D Rotor 37 case. Eventually, there have been several applications on the shape optimization of single-row turbomachinery using the adjoint method, such as by Wu et al. [54] and Papadimitriou and Giannakoglou [32] [33]. Adjoint solvers that supported multi-row applications appeared quite later; prominent first examples are the contributions of Frey et al. [14] and of Wang and He [53].
The major benefit of the adjoint method is that the cost of computing the gradient of the objective function is almost equal to the cost of one CFD solver run, in contrast to other gradient computing methods, such as finite differences, for which this cost is equal to N CFD solver runs, with N being the number of design variables. The cost of the adjoint method is proportional to the number of objective functions F instead, which is beneficial for CFD-based shape optimization problems, where the design variables significantly outnumber the objective functions.

One of the method's challenges is that it requires a second mathematical problem to be derived and solved, which must be consistent with the primal problem formulation. If the primal problem changes, then the adjoint method should be re-formulated. The same applies in case of changes to the objective function; any modification of the function of interest, requires once again development of the adjoint problem.

1.3 Thesis Background and Overview

This thesis deals with adjoint methods for aerodynamic shape optimization in industrial and other applications. It is divided in two parts; the first one deals with cases where the adjoint solver is unable to converge due to instabilities in the flow solution, and the second part is about developing the continuous adjoint method of an existing CFD solver which handles cartesian grids and uses the ghost-cell method to impose boundary conditions.

In particular, the first part of the thesis was conducted in Rolls-Royce Deutschland, a subsidiary of British aircraft engine maker Rolls-Royce plc, based in Dahlewitz, Germany. The work has been carried on Rolls-Royce's CFD solver, HYDRA. The subject of the analysis is to find a method to make the adjoint equations converge, in cases which the primal CFD solver is reaching limit cycle oscillations; an idea developed by Paolo Adami within the Methods Team of RRD. The first step is to introduce a new source term to the primal equations in order to make the solver converge to the mean value of the two extreme states of the limit cycle. In the next step, the source term is modified accordingly and introduced to the adjoint equations. The purpose is to make the adjoint problem converge into a steady state solution, to be used to compute the sensitivity derivatives of the objective function w.r.t the design variables. Combined with steepest descent, the design solution is changing towards the direction where the objective function is minimized, until it reaches an optimal design.

The second part of the thesis was conducted at NTUA using an in-house CFD solver [44]. In the first part, the existing 2D steady CFD solver is being parallelized using the MPI protocol. Next, the adjoint CFD solver using the continuous adjoint method is created based on the primal solver, which uses the immersed boundary method to solve the primal equations on cartesian grids. One of the biggest benefits of the immersed boundary methods on cartesian grids can be seen on applications where the geometry is moving with time. In such cases, with the traditional body-fitted mesh approach, a new mesh would have to be created at each timestep. However, the new mesh is not always of a good quality, especially as the shape gets more complex during optimization. As such, a CFD optimization tool could possibly produce non-reliable results. For an industrial application,

this would mean an unprofitable amount of time spent and that the simulation would need to be restarted from the beginning. With an immersed boundary CFD solver such scenarios cannot occur, as the mesh remains almost unchanged throughout unsteady simulations.

The Master thesis consists of 4 chapters, which are summarized below:

- Chapter 1 - Introduction
- Chapter 2 - 3D Flow Equations. As both CFD solvers described herein are used for aerodynamic applications, they solve the Navier-Stokes equations or a form of them. For this reason, the governing equations will be presented in the beginning in their most general form and later on, on each separate chapter, the equations will be recast accordingly.
- Chapter 3 - Part 1: Improvement of Adjoint Convergence Robustness for Steady CFD Applications. In this chapter, the HYDRA steady solver is outlined together with the discrete adjoint method. The method used to alleviate limit cycle oscillations is described, and the modifications made to the primal and adjoint solver are demonstrated. Finally, the results of the method are presented and analyzed in order to conclude on following steps.
- Chapter 4 - Part 2: Programming and Parallelization of a Flow Solver and the Continuous Adjoint Method using the Ghost-Cell Method. Applications in Aerodynamic Shape Optimization. In this chapter, the in-house steady solver of NTUA is outlined and modified to handle viscous flows. The software is parallelised using the MPI protocol. Finally, the adjoint system is derived from the primal equations, using the continuous adjoint method, and used to perform aerodynamic shape optimization.
- Chapter 5 - Summary and conclusions. A summary and overview of the methods and results is presented together with conclusions and potential next steps.

Chapter 2

Steady 3D Navier-Stokes Equations and Adjoint Optimization

As stated in the previous section, this thesis deals with two CFD based optimization tools. Both tools numerically solve the Navier-Stokes equations, in 3D or 2D, by using the adjoint method to compute the sensitivity derivatives of the objective function w.r.t the design variables.

In this chapter, the general form of the Steady 3D Navier- Stokes equations is presented in order to establish a reference system for all CFD calculations performed herein. In the following sections, these equations will be modified accordingly to adapt to each CFD solver. Furthermore, the general theory of the adjoint optimization method will be presented, to establish a common terminology. The adjoint equations will be presented for each software separately.

At the end of the chapter, the adjoint optimization theory will be outlined. The method is divided into two main parts, the continous and the discrete approach, each of them having its benefits and weaknesses. Each of the CFD solvers used herein, is combined with one of the two methods and for this reason, the methods will further be analyzed later in the thesis.

2.1 General form of the steady 3D Navier-Stokes equations

Observing an infinitesimally small element fixed in space with compressible fluid moving through it, the conservation form of the Navier-Stokes is formulated as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = S_c \tag{2.1}$$

$$\frac{\partial (\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v} \cdot \vec{v}^{\mathrm{T}} + p \overleftrightarrow{I} - \overleftrightarrow{\tau}) = \vec{S_m} \tag{2.2}$$

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho \vec{v} E + p \vec{v} - \overleftrightarrow{\tau} \cdot \vec{v} - k \nabla T) = S_e \tag{2.3}$$

where ρ is the density of the fluid, $\vec{v} = [u, v, w]^{\mathrm{T}}$ the Cartesian velocity, p the static pressure, $\overleftrightarrow{\tau}$ the stress tensor, E the relative total energy per unit mass, T the static temperature, k the thermal conductivity coefficient, $\overleftrightarrow{I}$ the identity matrix and $S_c, \vec{S_m}, S_e$ are source terms for each equation. Throughout this thesis only steady flows are considered and the equations will be numerically solved using the time-marching technique. For this reason, t refers to the pseudo-time.

The system is closed by assuming that the thermodynamic properties of the fluid satisfy the equation of perfect gas

$$p = \rho R T \quad \Rightarrow \quad c^2 = \gamma \frac{p}{\rho} \tag{2.4}$$

where R is the perfect gas constant, γ the heat capacity ratio and c the speed of sound. The total energy per unit mass E is given by

$$E = \frac{1}{\gamma - 1} \frac{p}{\rho} + \frac{1}{2} \vec{v}^2 \tag{2.5}$$

Assuming an isotropic Newtonian fluid and Stokes' hypothesis for the viscosity, the stress tensor components are given by

$$\tau_{ij} = \mu \left[\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) - \frac{2}{3} \delta_{ij} \frac{\partial v_k}{\partial x_k} \right] \tag{2.6}$$

with μ being the dynamic viscosity and δ_{ij} the Kronecker delta. Hereafter, twice repeated indices imply summation over $k = 1, 2, 3$ (according to the Einstein's convention).

The Navier-Stokes equation can be written in vector form as

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{F}^{inv}(\vec{U}) + \nabla \cdot \vec{F}^{visc}(\vec{U}) = \vec{S} \tag{2.7}$$

where $\vec{U} = [\rho, \rho u, \rho v, \rho w, \rho E]$ are the conservative flow variables. The flux vectors $\vec{F}^{inv}$ and $\vec{F}^{visc}$ include the inviscid and viscous terms of the Navier-Stokes equations respectively. In particular

$$\vec{F}^{inv} = \begin{bmatrix} \rho\vec{v} \\ \rho\vec{v}u + p\vec{e}_x \\ \rho\vec{v}v + p\vec{e}_y \\ \rho\vec{v}w + p\vec{e}_z \\ \rho\vec{v}E + p\vec{v} \end{bmatrix}, \qquad \vec{F}^{visc} = -\begin{bmatrix} 0 \\ \vec{\tau}_1 \\ \vec{\tau}_2 \\ \vec{\tau}_3 \\ \vec{\tau}_k v_k + q_k \end{bmatrix}, \qquad \vec{S} = \begin{bmatrix} S_c \\ S_{m1} \\ S_{m2} \\ S_{m3} \\ S_e \end{bmatrix} \tag{2.8}$$

where $\vec{e} = [\vec{e}_x, \vec{e}_y, \vec{e}_z]^T$ are the unit vector in the x, y, z directions respectively, $\vec{\tau}_k = [\tau_{k1}, \tau_{k2}, \tau_{k3}]^{\mathrm{T}}$ and, from Fourier's law of heat conduction, $\vec{q} = k\nabla T$.

2.1.1 Steady 3D RANS equations & Turbulence modeling

For the flow within a jet engine or around a car, the equations should be modified to account for the presence of turbulence. One of the most common methods to simulate turbulence was developed by O. Reynolds in 1895 [39] and is known as the Reynold-Averaged Navier-Stokes equations (RANS). The idea behind this method is that all the fluctuations of the flow variables, such as pressure or velocity, happen around a mean value. As a result, the variables are divided in a time-averaged and a fluctuating quantity which are inserted in the Navier-Stokes equations. By solving the RANS equations the time-averaged value of every variable is derived. This method inevitably introduces new variables to quantify the turbulence in the flow and need to be further modeled.

The RANS equations have the same form as the 3D Navier-Stokes in equations (2.1) - (2.3) with the only difference that the flow variables i.e. density, velocity and pressure are now time-averaged. Turbulence is modeled through the turbulent viscosity μ_t, as proposed by Boussinesq, using the hypothesis that when averaging the NS equations over time, the non-linear term resulting from turbulent fluctuations in velocity can be written as an additional diffusion term [9]. Thereby, the turbulent viscosity is introduced in the stress tensor of eq. (2.9) as

$$\tau_{ij} = (\mu + \mu_t)\left[\left(\frac{\partial \upsilon_i}{\partial x_j} + \frac{\partial \upsilon_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \upsilon_k}{\partial x_k}\right] \tag{2.9}$$

The Spalart-Allmaras Turbulence Model

For the applications presented in this thesis, the Spalart-Allmaras (SA) turbulence modeling is used and the turbulent viscosity is defined as

$$\mu_t = \rho\tilde{\nu} f_{v1}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu} \tag{2.10}$$

where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity. The SA variable $\tilde{\nu}$ is computed by the standard form of the model's equation [4]

$$\begin{aligned} \frac{\partial \tilde{\nu}}{\partial t} + \underbrace{\frac{\partial(\tilde{\nu}\upsilon_k)}{\partial x_k}}_{Conv} &= \underbrace{\frac{1}{\sigma}\left[\frac{\partial}{\partial x_k}\left((\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_k}\right) + c_{b2}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right]}_{Diff} \\ &+ \underbrace{c_{b1}\tilde{\Omega}\tilde{\nu}}_{Prod} - \underbrace{(c_{w1} f_w)\left(\frac{\tilde{\nu}}{d}\right)^2}_{Destr} \end{aligned} \tag{2.11}$$

where term *Conv* expresses the transport of $\tilde{\nu}$, term *Diff* the molecular and turbulent diffusion, terms *Prod*, *Destr* model the turbulence production and destruction.

Regarding the rest of the quantities appearing in eq. (2.11), $\tilde{\Omega}$ is the modified vorticity

$$\tilde{\Omega} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \tag{2.12}$$

$$\Omega = \sqrt{\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right)^2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}$$

where Ω is the magnitude of the vorticity and d the distance to the nearest wall. The function f_w is given by

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

and the closure constants are

$$c_{b1} = 0.1355, \qquad c_{b2} = 0.622, \qquad \sigma = \frac{2}{3}, \qquad c_{v1} = 7.1,$$
$$c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}, \qquad c_{w2} = 0.3, \qquad c_{w3} = 2, \qquad \kappa = 0.41$$

2.2 Adjoint Optimization

In this section, an introduction of the adjoint optimization method is given. More detailed analysis as well as the mathematical formulation is given in later sections. As mentioned before, solving the 3D Steady Navier-Stokes or RANS equations for industrial applications is time-consuming. For this reason, such CFD solvers are usually combined with gradient-based optimization methods, which are significantly faster than stochastic optimization algorithms, in order to avoid adding further computational cost. This thesis deals with gradient-based optimization methods, by making use of the adjoint method.

In their general form, adjoint methods are for computing the derivatives of an objective function, and at the same time ensuring that the problem's governing equations are satisfied. The optimization problem is viewed as a constrained problem; the design variables vector $\vec{b}$ that minimizes the objective function F is sought with the constraint of satisfying the governing equations i.e. the primal problem, which in the purposes of this thesis, are the Navier-Stokes equations. The objective function can, in general, be written as

$$F = F(\vec{U}, \vec{b}) \tag{2.13}$$

where $\vec{U}$ is the vector of the flow variables. For an infinitesimal change $\delta\vec{b}$ in the design variables' vector $\vec{b}$ the change in the objective function is

$$\delta F = \frac{\partial F}{\partial \vec{U}} \delta\vec{U} + \frac{\partial F}{\partial \vec{b}} \delta\vec{b} \tag{2.14}$$

Additionally the governing flow equations can, in general, be written as

$$\vec{R} = \vec{R}(\vec{U}, \vec{b}) = 0 \tag{2.15}$$

And again for an infinitesimal change $\delta\vec{b}$ in the design variables' vector $\vec{b}$ the

change in the governing equations vector is

$$\delta\vec{R} = \frac{\partial\vec{R}}{\partial\vec{U}}\delta\vec{U} + \frac{\partial\vec{R}}{\partial\vec{b}}\delta\vec{b} = 0 \tag{2.16}$$

The adjoint problem is formulated by introducing an augmented form of the objective function using a Lagrangian multiplier $\vec{\Psi}^T$ multiplied by the $\vec{R}$ vector.

$$F_{aug} = F + \vec{\Psi}^T\vec{R} \tag{2.17}$$

The augmented function is now differentiated w.r.t the design variables $\vec{b}$ using equations 2.14 and 2.16

$$\delta F_{aug} = \frac{\partial F}{\partial\vec{U}}\delta\vec{U} + \frac{\partial F}{\partial\vec{b}}\delta\vec{b} - \vec{\Psi}^T\left(\frac{\partial\vec{R}}{\partial\vec{U}}\delta\vec{U} + \frac{\partial\vec{R}}{\partial\vec{b}}\delta\vec{b}\right) \tag{2.18}$$

and by rearranging the terms

$$\delta F_{aug} = \left(\frac{\partial F}{\partial\vec{U}} - \vec{\Psi}^T\frac{\partial\vec{R}}{\partial\vec{U}}\right)\delta\vec{U} + \left(\frac{\partial F}{\partial\vec{b}} - \vec{\Psi}^T\frac{\partial\vec{R}}{\partial\vec{b}}\right)\delta\vec{b} \tag{2.19}$$

By demanding the elimination of the most expensive terms, namely the variation of the flow variables, the adjoint equations are derived

$$\frac{\partial F}{\partial\vec{U}} - \vec{\Psi}^T\frac{\partial\vec{R}}{\partial\vec{U}} = 0 \tag{2.20}$$

and by solving 2.20 the derivative of the objective function is computed as

$$\frac{\delta F_{aug}}{\delta\vec{b}} = \frac{\partial F}{\partial\vec{b}} - \vec{\Psi}^T\frac{\partial\vec{R}}{\partial\vec{b}} \tag{2.21}$$

There are two main categories of adjoint methods in CFD applications, the continuous and the discrete method. In the continuous adjoint [21], the governing equations are first differentiated w.r.t. the design variables and then discretized on the available grid. The reverse process is followed in discrete adjoint [16] [22], where the flow equations are first discretized and then, their differentiation leads to the system of adjoint equations already in discrete form. The advantages and disadvantages of each approach have been extensively

discussed in the literature [35] [26], and it is generally agreed that both approaches are equally efficient on computing gradients in CFD-based optimization problems. Furthermore, there are many different approaches to formulating the continuous and discrete adjoint method, and the details of their implementation result in each having advantages and disadvantages. This thesis deals with both continuous and discrete adjoint methods. More details about the theory and formulation of each method will be discussed separately in the later sections.

Chapter 3

Improvement of Adjoint Convergence Robustness for Steady CFD Applications

The first part of the thesis was conducted in Rolls-Royce Deutschland using the in-house software, HYDRA [51]. HYDRA is a CFD solver for unsteady 3D flows used within the turbine and compressor design teams. In this work, the 3D steady RANS equations are coupled with the Spalart-Allmaras turbulence model as presented in chapter 2 and are solved iteratively, using a Runge-Kutta pseudo-time stepping scheme. The software is combined with a discrete adjoint CFD solver to perform aerodynamic shape optimization. The adjoint system is solved with a, consistent with the primal, Runge-Kutta solver to derive the sensitivities of the objective functions [28], [52]. The sensitivity derivatives can then be inserted in an optimization tool, for example using the steepest descent method to obtain the new design variables, which define a new geometry, and the process starts again. After a number of optimization cycles, the new optimized geometry results.

In this first part of the MSc thesis, a method to alleviate such limit cycle oscillations is developed in order to allow for further shape optimization process. In particular, a source term is introduced into the primal equations which aims to act as an artificial force to the system with the purpose to allow the primal problem to converge to a mean solution. From a primal stable and converged solution, the adjoint system can more easily converge and allow for the computation of the sensitivity derivatives and the optimization.

In what follows, an overview of HYDRA will be presented including the primal and adjoint equations. The primal equations will be first discretized on an unstructured grid, according to the vertex-centered finite-volume method and the boundary conditions will be imposed in order to close the system. Then, the discrete adjoint theory and method will be presented. Further on, the source term will be introduced together with the modifications in the primal and adjoint equations. The new system of equations will be used to solve the flow over a compressor vane to verify the convergence of the primal solution. The resulting flow field from the primal solver will be used to solve the adjoint problem for the same case. Finally, the adjoint solution will be used to calculate the sensitivity derivatives of the objective function, which in this case is the total pressure loss coefficient over the vane.

3.1 Discretization of the Flow Equations

The primal equations solved within HYDRA are the 3D RANS equations 2.1-2.3 combined with the Spalart-Allmaras model 2.11 as described in Section 2.1.1 The RANS equations are discretized on an unstructured grid using vertex-centered control volumes, as illustrated for a 2D case in figure 3.1.

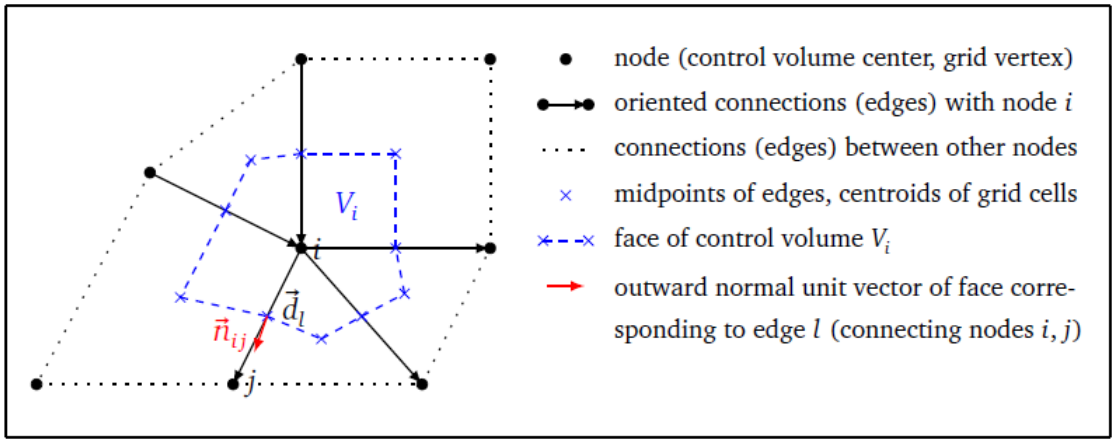


Figure 3.1: *Vertex-centered control volume in a 2D unstructured grid [52].*

Integrating eq. 2.7 over the control volume V_i and applying the Green-Gauss theorem gives the RANS equations expression on node i

$$V_i \frac{\partial \vec{U}}{\partial \vec{Q}_i} \frac{\partial \vec{Q}_i}{\partial t} + \vec{R}_i = \vec{S}_i \tag{3.1}$$

where $\vec{U}$ is the conservative flow variables vector, $\vec{Q}$ is the primitive flow variables

vector, t is the pseudo-time $\vec{R_i}$ is the residual at node i and S_i are the source terms.

$$R_i = \sum_{j \in E_i} \left(\vec{F^{inv}}_{ij} + \vec{F^{visc}}_{ij} \right) \cdot \vec{n}_{ij} \Delta s_{ij} \tag{3.2}$$

where E_i is the set of nodes connected to node i via an edge, n_{ij} the unit vector vertical to the edge connecting nodes i and j and Δs_{ij} the area of the face associated with the same edge. To simplify the notation, the oriented face area is defined as $\vec{s}_{ij} = \vec{n}_{ij} \Delta s_{ij}$.

3.1.1 Inviscid Fluxes

The inviscid fluxes $\vec{F^{inv}}_{ij}$ across the interface between control volumes V_i, V_j are computed based on the Roe scheme [42] by a combination of a central difference term with a numerical dissipation term $\vec{F}_{ij}^{ND}$:

$$\sum_{j \in E_i} \vec{F}_{ij}^{inv} \cdot \vec{s}_{ij} \;=\; \sum_{j \in E_i} \frac{1}{2} \left(\vec{F}_i^I + \vec{F}_j^I \right) \cdot \vec{s}_{ij} \;+\; \sum_{j \in E_i} \vec{F}_{ij}^{ND} \cdot \vec{s}_{ij} \tag{3.3}$$

The addition of the numerical dissipation term $\vec{F}_{ij}^{ND}$ ensures numerical stability of the solver method. Based on Roe's approximation [42] this term becomes

$$\vec{F}_{ij}^{ND} \cdot \vec{s}_{ij} = -\frac{1}{2} \frac{\partial \vec{U}_{ij}}{\partial \vec{Q}_{ij}} \left| \underline{\underline{A}}_{ij} \right| \Delta s_{ij} \Delta Q_{ij} \tag{3.4}$$

where $\underline{\underline{A}}_{ij}$ is the primitive flux Jacobian and its absolute value is defined as $\underline{\underline{T}} \left| \underline{\underline{\Lambda}} \right| \underline{\underline{T}}^{-1}$, with $\left| \underline{\underline{\Lambda}} \right|$ being the diagonal matrix of absolute eigenvalues (of $\underline{\underline{A}}$) and $\underline{\underline{T}}$ the corresponding matrix of right eigenvectors. The term is computed as

$$\underline{\underline{A}}_{ij} = \frac{\partial \vec{U}}{\partial \vec{Q}_{ij}} \frac{\partial \left(\vec{F}^{inv} \cdot \vec{n} \right)}{\partial \vec{Q}} \bigg|_{ij} \tag{3.5}$$

where the primitive variables are $Q_{ij} = \frac{1}{2} \left(Q_i + Q_j \right)$.

The formulation of the difference ΔQ_{ij} in eq. (3.4) determines the numerical accuracy of the scheme. For example if $\Delta Q_{ij} = Q_j - Q_i$, the scheme becomes first-order accurate in space. Herein a non-linear blend between the first-order upwind and a second-order scheme with third-order numerical dissipation fluxes is used based on the Jameson-Schmidt-Turkel (JST) scheme [20].

3.1.2 Viscous Flux

The viscous flux $\vec{F}_{ij}^{V}$ at the control volume face associated with the edge connecting nodes i and j, is computed using equation 2.8 combined with the Spalart-Allmaras turbulence model. The term becomes

$$\vec{F}_{ij}^{visc} = -\begin{bmatrix} 0 \\ \vec{\tau}_{1,ij} \\ \vec{\tau}_{2,ij} \\ \vec{\tau}_{3,ij} \\ \left(\vec{\tau}_k \upsilon_k\right)_{ij} + \frac{\gamma}{\gamma-1}\left(\frac{\mu_{ij}}{Pr_l} + \frac{\mu_{t,ij}}{Pr_t}\right)\left(\frac{1}{\rho_{ij}}\nabla p_{ij} - \frac{p_{ij}}{\rho_{ij}^2}\nabla \rho_{ij}\right) \\ \left(\mu_{ij} + \frac{\mu_{t,ij}}{\sigma_\phi}\right)\nabla \phi_{ij} \end{bmatrix} \tag{3.6}$$

For computing the primitive flow variables, the arithmetic average is used, whereas for the turbulent viscosity, the geometric average is preferred because of its exponential growth near viscous walls. Hence:

$$Q_{ij} = \frac{1}{2}\left(Q_i + Q_j\right) \tag{3.7}$$

$$\mu_{ij} = \frac{1}{2}\left(\mu\left(Q_i\right) + \mu\left(Q_j\right)\right) \tag{3.8}$$

$$\mu_{t,ij} = \sqrt{\mu_t\left(Q_i\right)\mu_t\left(Q_j\right)} \tag{3.9}$$

Discretizing the inviscid and viscous flux terms, as discussed in this section, leads to an overall second-order spatial discretization scheme, which is used throughout this thesis.

3.1.3 Boundary Conditions

Walls

For the nodes laying along the solid walls, zero mass flux is imposed by setting the velocity components and the turbulent viscosity to zero. The discrete coupled RANS equations for the solid wall are

$$\begin{aligned} \left(I - B\right)\vec{r}_{wall} &= 0 \\ B\vec{u}_{wall} &= 0 \end{aligned} \tag{3.10}$$

where I the identity matrix and $\vec{r}_{wall}$ the residual vector of the wall nodes. For the 3D compressible flow the 5-equation system 2.1-2.3 considered, along with a one-equation turbulence model, and so the B matrix extracts the components of velocity and turbulent viscosity from wall nodes on which the boundary conditions are imposed.

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

The solid wall boundaries are considered adiabatic, so the wall heat flux is set to zero.

Inlet/Outlet

For the inlet, the total pressure, total temperature, flow angles and the turbulent viscosity whereas at the exit the static pressure are imposed. These boundary conditions are weakly imposed using the inviscid boundary flux across the boundary interface $\vec{s_b}$ in eq. 3.3

$$\vec{F}_b^{inv}\vec{s}_b = \frac{1}{2}\left(\left(\vec{F}^{inv}\left(\vec{Q}_{in/out}\right) + \vec{F}^{inv}\left(\vec{Q}_b\right)\right)\vec{s}_b - \frac{\partial\vec{U}}{\partial\vec{Q}}\left|\bar{A}_b\right|\left(\vec{Q}_b - \vec{Q}_{in/out}\right)\right) \tag{3.12}$$

where subscript b denotes the boundary face and $Q_{in/out}$ refers to the pseudo-nodes outside of the domain, which are used to impose the boundary conditions. The outlet static pressure is computed at the hub radius and its radial distribution from hub to tip is derived by solving the radial equilibrium equation

$$\frac{\partial p}{\partial r} = \rho\frac{v_\theta^2}{r} \tag{3.13}$$

where r is the radius and v_θ the circumferential component of the velocity. Within HYDRA, a target mass flow rate can be chosen as an outlet boundary condition. Here, the mass flow rate through the outlet cross-section A_{out} is

$$\dot{m} = \int_{A_{out}} \rho u dA \tag{3.14}$$

The target mass flow is achieved by updating the outlet static pressure p_{hub} at each iteration as

$$\Delta p_{hub} = (\dot{m}^{tar} - \dot{m}) \left(\frac{d\dot{m}}{dp} \right)^{-1} \tag{3.15}$$

where $\frac{d\dot{m}}{dp}$ is hand-derived within the code.

3.1.4 Implicit Runge-Kutta scheme

At this point the set of RANS equations 3.1 is re-written here

$$V_i \frac{\partial \vec{U}}{\partial \vec{Q}}_i \frac{\partial \vec{Q}_i}{\partial t} \; + \; \vec{R}_i \left(\vec{Q} \right) = S_i \tag{3.16}$$

$$R_i \left(\vec{Q} \right) = \sum_{j \in E_i} \frac{1}{2} \vec{F}^{inv} \cdot \vec{s}_{ij} + \sum_{j \in E_i} \vec{F}^{ND}_{ij} \cdot \vec{s}_{ij} + \sum_{j \in E_i} \vec{F}^{visc}_{ij} \cdot \vec{s}_{ij} \tag{3.17}$$

The implicit RK scheme in Hydra was developed by Misev [24], based on the work of Swanson et al. [48, 49] and further developed by Vasilopoulos [52]. In the general form, the scheme can be written as

$$\frac{V}{\Delta t} \frac{\partial \vec{U}}{\partial \vec{Q}} \left(\vec{Q}^{n+1} - \vec{Q}^{n} \right) = -R \left(\vec{Q}^{n+1} \right) \tag{3.18}$$

where n denotes the pseudo-time step. After the system is discretized in pseudo-time is solved iteratively using a time-marching technique for the update of the primitive variables u. The final form of the implicit RK 3-step scheme derived from 3.18

$$\vec{Q}^{(n,k)} \; = \; \vec{Q}^{n} - \alpha_k \sigma \left(\frac{1}{\sigma_{imp}} \underline{P}^{n} + \underline{J}^{n} \right)^{-1} \vec{R}^{(n,k)}, \qquad k = 1, \ldots, s \tag{3.19}$$

where J^n is the Jacobian matrix, P^n the preconditioner, α_k the primary RK coefficients, σ_{imp} the implicit CFL number, here $\sigma_{imp} = 2$ and s the number of RK steps, here $s = 3$ [52].

By expanding equation 3.19 [52]

$$\vec{Q}^{(n,0)} = \vec{Q}^{n} \tag{3.20}$$

$$\vec{R}^{D(n,k)} = \beta_k \vec{R}^{D}\left(\vec{Q}^{(n,k-1)}\right) + (1-\beta_k)\vec{R}^{D(n,k-1)} \tag{3.21}$$

$$\vec{Q}^{(n,k)} = \vec{Q}^{n} - \alpha_k \sigma\left(\underline{K}^{n}\right)^{-1}\left(\vec{R}^{I}\left(\vec{Q}^{(n,k-1)}\right) + \vec{R}^{D(n,k)} - V\vec{S}\left(\vec{Q}^{(n,k-1)}\right)\right) \tag{3.22}$$

$$\vec{Q}^{n+1} = \vec{Q}^{(n,s)} \tag{3.23}$$

where $k = 1, 2, 3$, $\underline{K}$ is the implicit preconditioner matrix

$$\underline{K} = \frac{1}{\sigma_{imp}}\underline{P} + \underline{J} \tag{3.24}$$

and R^I and R^D the inviscid and diffusive components computed using the current estimate of $\vec{Q}$. The source term S will be used to introduce the artificial force to control the convergence of the primal solver, and will be discussed in the section 3.3.

3.2 Hydra discrete adjoint solver

In this section, the discrete adjoint theory will be presented [15] and applied to derive a, consistent to the primal, adjoint Runge-Kutta scheme in order to solve the adjoint equations to further compute the derivatives of the objective function. The aim of solving the primal equations is to drive the non-linear residual within each control volume to zero

$$\vec{R}(\vec{U}, \vec{X}) = 0 \tag{3.25}$$

where $\vec{R} = [\vec{R}_1, ..., \vec{R}_i, ..., \vec{R}_N]$ is the residual of all nodes 3.2, $\vec{U} = [\vec{U}_1, ..., \vec{U}_i, ..., \vec{U}_N]$ the primitive flow variables and $\vec{X} = [\vec{x}_1, ..., \vec{x}_i, ..., \vec{x}_N]$ the coordinates of all (N) grid nodes within the computational domain. To perform gradient-based optimization in a problem with an objective function F, the derivative of F w.r.t the design variables must be computed. This is a function of the flow solution U as well as the grid coordinates X, which are functions of the design variables $\vec{b}$, all of this leading to the relation

$$F = F(\vec{U}(\vec{X}(\vec{b})), \vec{X}(\vec{b})) \tag{3.26}$$

Hence the gradient of the objective function F can be written using the chain rule as

$$\frac{dF}{d\vec{b}} = \frac{dF}{d\vec{X}}\frac{d\vec{X}}{d\vec{X}_s}\frac{d\vec{X}_s}{d\vec{b}} \tag{3.27}$$

where X_s are the coordinates of the surface grid nodes. Using the adjoint method, the term $\frac{dF}{d\vec{X}}$, which stands for the volume sensitivity derivatives or simply sensitivity derivatives, is calculated. First step of the adjoint method is to create an augmented objective function

$$F_{aug} = F - \vec{\Psi^T}\vec{R} \tag{3.28}$$

where Ψ is the adjoint variables vector. The derivative of eq. 3.28 w.r.t $\vec{X}$ is now

$$\frac{dF_{aug}}{d\vec{X}} = \frac{dF}{d\vec{X}} - \vec{\Psi^T}\frac{dR}{d\vec{X}} - \frac{d\vec{\Psi^T}}{d\vec{X}}\cancelto{0}{\vec{R}} \tag{3.29}$$

$$= \frac{\partial F}{\partial \vec{X}} + \frac{\partial F}{\partial \vec{U}}\frac{d\vec{U}}{d\vec{X}} - \vec{\Psi^T}\left(\frac{\partial \vec{R}}{\partial \vec{X}} + \frac{\partial \vec{R}}{\partial \vec{U}}\frac{d\vec{U}}{d\vec{X}}\right) \tag{3.30}$$

$$= \frac{\partial F}{\partial \vec{X}} - \vec{\Psi^T}\frac{\partial \vec{R}}{\partial \vec{X}} + \left(\frac{\partial F}{\partial \vec{U}} - \vec{\Psi^T}\frac{\partial \vec{R}}{\partial \vec{U}}\right)\frac{d\vec{U}}{d\vec{X}} \tag{3.31}$$

The main idea now is to avoid computing the most expensive term $\frac{d\vec{U}}{d\vec{X}}$, and for this reason its multiplier is set to zero. This yield to the discrete adjoint equation

$$\left[\frac{\partial \vec{R}}{\partial \vec{U}}\right]^T \Psi = \left[\frac{\partial F}{\partial \vec{U}}\right]^T \tag{3.32}$$

After the adjoint field is computed, the sensitivity derivatives are computed as

$$\frac{dF}{d\vec{X}} = \frac{\partial F}{\partial \vec{X}} - \vec{\Psi^T}\frac{\partial \vec{R}}{\partial \vec{X}} \tag{3.33}$$

In general, it is preferred that the adjoint solver converges with the same rate as the primal, which is called duality preserving [17]. Without going into further detail, as it is beyond the scope of this thesis, the equivalent to the primal, implicit Runge-Kutta scheme to solve the adjoint equations used herein [52].

An overview of the implicit adjoint solver algorithm implemented in Hydra is shown in figure 3.2.

Algorithm 3: Steady implicit adjoint solver.

read in $U^{N^{iter}}$;
compute objective_adj $\rightarrow G$;
compute implicit preconditioner K;
construct transpose matrix K^{T};
initialize Ψ, $iter \rightarrow 0$;
while $(iter < iter_{max})$ **and** $(R_a^{RMS} > \epsilon)$ **do** // pseudo-time loop
  $stage_{RK} \rightarrow 3$;
  while $stage_{RK} \geq 1$ **do** // 3-stage Runge-Kutta loop
    impose BCs_adj;
    compute fluxes_adj $\rightarrow \mathrm{C}^{\mathrm{T}}\Psi, \mathrm{D}^{\mathrm{T}}\Psi$;
    compute residual R_a;
    solve $\mathrm{K}^{\mathrm{T}}\tilde{\Psi} = R_a$; // linear solver nested loop
    update solution $\Psi \rightarrow \Psi + \alpha_{RK}\sigma\tilde{\Psi}$;
    $stage_{RK} \rightarrow stage_{RK} - 1$;
  end
  $iter \rightarrow iter + 1$;
end

Figure 3.2: *Overview of Hydra's implicit adjoint solver algorithm [52].*

More detail on how the Runge-Kutta system is derived can be found in [28] [52].

3.3 Brute Force Method for cases with Limit Cycle Oscillation

Limit-cycle oscillation (LCO) is a fixed-amplitude oscillation appearing in the convergence diagram of the governing equations due to non-linear phenomena in the system. The amplitude of the oscillations remains the same regardless of the initial conditions. One example of LCO behaviour in a converging equation is shown in figure 3.3. The details of the case will be presented in the next section.

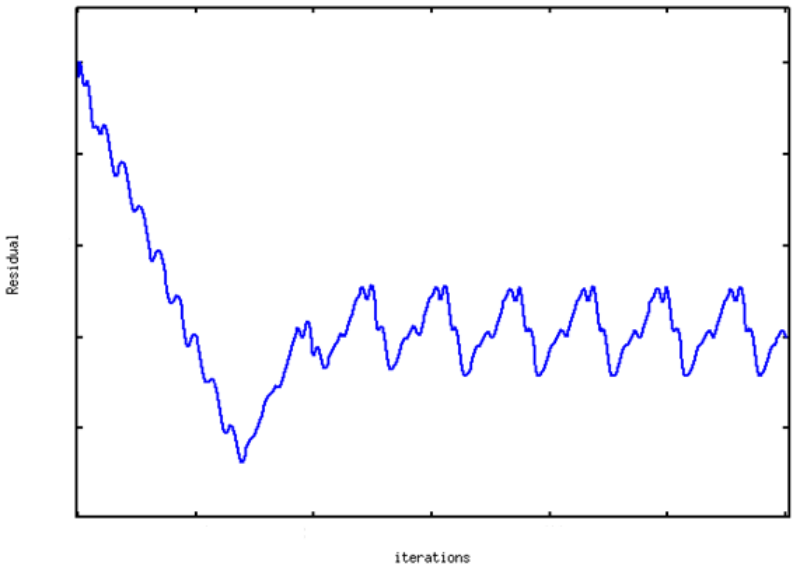


Figure 3.3: *RMS residual of the primal equations.*

The solution of the equation is oscillating between two extreme values. And while this behaviour could be negligible when it comes to an already converged solution of the primal (flow) equations, e.g. the two extreme values oscillate between 10^{-5} and 10^{-7}, it is of utmost importance when it comes to solving the adjoint equations after that. The convergence of the adjoint problem is extremely sensitive to the convergence of the primal problem's solution. For the same case, which exhibited LCO in the solution of the primal problem, the convergence diagram of the RMS residual of the adjoint equations. is shown in figure 3.4.

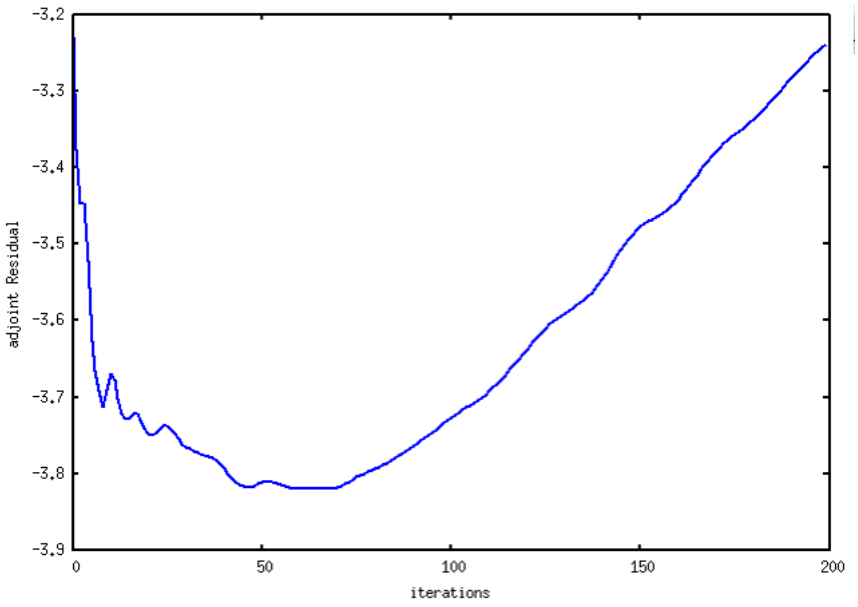


Figure 3.4: *RMS residual of the adjoint equations.*

One can see that the adjoint problem is not converging when the primal problem is exhibiting oscillations as in figure 3.3.

In order to fix this problem, an idea developed by Paolo Adami within the Methods Department of RRD is introduced, called Brute Force Method for cases with Limit Cycle Oscillation convergence. In what follows, the method will be referred to as "the brute force method". The basic principle is to introduce a term in the primal equations which will eliminate the oscillations without affecting the final solution and which, after being differentiated, will make the adjoint problem converge. The source term introduced in the governing equations is equivalent to a "force" and will drive the residual of the primal equations to converge into the mean value of the two extreme states of the limit cycle. The source term is differentiated accordingly and introduced into the adjoint equations, which will then be used to compute the sensitivity derivatives of the objective function w.r.t the design variables.

The first step of the process is to obtain the mean value of the local flow variables, within N iterations, as shown in figure 3.5, using the equation

$$\bar{u} = \frac{\sum_{i=1}^{N} v_i}{N} \tag{3.34}$$

The choice of N will be made in the next section.

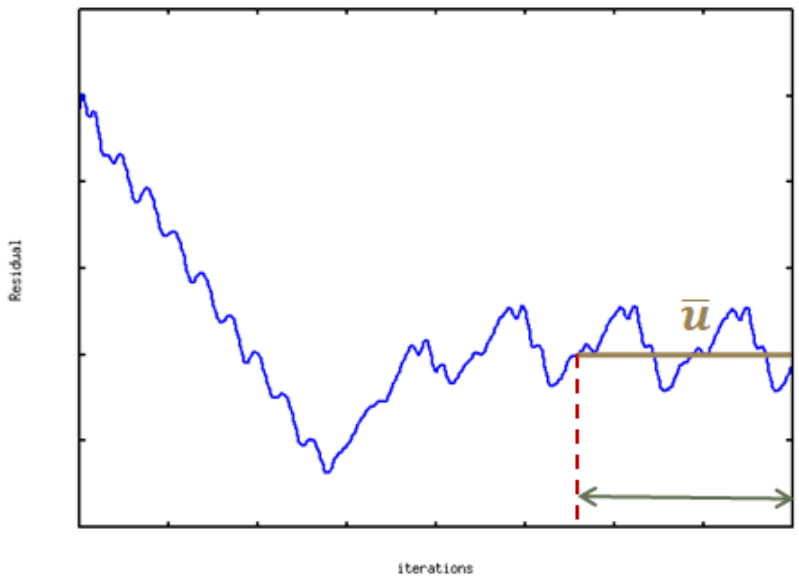


Figure 3.5: *Calculation of mean value of flow solution.*

To avoid saving any additional flow solution other than the last one, the sum $\sum_{i=1}^{N} v_i$ is calculated while the solver is running and for the last N iterations. At the end of the simulation, the sum is divided by N to derive the mean value, which is calculated only once throughout the simulation.

Now the purpose of the source term is to force the primal solution to converge fully to a single solution rather than oscillating between two. The solution which the equations are driven to converge to, is the mean value of the LCO. This means that the source term should be formulated so that, the areas where the solution is far away from the mean value, have a bigger source term whereas the areas where the solution is close to the mean value, have a small value. While there are other methods and numerical techniques to drive the system to converge to its mean value and a consequently seeking convergence of the adjoint solver [29], the brute force method was preferred due to its simplicity both in mathematical formulation as well as in software developing. The source term is introduced into the RHS of eq. 2.7 in the form

$$\vec{S_f} = \frac{1}{\tau} \begin{bmatrix} \rho - \bar{\rho} \\ u - \bar{u} \\ v - \bar{v} \\ w - \bar{w} \\ E - \bar{E} \end{bmatrix} \tag{3.35}$$

where the 'bar' indicates the mean value of the variable calculated from eq. 3.34 and τ is a parameter with units of time, used in the denominator to scale the source term to fit in the primal equations. The choice of τ will be made after parametric study in the next section. The term is discretised and introduced into the RHS of eq. 3.16 as

$$S_{fi} = \frac{u_i - \bar{u}_i}{\tau} V_i \tag{3.36}$$

After the discretization of the system, the source term 3.36 is part of the residual R in equation 3.19. An additional component is added to the diagonal terms of the Jacobian J of the system, resulting from the term

$$\frac{\partial S_{fi}}{\partial u_i} = \frac{V_i}{\tau} \tag{3.37}$$

The same term is added to the adjoint equation 3.32 to derive the adjoint variables.

In the beginning, the user defines an acceptable residual for convergence and a maximum number of iterations for the solver to run. In cases with LCO, the residual does not meet the convergence criteria and, therefore, reaches the maximum iteration limit. However, during the last N iterations, the mean value

of the oscillating solution is being computed. Now, instead of stopping the simulation, the source term, which includes the mean value, is switched on and the 'new' governing equations are solved again until the convergence criteria is reached.

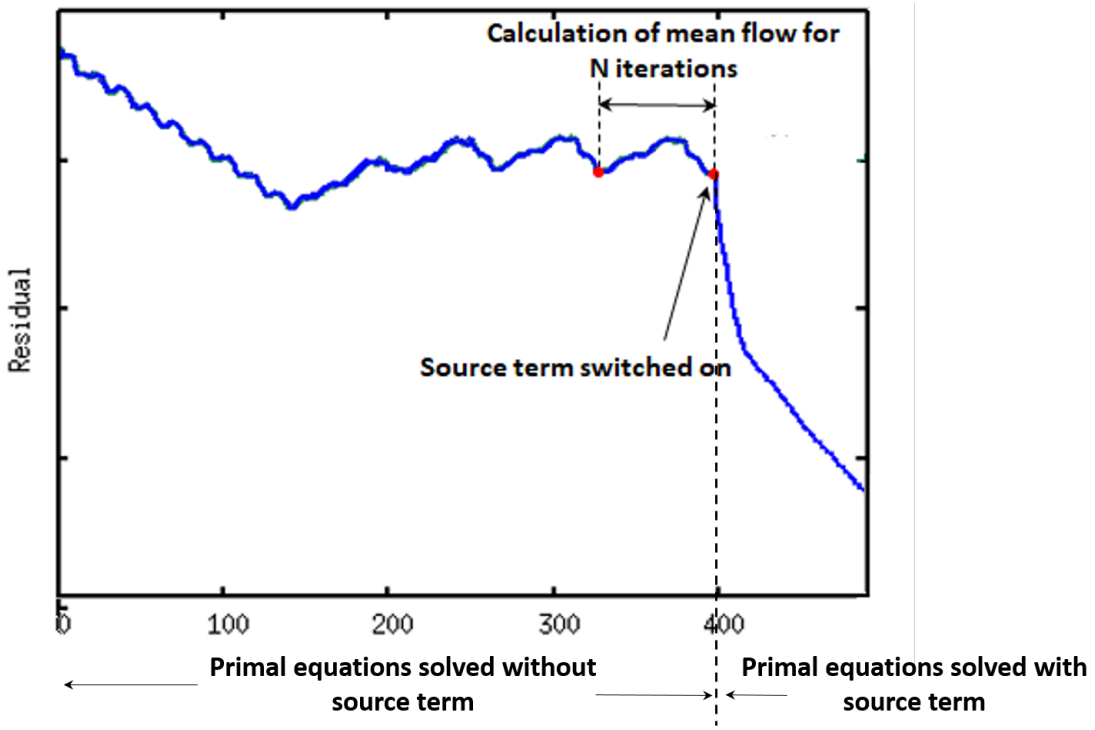


Figure 3.6: *Brute Force Method Overview*

3.4 Brute Force method assessment

The geometry used to evaluate the aforementioned method, is a compressor stator, designed in the Technical University of Berlin (TUB) in Germany. The design is based on a representative outlet guide vane (OGV) of modern jet engines and is shown in figure 3.7 .

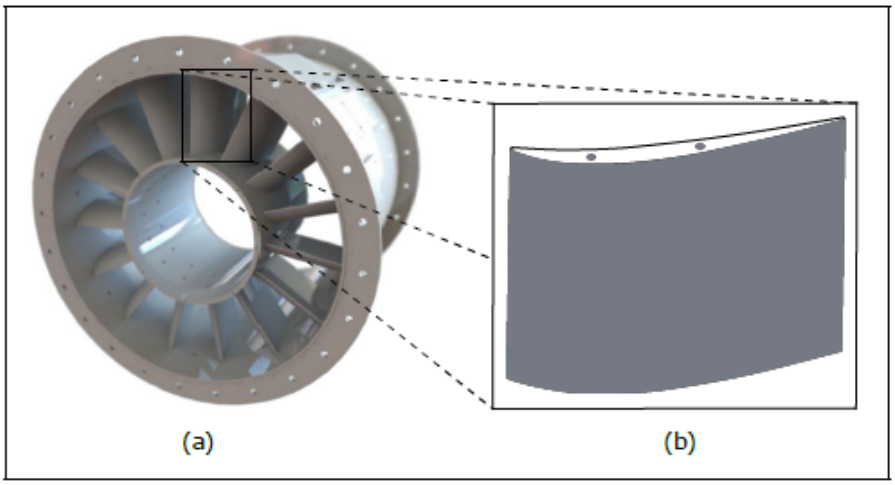


Figure 3.7: *Compressor OGV designed at the Chair of Aero Engines of TUB.(a) Photo of rig test (b) 3D shape [52]*

.

The computational mesh around the stator was created using the Rolls-Royce in-house tool PADRAM (PArametric Design and Rapid Meshing [23]. The resulting body-fitted mesh has around $2 \cdot 10^6$ nodes and is a combination of an O-type grid around the blade together with four H-type grids for the rest of the domain. A section of the grid is shown in figure 3.8

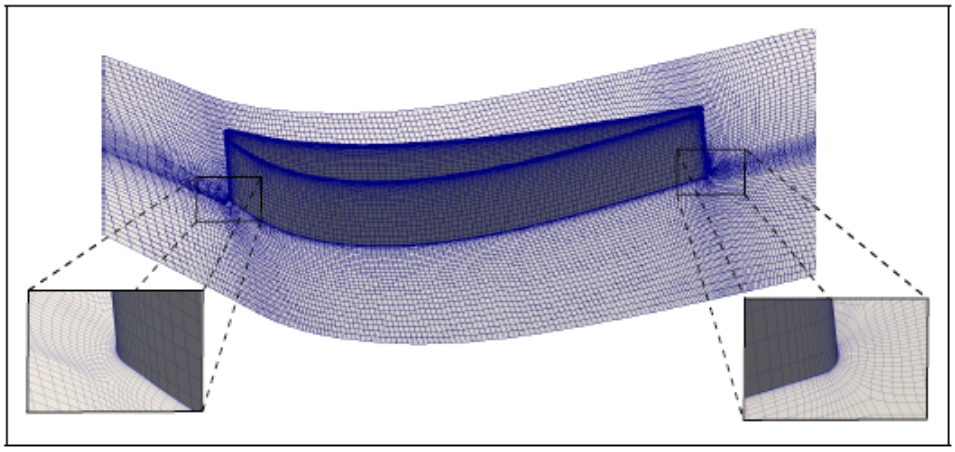


Figure 3.8: *Mesh around compressor OGV at midspan section. [52]*

The objective function of interest is the total pressure losses between the inlet and outlet of the CFD domain, defined by the pressure loss coefficient

$$\omega = \frac{p_{t,inlet} - p_{t,outlet}}{p_{t,inlet}} \tag{3.38}$$

Two cases are simulated using this geometry. The OGV is first subjected to flow with inlet flow angle of 46^o which is 4^o greater than its design case. In what follows, the case will be referred as '$+4^o$ case'. For this flow, the primal problem is converging normally, without oscillations. The adjoint problem is also converging and the sensitivity map can be derived. In order to evaluate the brute force method, a case in which the primal flow exhibits LCO must be tested. One such case is when the OGV is subjected to flow with inlet flow angle by 5^o higher than its design value. In what follows, the case will be referred as '$+5^o$ case'. Here, the primal problem is exhibiting LCO and the adjoint problem is thus diverging. The brute force method is then introduced in the second case to help the adjoint convergence.

As there are no available data to validate the results of the $+5^o$ case, the code is validated against the $+4^o$ case.

Study case: $+4^o$ off-design

Primal problem

At first the vane was subjected in airflow of 4^o higher than its design case. For this case, both the primal and adjoint problem converge without the need of the source term. The Root-mean-squared residual (RMS) of the primal and adjoint equations is shown in figures 3.9 and 3.10 respectively.

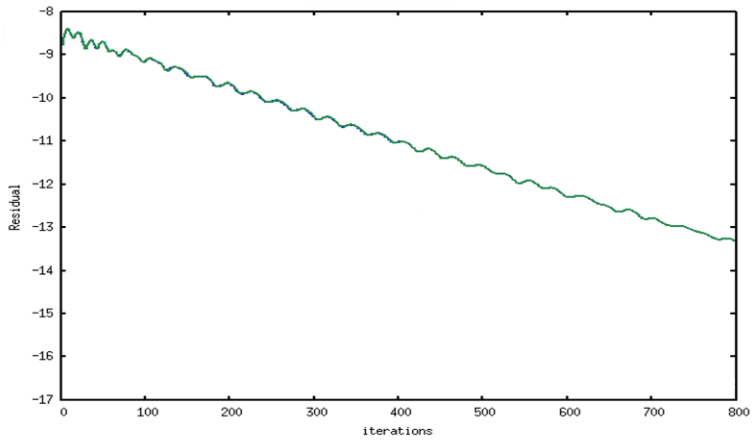


Figure 3.9: *Case +4. RMS residual of the primal equations without source term.*

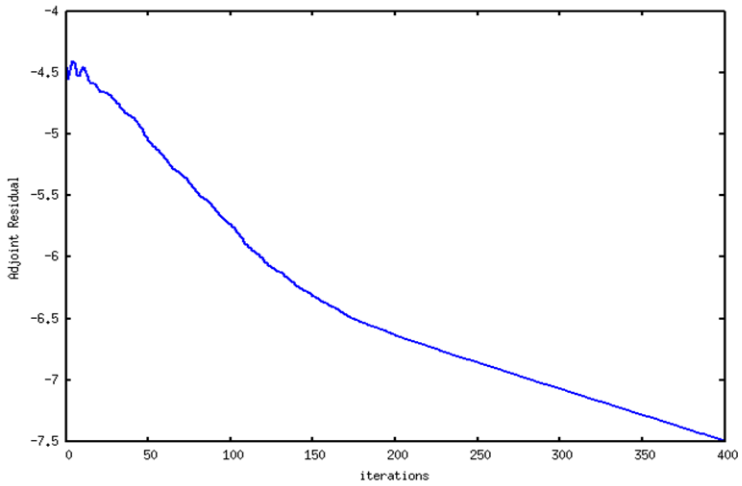


Figure 3.10: *Case +4. RMS residual of the adjoint equations without source term.*

The first step of the method is to compute the mean value of an oscillating CFD solution based on eq. 3.34. Assuming that M is the maximum number of

iterations that the solver is running set by the user in the beginning of the simulation and N is the amount of iterations used to derive the mean flow solution. For the parametric study, the N value was set to $5\%M$, which means that the mean value will be computed for the last $5\%M$ iterations. Next, the number was increased to $10\%, 15\%, 20\%$ and $25\%M$ to see how the average mean solution changes. By increasing the iterations to $25\%M$, the average mean solution only changes for about 1% which is considered to be small. For this reason, the final value of N was set to $20\%M$, which for this problem is 80 iterations.

N **(%max iterations)**	**Delta % from previous**
5%	-
10%	6.8%
15%	4.3%
20%	2.1%
25%	1.1%

Table 3.1: *Change of the mean flow solution with respect to the number of samples-flow solutions being averaged.*

Second step is to derive the parameter τ from the equation 3.36. As stated earlier, it is a parameter with units of time and its value is derived after a parametric study, based on how much the RMS residual of the pirmal equations including the source term, changes. The case used for the parametric study is the $+4^o$ case, where the value of τ was set to $3, 5, 7$ and 10. The RMS residual of the primal equations after several runs, is shown in figure 3.11. The chosen value for τ is $\tau = 7$, because for numbers higher than 7, the residual doesn't change significantly.

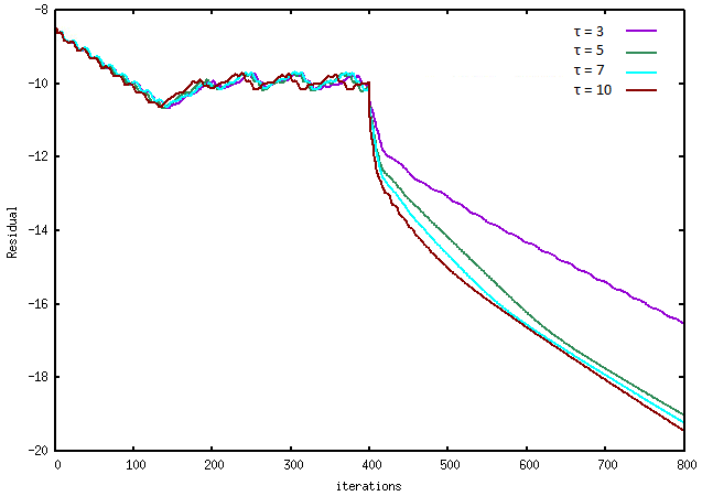


Figure 3.11: *Parametric study to derive τ constant. RMS residual of primal equations including the source term. Purple: $\tau = 3$, Green: $\tau = 5$, Blue: $\tau = 7$, Brown: $\tau = 10$*

After choosing $\tau = 7$ the $+4^o$ case is simulated with the new solver. A comparison of the RMS residual of the primal equations with and without the source term is shown in figure 3.12.

Figure 3.12: *Case +4. Comparison of primal equations' convergence with and without source term. Green - without source term. Blue - with source term.*

For $M = 800$ max iterations set by the user, one can see the source term being switched on after the first 400 iterations. Afterwards, the equations continue to be solved for another 400. During the first 400 iterations, the residual was having some small oscillations, although the mean value was reducing with time. After the introduction of the source term, the oscillations vanish and the equations converge in a faster rate. It is important to mention that the source term tends to zero as the simulation goes on, because the solution approaches the mean value.

The final solution of the flow variables is shown in figures 3.13-3.16.

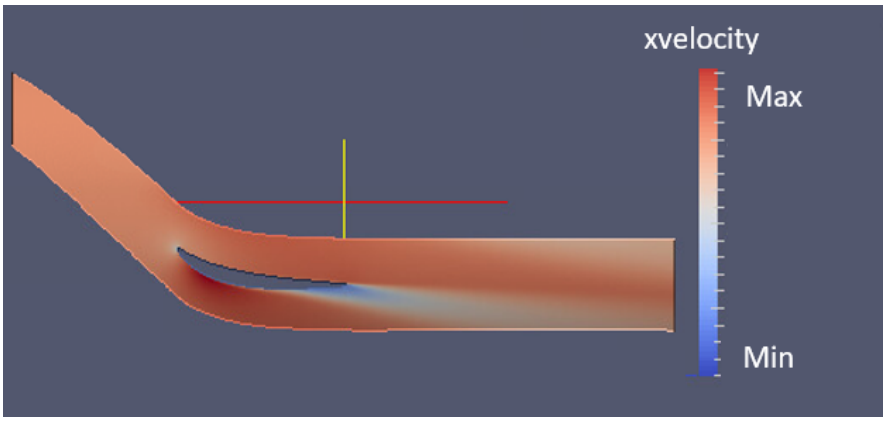


Figure 3.13: *Case +4. Isoareas of the mean x-velocity field. mid-span section on xy axis.*

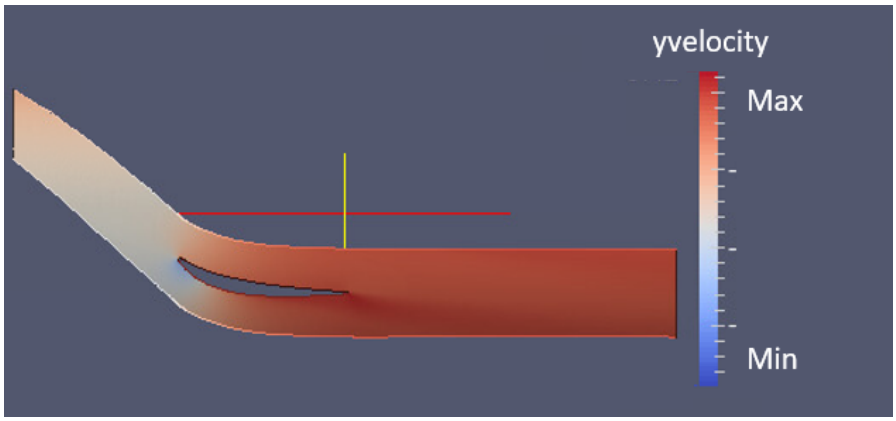


Figure 3.14: *Case +4. Isoareas of the mean y-velocity field. 2D slice on xy axis.*

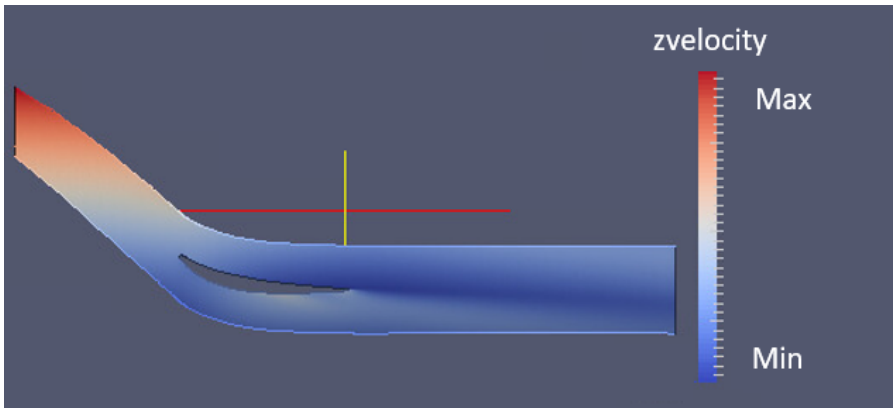


Figure 3.15: *Case +4. Isoareas of the mean z-velocity field. 2D slice on xy axis.*

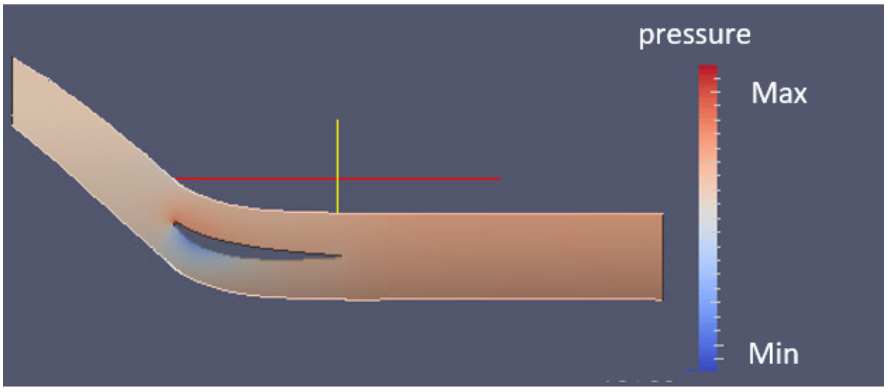


Figure 3.16: *Case +4. Isoareas of the mean pressure field. 2D slice on xy axis.*

The source term at the end of the simulation is close to zero for all equations. The highest divergence of the term is found in the x-velocity, close to the trailing edge of the vane where the highest unsteadiness of the flow exists. However the value is still very low in the order of 10^{-10}. The source term of the x-momentum equation is shown in figure 3.17.

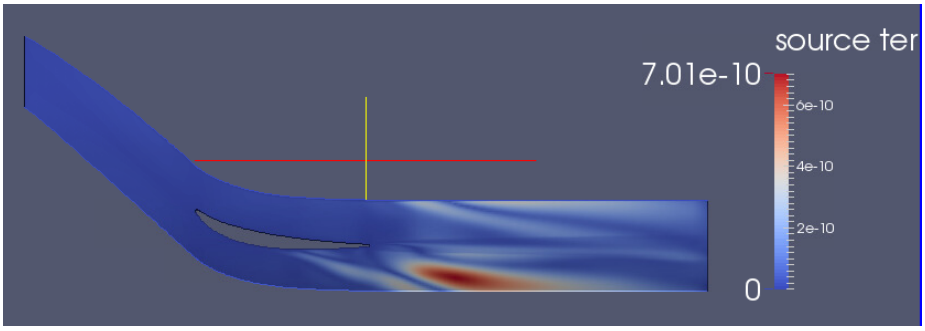


Figure 3.17: *Case +4. Isoareas of the source term by the end of the simulation.*

Using this solution of the primal problem, the adjoint system is solved and the adjoint solution is used to compute the sensitivities of the objective function 3.38. The sensitivities are computed using the equation 3.33. The source term needs to be added to the last term $\frac{\partial \vec{R}}{\partial \vec{X}}$ of the equation as

$$\frac{\partial S_i}{\partial X_i} = \frac{u_i - \bar{u}}{\tau} \frac{\partial V_i}{\partial X_i} \tag{3.39}$$

The sensitivity map is shown in figure 3.18

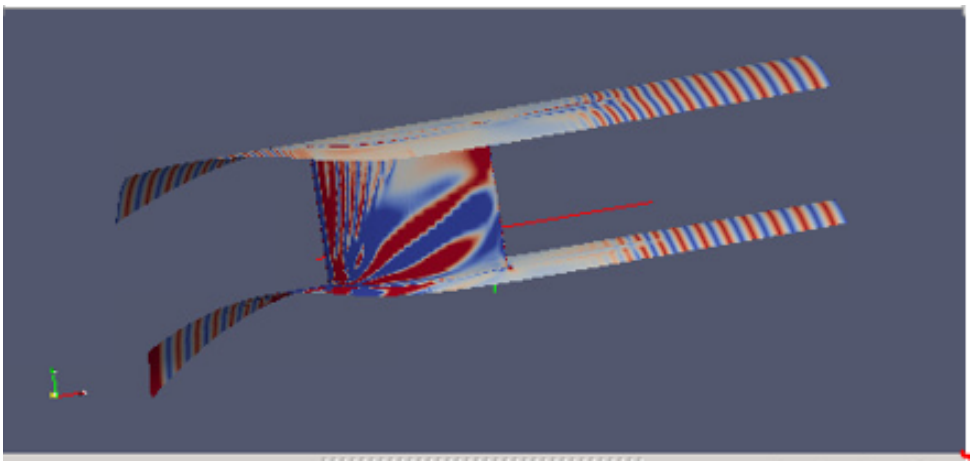


Figure 3.18: *Case+4 sensitivity derivatives.*

The objective function is the total pressure loss coefficient and the map shows which areas need to be reshaped in order to reduce the pressure losses. Red colour indicates areas that need to be pushed inwards, while blue areas need to be pulled outwards.

Study case: $+5^o$ off-design

Primal problem

As already mentioned in the beginning of this section, the second case to be studied is the $+5^o$ off-design case of the same compressor stator. The results are compared with the ones for the $+4^o$ case in order to assess the brute force method. In this case, the primal flow solution exhibits limit cycle oscillations and although the solution doesn't change significantly when compared to the $+4^o$ case, the adjoint solver is unable to converge. The RMS residual of the primal flow equations and the adjoint equations, without the source term, are shown in figures 3.19 and 3.20.

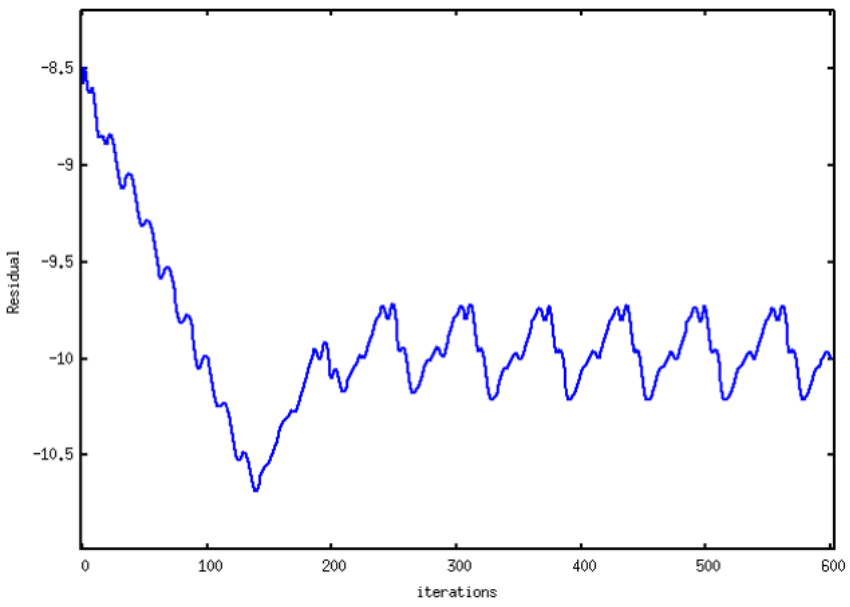


Figure 3.19: *Case +5. RMS residual of the primal equations without source term.*

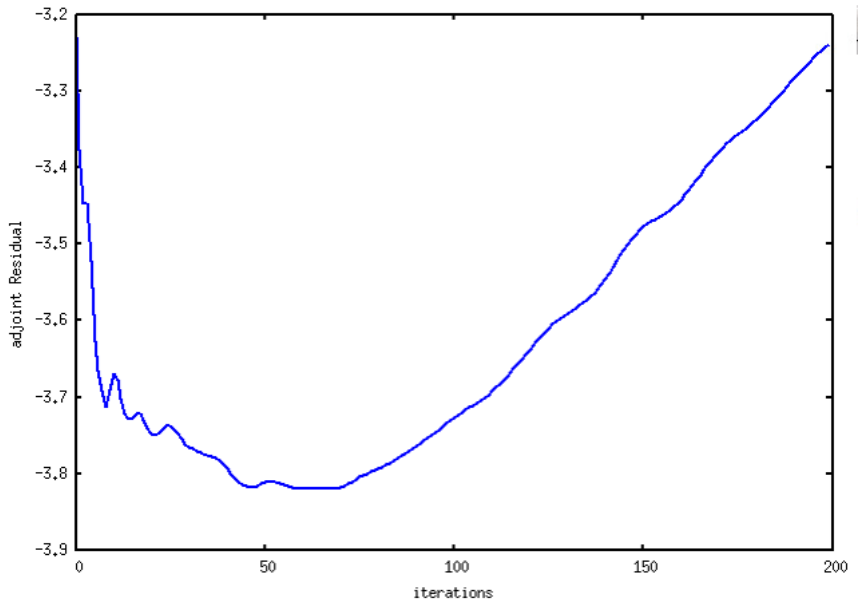


Figure 3.20: *Case +5. RMS residual of the adjoint equations without source term.*

In order to evaluate the brute force method, the primal problem is simulated with the source term switched on. The RMS residual of the primal equations is shown in figure 3.21.

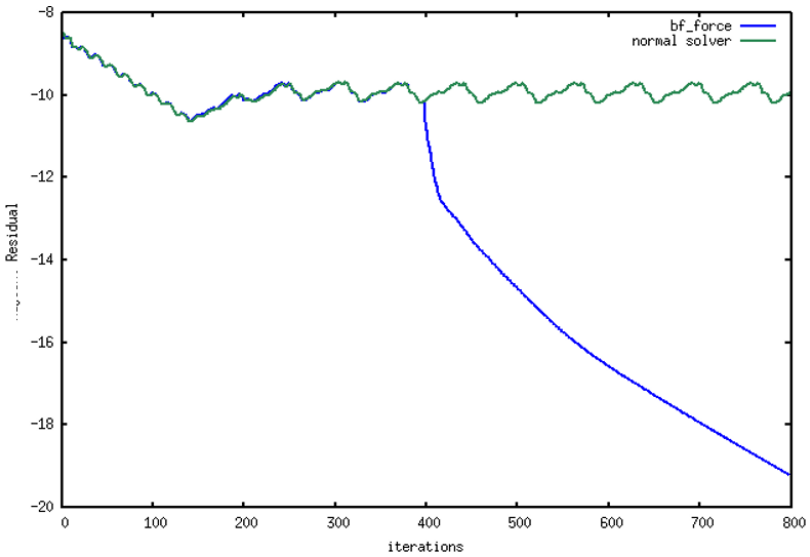


Figure 3.21: *Case +5. Comparison of primal equations' convergence of the solver with and without source term. Green - without source term. Blue - with source term.*

The green line shows the residual without using the source term and the blue one is after the introduction of the term. At 320 iterations the mean value of the flow starts to be computed. After 400 iterations, when the solution is already

oscillating for about 3 periods, the source term is switched on. The solution now starts to converge without any oscillations and by the end of the simulation, the RMS residual of the primal equations has dropped 9 orders of magnitude and reached a value of about 10^{-19} compared to the initial equations without the source term, where the residual dropped only one order of magnitude down to 10^{-10}. The mean value and subsequently the final solution of the flow variables is shown in figures 3.22-3.25.

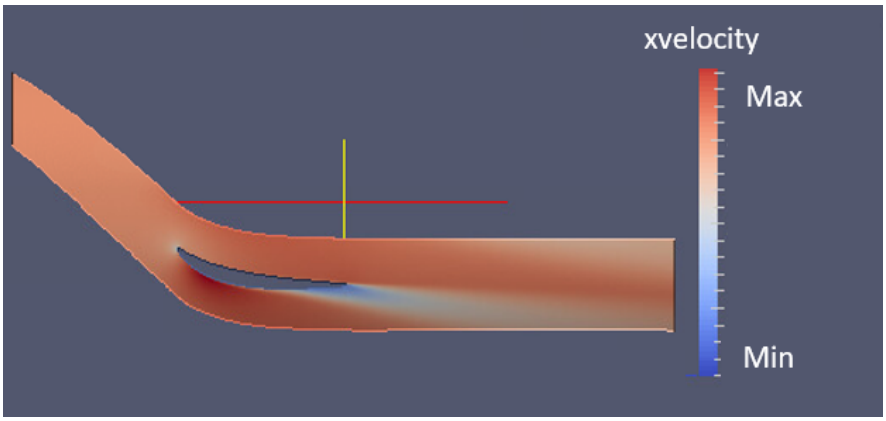


Figure 3.22: *Case +5. Isoareas of the mean x-velocity field. 2D slice on xy axis.*

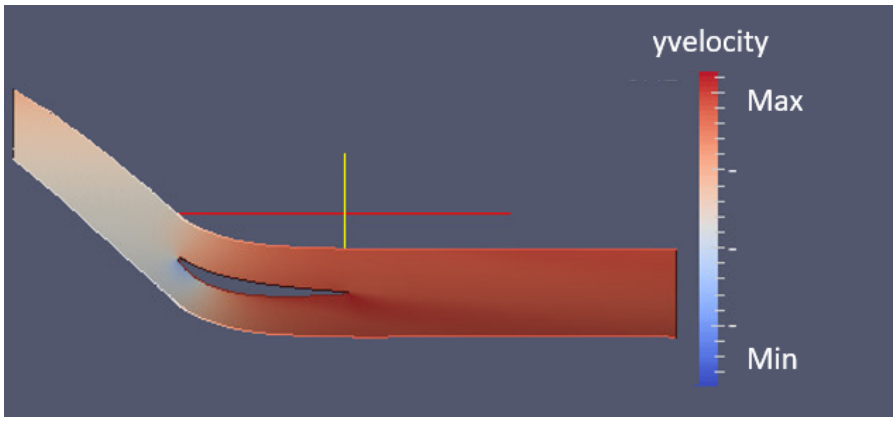


Figure 3.23: *Case +5. Isoareas of the mean y-velocity field. 2D slice on xy axis.*

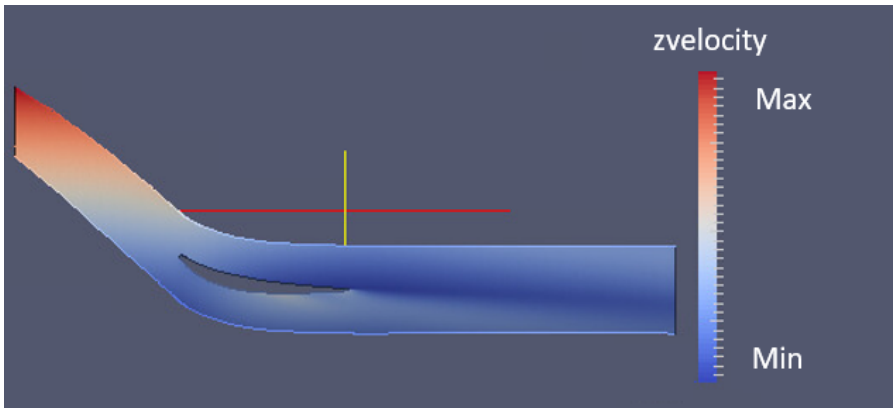


Figure 3.24: *Case +5. Isoareas of the mean z-velocity field. 2D slice on xy axis.*

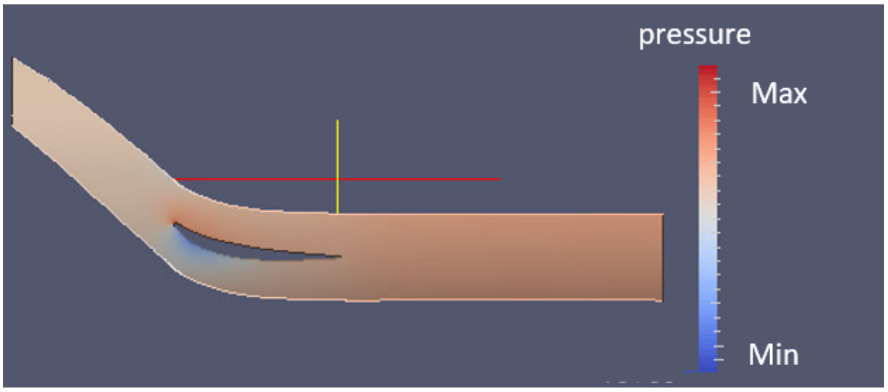


Figure 3.25: *Case +5. Isoareas of the mean pressure field. 2D slice on xy axis.*

Isoareas of the source term are shown in figure 3.26.

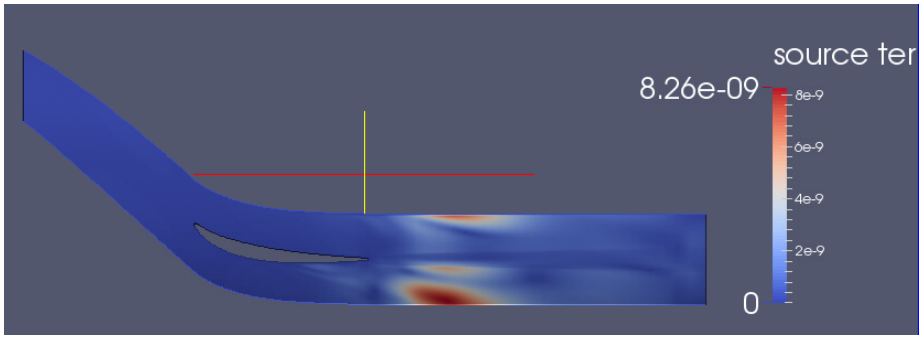


Figure 3.26: *Case +5. Isoareas of the source term.*

Again here, the source term is higher on the suction side of the blade, towards the trailing edge, where the highest unsteadiness of the flow due to separation can be found.

Adjoint Problem and Sensitivity Calculation

After the computation of the primal fields, the adjoint system is solved using the fully converged primal solution. The RMS residual of the adjoint equations with the source term is shown in figure 3.27.

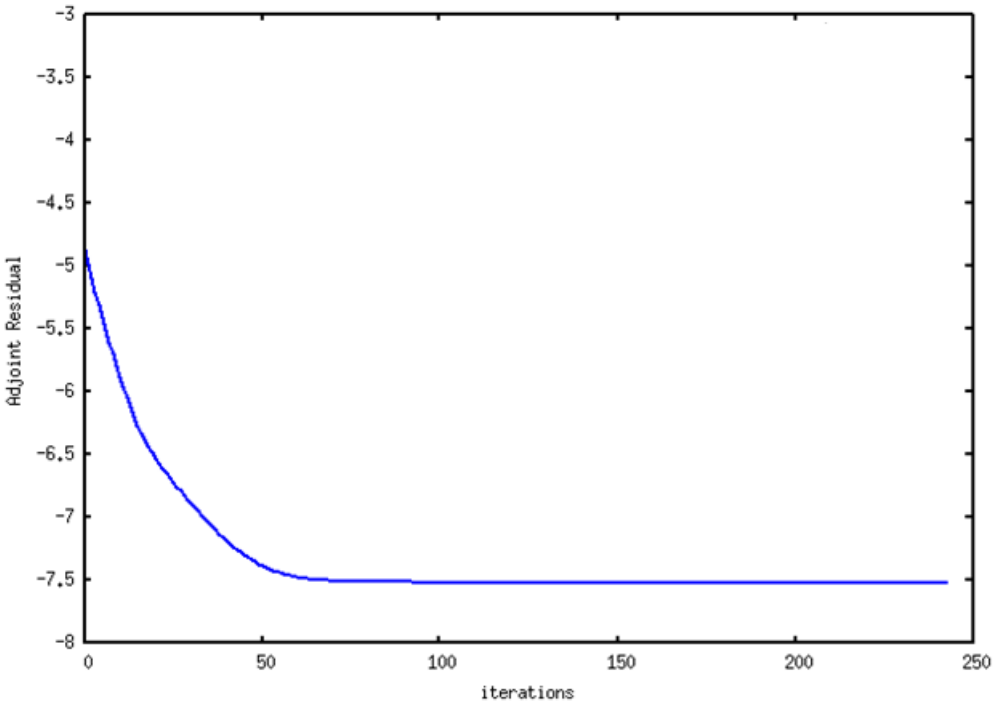


Figure 3.27: *Case $+5^o$. RMS residual of the adjoint equations with the source term.*

One can see that the equations are now converging and after around 70 iterations the RMS residual stabilizes. Comparing the residual with the one in figure 3.20 it appears that the introduction of the source term has contributed to the convergence of the adjoint equations even for 2.5 orders of magnitude which proves how sensitive is the adjoint system to the nature of the primal solution given.

Going forward, the adjoint solution is used to compute the sensitivities of the objective function 3.38. The sensitivities are calculated the same way as for the $+4^o$ case. The isoareas of the sensitivities is shown in figure 3.28.

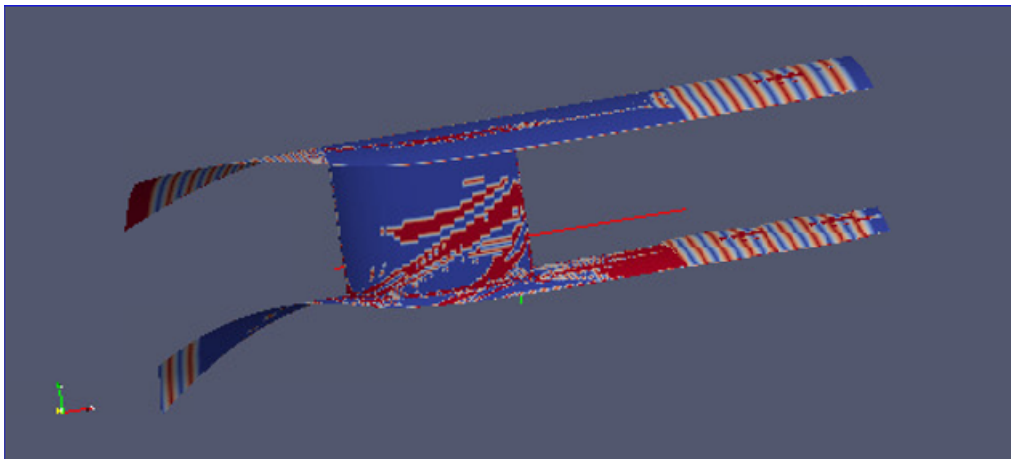


Figure 3.28: *Case+5 sensitivity derivatives.*

As observed, the values of the sensitivities are quite low, very close to zero throughout the whole domain. One reason why the sensitivities are low is that, despite the adjoint equations converging, the residual has not achieved deep convergence. Looking at the stabilized residual one can see that it dropped 2.5 orders of magnitude down to 10^{-7} which is still higher than the primal residual which dropped 9 orders of magnitude down to 10^{-14}.

However the most important factor in such a sensitivity map is the sign of the sensitivities rather than the absolute number. To examine that, a comparison with the sensitivity map of the $+4^o$, converged case has been done. The expectation is that the two cases would give similar pattern when it comes to the sensitivity map. Comparing the map with figure 3.18 one can see that the direction of the sensitivities on the stator is comparable and for the $+5^o$ case the sensitivities appear to have the expected sign.

In order to validate the method, the next step would be to use the sensitivity map and perform at least one optimization step. This would show if the objective function is actually reducing which would mean that the direction of the sensitivities is correct. However, this is currently out of the scope of this work. This work has set the ground for further analysis on the adjoint convergence by introducing and testing the brute force method in two cases.

Chapter 4

Programming and Parallelization of a Flow Solver and the Continuous Adjoint Method using the Ghost-Cell Method. Applications in Aerodynamic Shape Optimization.

The second part of the thesis was conducted at the Parallel CFD & Optimization Unit of the NTUA on an in-house ghost-cell flow solver of inviscid flows, to be extended to viscous flows. The purpose is to apply methods that improve the speed of the software and additionally, develop the continuous adjoint method in order for the software to be further used in aerodynamic shape optimization. It is a 2D steady CFD solver which uses cartesian grids and the ghost cell method to apply the boundary conditions on the geometry walls. A cartesian grid consists of only horizontal and vertical lines as shown in figure 4.1.

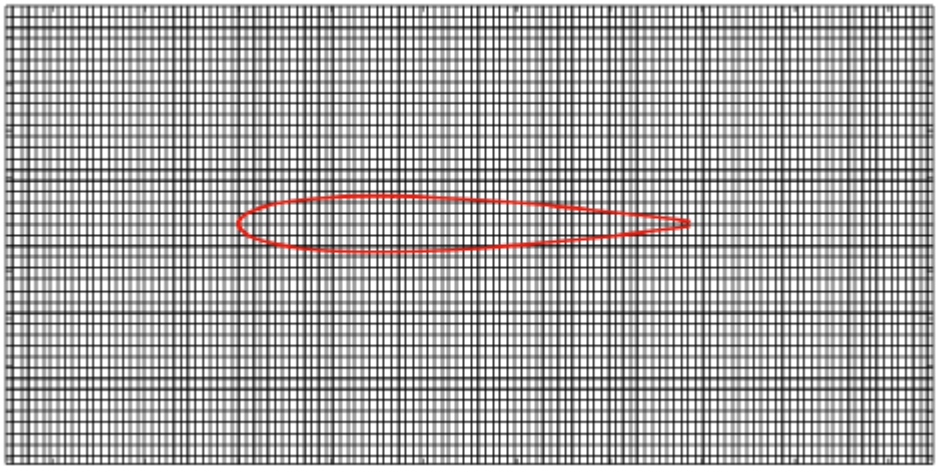


Figure 4.1: *Cartesian grid around an airfoil*

One can see that the form of the mesh is independent of the geometry it surrounds, as the boundary of the geometry intersects with the computational cells arbitrarily. The biggest benefit of cartesian grids can be seen when simulating flow around moving geometries. In such cases, if the grid is body-fitted, then at every time-iteration, while the geometry is moving, a new mesh must be created in order to follow the moving boundary. However if a cartesian, non-body-fitted grid is used, then it remains unchanged throughout the simulation. The main drawback comes form the fact that cartesian grids are not body-fitted, which means that the grid nodes are not attached to the solid boundary. In this case, imposing boundary conditions is not a straightforward process and has to be done carefully, in order not to violate the physical laws. Throughout the years, a number of techniques has been developed in order to impose boundary conditions on a non body-fitted mesh. All these methods are called "Immersed Boundary Methods" (IBM) [40], a term inspired from the fact that the geometry looks as if it is immersed into the mesh.

Immersed Boundary Methods - IBM

The Immersed Boundary Methods are generally divided in two categories, the continuous methods and the discrete ones [40]. Their main purpose is to simulate the effect of the solid boundary on the fluid flow, either by changing the governing equations directly, or by interfering with the discretization method. In particular, within the continuous methods, a new term $\vec{f}$ is introduced in the equations to simulate the force exerted from the solid to the fluid. The force becomes bigger as the flow approaches the boundary and fades out as the flow goes away from it. For a steady inviscid flow of a compressible fluid around a solid body, the Euler

equations of a ompressible fluid apply [13],

$$\begin{aligned} \nabla \cdot \rho\vec{u} &= 0 \\ \nabla \cdot \rho\vec{u}\vec{u} + \nabla p &= 0 \end{aligned} \tag{4.1}$$

The force term is added to the RHS of the momentum equations as

$$\nabla \cdot \rho\vec{u}\vec{u} + \nabla p = \vec{f} \tag{4.2}$$

The equations are then discretized and solved on the nodes or control volumes of the cartesian grid. The continuous methods are mainly used in biology where the geometrical boundaries are elastic e.g. the surface of the heart, while they are not so commonly used for airfoils which have solid boundaries. The first one to introduce the IBM methods was C. Peskin [34] while studying the blood flow inside the human heart. One drawback of the continuous methods is that they demand the equations to be solved within the entire solid geometry, which adds to the computational cost without providing any additional information, as the solution inside the solid geometry has no physical meaning.
In the discrete methods, the governing equations are first discretized and then modified in the neighbourhood of the solid boundary to account for its effect on the fluid. In contrast to the continuous methods, here the equations are only solved within the fluid domain [30]. The two main discrete methods are the Cut-Cell Method and the Ghost-Cell Method; in the first one, the parts of the cell which are inside the body are trimmed and the equations are solved only within the fluid portion, while in the second one, an additional equation is introduced and solved within the solid body to simulate the force on the fluid. This thesis deals with the Ghost-Cell Method which will further be analyzed in the next section and from now on will be referred to as GC. The existing software only handles inviscid flows, so initially, the viscous terms will be added to the equations. To continue, the software will be parallelised using the MPI protocol. Finally, the continuous adjoint method will be derived from the primal problem, in order to perform aerodynamic shape optimization. The optimisation software will be tested on two airfoils, which will be optimised for maximum lift.

4.1 2D Steady Viscous Flows

In this section, the governing equations used to simulate viscous flows are presented. As the software was initially developed to handle steady 2D inviscid flows [12] so the first task is to modify the equations to simulate viscous flows. At first, the process of developing a cartesian grid is described and afterwards the 2D Navier - Stokes equations are presented and discretized within the

computational domain. The application of the boundary conditions based on the Ghost-Cell Method is shown. In the next sections, the software is validated against the analytical solution of the flow over a flat plate and some additional experimental data on an airfoil.

Cartesian Mesh Generation

The mesh generation process presented here was already part of the software and was developed by Samouchos [44]. The generation of a cartesian grid starts by dividing the computational domain in four cells of equal volume, which connect on the barycenter of the initial domain. Each one of the cells is again divided in four cells of equal volume and the process continues until the maximum volume is less than the maximum accepted volume defined by the user [44]. At this stage, the mesh quality is not good, and so a process of gradual refinement is applied, in order to achieve higher solution accuracy close to the solid boundary. The refinement is done based on a sigmoid function 4.2, which defines the volume V_i of each cell as a function of its distance from the solid boundary.

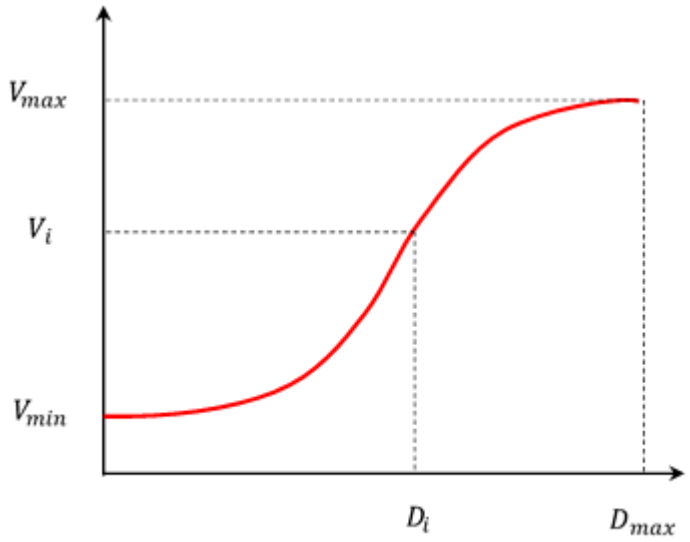


Figure 4.2: *Definition of the volume cell V as a function of the distance from the solid boundary.*

By the end of this process, the mesh is of good quality and can provide the highest accuracy near the boundary. An example of a cartesian refined mesh around an airfoil is shown in figure 4.3.

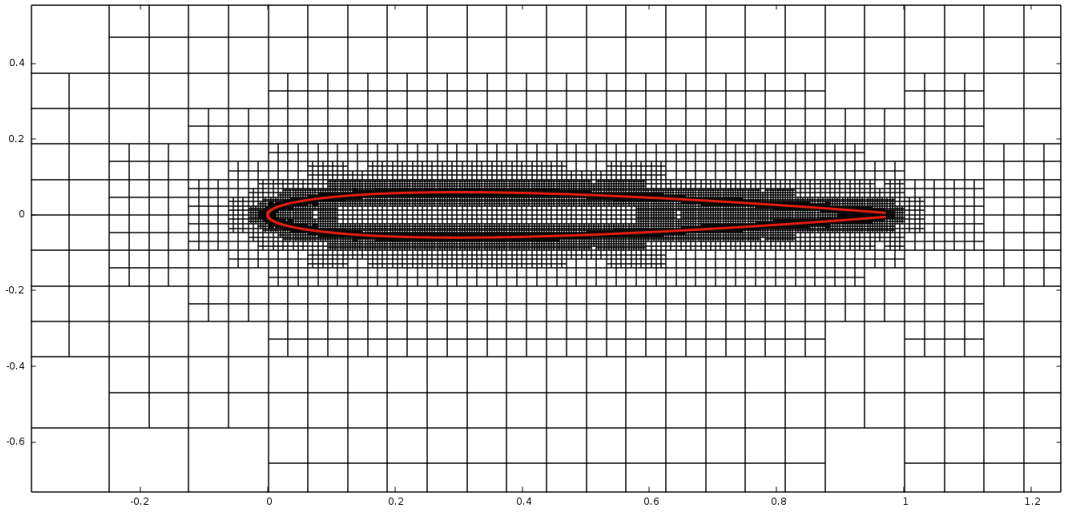


Figure 4.3: *Refined cartesian grid around an airfoil.*

2D Steady Navier-Stokes Equations and Discretization on Finite Volumes

As mentioned before, the software was only able to simulate inviscid flows, so as a first step, the viscous terms were added to the equations. Here, the final 2D Navier-Stokes equations are presented together with their discretization on the computational domain and afterwards the software is validated.
Based on the analysis made on chapter 2, equation 2.7 for 2D steady viscous flow of compressible fluids is applied here

$$\nabla \cdot \vec{F}^{inv}(\vec{U}) + \nabla \cdot \vec{F}^{visc}(\vec{U}) = 0 \tag{4.3}$$

where

$$\vec{F}^{inv} = \begin{bmatrix} \rho\vec{v} \\ \rho\vec{v}u + p\vec{e}_x \\ \rho\vec{v}v + p\vec{e}_y \\ \rho\vec{v}E + p\vec{v} \end{bmatrix}, \qquad \vec{F}^{visc} = -\begin{bmatrix} 0 \\ \vec{\tau}_1 \\ \vec{\tau}_2 \\ \vec{\tau}_k v_k + q_k \end{bmatrix}, \tag{4.4}$$

where $\vec{\tau}_k = [\tau_{k1}, \tau_{k2}, \tau_{k3}]^{\mathrm{T}}$ is computed based on eq. 2.6 and from Fourier's law of heat conduction $\vec{q} = k\nabla T$.
Equations 4.3 are discretized in finite volumes and the values of the flow variables are saved at the center of the volumes (cell-centered method). This means that

the finite volumes Ω are the computational cells of the mesh.

$$\int_{\Omega} \left(\frac{\partial F_{ik}^{inv}}{\partial x_k} - \frac{\partial F_{ik}^{visc}}{\partial x_k} \right) d\Omega = 0 \tag{4.5}$$

where $i = 1, 2, 3, 4$ equations and $k = 1, 2$ directions. Twice repeated indices imply summation according to Einstein's convention. Using the Green-Gauss theorem, the spatial integral becomes surface around the surface S of a cell

$$\int_{S} (F_{ik}^{inv} - F_{ik}^{visc}) n_k dS = 0 \tag{4.6}$$

where n_k is the vector vertical to each edge of the cell. Assuming that the flux field $\vec{f}$ is uniform on every edge, the term is discretized

$$\int_{S} (F_{ik}^{inv} - F_{ik}^{visc}) n_k dS = \sum_{j=1}^{n} (F_{ik}^{inv} - F_{ik}^{visc}) n_k^j \Delta s^j \tag{4.7}$$

where n is the number of edges attached to the fluid.

The vector $\vec{F}^{inv}$ is again computed based on Roe's scheme [42] as

$$\vec{F}^{PQ_i} = \frac{1}{2}(\vec{F}^P + \vec{F}^{Q_i}) - \frac{1}{2}|\tilde{A}|(\vec{U}^L - \vec{U}^R) \tag{4.8}$$

where $\vec{F}^P$ is the inviscid flux of the current cell and $\vec{F}^{Q_i}$ is the flux of the $i - th$ neighbour of the cell as shown in figure 4.4

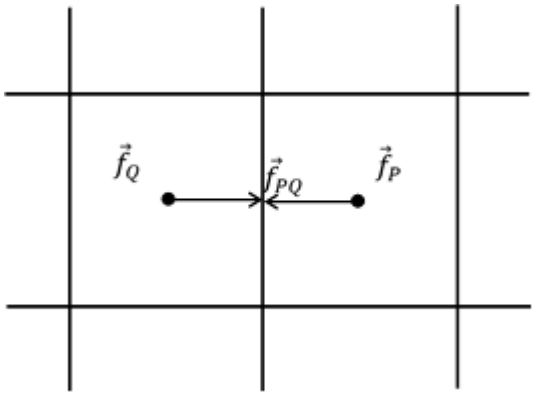


Figure 4.4: *Inviscid fluxes on the common edge between cells P and Q.*

Vectors U_L, U_R are the flow variables of Q and P as shown in figure 4.5

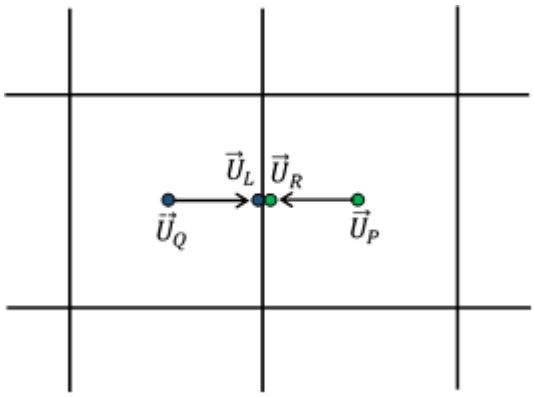


Figure 4.5: *Flow variables on the common edge between cells P and Q.*

The accuracy of the discretization scheme depends on how the flow variables are interpolated on the common edge. For first-order accuracy, the values are simply copied on the edge as

$$\begin{aligned} U_R &= U_P \\ U_L &= U_Q \end{aligned} \tag{4.9}$$

For second order accuracy, the Taylor expansion is used as

$$U_R = U_P + \left(\frac{\partial U}{\partial x}\right)_P \Delta x + \left(\frac{\partial U}{\partial y}\right)_P \Delta y \tag{4.10}$$

where $\Delta x = x_f - x_P$ and $\Delta y = y_f - y_P$ with f being the barcenter of the common edge. The vector U_L is computed accordingly using the U_Q values. The spatial derivatives $\frac{\partial U}{\partial x}, \frac{\partial U}{\partial y}$ of P are computed using the values U_Q of all the neighbours of P as described below. From Taylor expansion

$$\vec{U}_Q = \vec{U}_P + \frac{\partial \vec{U}_P}{\partial x}\Delta x + \frac{\partial \vec{U}_P}{\partial y}\Delta y \tag{4.11}$$

where $\Delta x = x_Q - x_P$ and $\Delta y = y_Q - y_P$. Setting $\Delta U = U_Q - U_P$ the equation becomes

$$\frac{\partial \vec{U}_P}{\partial x}\Delta x + \frac{\partial \vec{U}_P}{\partial y}\Delta y - \Delta \vec{U} = 0 \tag{4.12}$$

There are two cases that might appear; one cell sharing an edge with a cell of the same size, or sharing an edge with two smaller cells. On the first case, shown in figure 4.6, the derivatives are computed using central finite difference scheme as

$$\left(\frac{dU}{dx}\right)_P = \frac{U_Q - U_R}{2\Delta x} \tag{4.13}$$

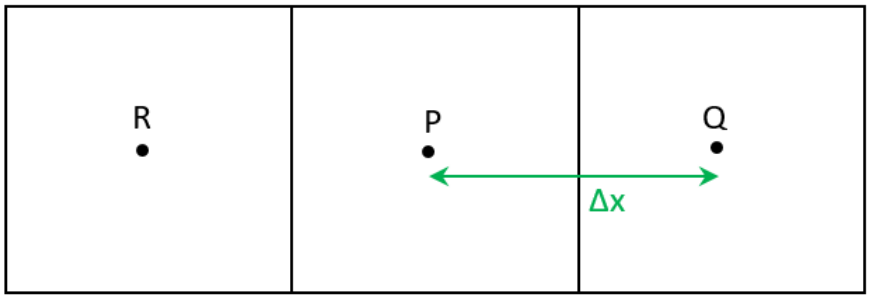


Figure 4.6: *Finite difference scheme for cells with the same volume.*

On the second case, where the cell shares an edge with two smaller cells as shown in figure 4.7, the mean value of the flow variables of the smaller cells is first computed and then the flow variables are transferred to the common edge as

$$\left(\frac{dU}{dx}\right)_P = \frac{U_{Qm} - U_R}{0.75\Delta x} \tag{4.14}$$

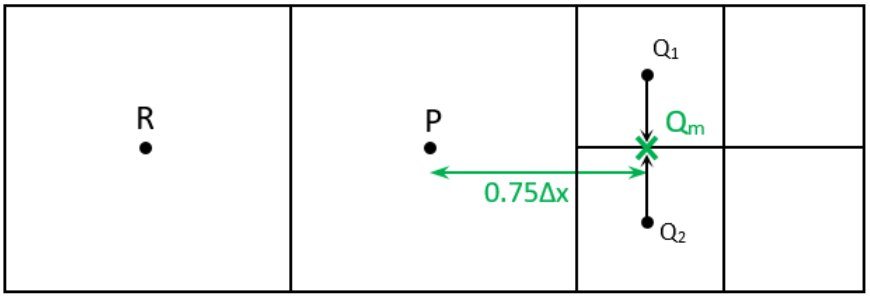


Figure 4.7: *Finite difference scheme for cells with different volume.*

The matrix $|\tilde{A}|$ from Roe's scheme eq. (4.8) is computed in the same way as the Jacobian matrix $A = (A_x, A_y) = (\frac{\partial \vec{F}_x^{inv}}{\partial \vec{U}}, \frac{\partial \vec{F}_y^{inv}}{\partial \vec{U}})$ [43] using the Roe averaged primitive variables which are [41]:

$$\vec{\tilde{Q}} = \begin{bmatrix} \tilde{\rho} \\ \tilde{u} \\ \tilde{v} \\ \tilde{h} \end{bmatrix} \tag{4.15}$$

where $\tilde{\rho} = \sqrt{(\rho_L \rho_R)}$ and $\tilde{Q}_{2,3,4} = \frac{\sqrt{\rho^L} W^L_{2,3,4} + \sqrt{\rho^R} W^R_{2,3,4}}{\sqrt{\rho^L} + \sqrt{\rho^R}}$.

The stress tensor of eq. 4.4 is calculated based on eq. 2.6 described in Section 2.1.

Flow Solver

Assuming $\vec{U}^*$ is the exact flow solution, which satisfies the equations

$$\vec{R}(\vec{U}^*) = 0 \tag{4.16}$$

where

$$\vec{R} = \frac{\partial F_{ik}^{inv}}{\partial x_k} - \frac{\partial F_{ik}^{visc}}{\partial x_k} \tag{4.17}$$

and also $\vec{U}$ is the current solution. In an iterative method, the solution $\vec{U}$ is getting updated after each iteration until it reaches $\vec{U}^*$. This means that $\vec{U}$ differs from $\vec{U}^*$ by $\Delta\vec{U}$.

$$\vec{U^*} = \vec{U} + \Delta\vec{U} \tag{4.18}$$

Equation (4.16) becomes

$$\vec{R}(\vec{U} + \Delta\vec{U}) = \vec{0} \tag{4.19}$$

And using Taylor expansion around $\vec{U}$ the equation becomes

$$\frac{\partial\vec{R}}{\partial\vec{U}}\Delta\vec{U} = -\vec{R}(\vec{U}) \tag{4.20}$$

Equation 4.20 is called delta formulation [7] because in each iteration, instead of computing the value of the flow variables $\vec{U}$, the solver computes the correction $\Delta\vec{U}$. Solving the system using delta formulation makes it in general more stable.

From equations 4.5-4.8, the term $\vec{R}_P$ can be derived as

$$\vec{R_P} = \sum_{i=1}^{nb}\left[\frac{1}{2}(\vec{F^P} + \vec{F^{Q_i}} - \frac{1}{2}|\tilde{A}|_{PQ}(\vec{U^L} - \vec{U^R})\right] n^i \Delta s^{Q_i} \tag{4.21}$$

The term $\frac{\partial R}{\partial U}$ consists of the diagonal entries

$$D_p = \frac{\partial R_P}{\partial U_P} \tag{4.22}$$

and the off-diagonal entries

$$OD_{PQ} = \frac{\partial R_P}{\partial U_Q} \tag{4.23}$$

from all its neighbours. From eq. 4.21

$$\frac{\partial R_P}{\partial U_P} = \frac{1}{2}\sum_{i=1}^{nb}(A + |\tilde{A}|)n^{Q_i}\Delta s^{Q_i} \tag{4.24}$$

$$\frac{\partial R_P}{\partial U_{Q_i}} = \frac{1}{2}(A - |\tilde{A}|)n^{Q_i}\Delta s^{Q_i} \tag{4.25}$$

For the rest of the $\frac{\partial R_P}{\partial U_W}$, where W are all the cells apart from the neighbours of P, the derivative is zero. Finally, for n cells, the system becomes

$$\begin{bmatrix} D_1 & \dots & OD_{1n} \\ \vdots & \ddots & \vdots \\ OD_{n1} & \dots & D_n \end{bmatrix} \begin{bmatrix} \Delta U_1 \\ \vdots \\ \Delta U_n \end{bmatrix} = - \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} \tag{4.26}$$

The system (4.26) is solved using the $Jacobi$ method. The correction of the flow variables is

$$\Delta U_P = -D^{-1}(R + \sum_{i=1}^{nb}(OD_{Q_i}\Delta U_{Q_i})) \tag{4.27}$$

and the solution after each iteration is updated based on (4.18).

4.2 Ghost-Cell Boundary Conditions

In order to solve the equations, the boundary conditions must be imposed. As already mentioned in the beginning of the chapter, the boundary conditions on the solid walls will be imposed using the Ghost-Cell Method. For viscous flow over the boundary of a static body, the boundary condition to be satisfied is that the velocity on the wall needs to be zero.

$$\vec{u} = 0 \tag{4.28}$$

The main idea of the method is to compute values for the flow velocity within an area of the solid body. The velocity within this area will then counteract the effect of the fluid velocity, so that finally the velocity right on the boundary will be zero. The cells within the solid body that get value for the velocity, are called Ghost-Cells [31]. For the rest of the solid cells, the governing equations are not

being solved so the flow variables don't need to be saved. An example of ghost-cells inside a cylinder are shown in figure 4.8

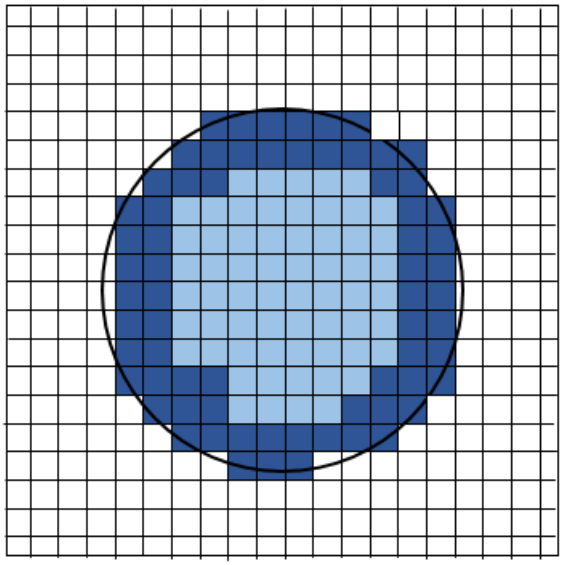


Figure 4.8: *Ghost-Cells within solid body. Dark blue: Ghost-Cells, Light blue:Solid cells*

The solid body is within the circle, while the fluid is outside. The ghost-cells are indicated with dark blue colour and this is where the flow velocity should be computed. In the simplest case, where the solid boundary is right on top of a mesh line, as shown in figure 4.9, then the velocity at the solid cells is set to be exactly the opposite of that of the fluid.

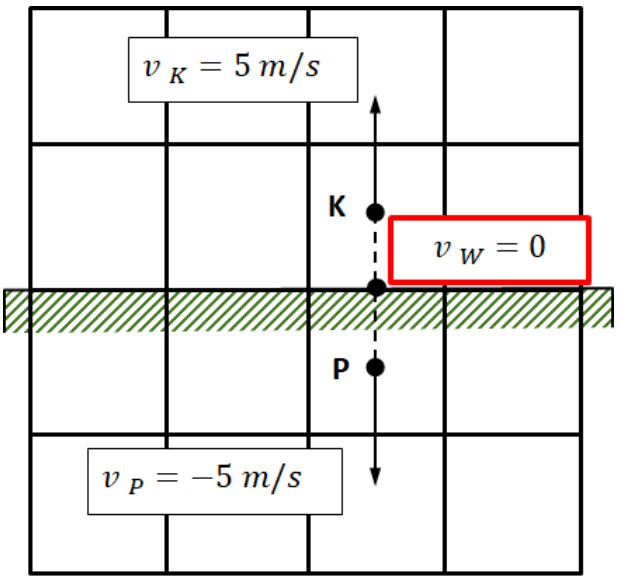


Figure 4.9: *Flow velocity on solid cells for the ghost-cell method.*

Initially, the velocity of the fluid cell K is copied to the solid cell P and then is getting reversed. By doing this, the velocity in the middle of the distance between K and P i.e. on the boundary, is zero. In the general case, the grid lines line up with the solid boundary as shown in figure 4.10

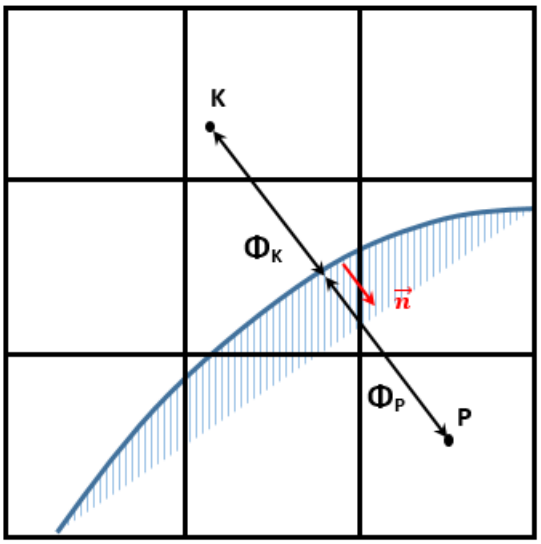


Figure 4.10: *Grid line that does not line up with the solid boundary.*

In order to copy the flow velocity inside the ghost cells, a new differential equation is solved within the solid cells as described below. The computational cost of solving one such equation is negligible compared to the solution of the governing equations. The equation demands that the velocity remains unchanged over the direction vertical to the solid boundary $\vec{n}$ as shown in figure 4.10. The direction of the vector $\vec{n}$ looks from the fluid towards the solid and its magnitude is

$$\vec{n} = (\frac{\partial \Phi}{\partial x}, \frac{\partial \Phi}{\partial y}) \tag{4.29}$$

where Φ is the distance of the cells from the boundary i.e. the level-set field, as shown in figure 4.10. The final equation which transfers the velocity inside the ghost cells is

$$\frac{\partial U_i}{\partial t} + \frac{\partial U_i}{\partial x_k}\frac{\partial \Phi}{\partial x_k} = 0 \tag{4.30}$$

By solving equation 4.30 the flow variables on the ghost cells are updated after each iteration from U_i^n to U_i^{n+1}.

$$U_i^{n+1} = U_i^n - \Delta t \frac{\partial U_i^n}{\partial x_k}\frac{\partial \Phi}{\partial x_k} \tag{4.31}$$

By solving the equation, the flow velocity is copied at the ghost cells and afterwards is getting reversed. It is proven that the correction on the velocity vector inside the ghost cells is right even when the equation 4.30 has not fully converged. It appears that solving equation 4.30 for 15 iterations provides accurate results. The pseudo-time step is

$$\Delta t = min(\frac{\Delta x}{n_x}, \frac{\Delta y}{n_y}) \tag{4.32}$$

After the flow variables are copied to the ghost cells, the velocity vector is reversed as

$$\vec{U_S} = -\vec{U_F} \tag{4.33}$$

where F indicates fluid cells and S indicates solid cells.

4.2.1 Interpolation of flow variables on the solid boundary

As the flow variables are not available directly on the solid boundary, it is sometimes needed to interpolate them there, in order to compute other variables such as lift coefficient or skin-friction coefficient. In this section, the method to interpolate variables from the cells surrounding the boundary, on top of the solid wall is presented.
A variable Φ is interpolated at a geometry node k as

$$\Phi_k = \sum_{i=0}^{nb} \Phi_i w_i \tag{4.34}$$

where nb are the grid nodes in the vicinity of k. Using the Dirac equation Phi is written as

$$\Phi_k = \int_\Omega \Phi \delta d\Omega \approx \sum_{i=0}^{nb} \Phi_i \delta \Delta\Omega \tag{4.35}$$

And comparing 4.34 and 4.35 the weights are $w_i = \delta\Delta\Omega$. For the numerical approximation of the Dirac function, the following equation is used [44]

$$\delta \approx \frac{1}{2\pi dr^2} e^{-\frac{1}{2}\frac{r^2}{dr^2}} \tag{4.36}$$

4.3 2D Steady Viscous Solver Validation

4.3.1 Flow over a flat plate

The first case used to validate the software is the flow over a flat plate where the results can be verified when compared with the analytical solution of the skin-friction coefficient, developed by Blasius [8]. Figure 4.11 shows the boundary layer over a flat plate when exposed to a flow with U_o velocity.

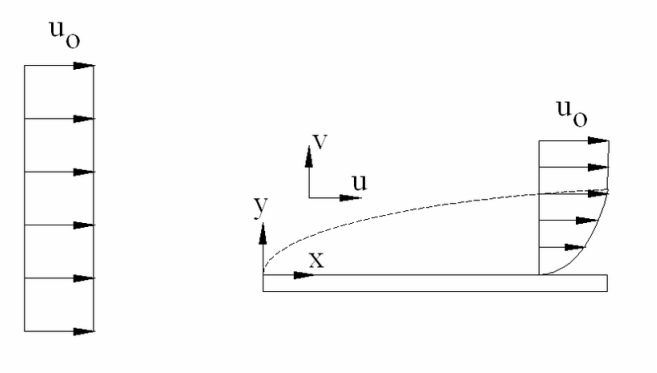


Figure 4.11: *Boundary layer over a flat plate. [25]*

Figure 4.12 shows the cartesian mesh created around the flat plate with negligible thickness. The mesh consists of 36000 cells.

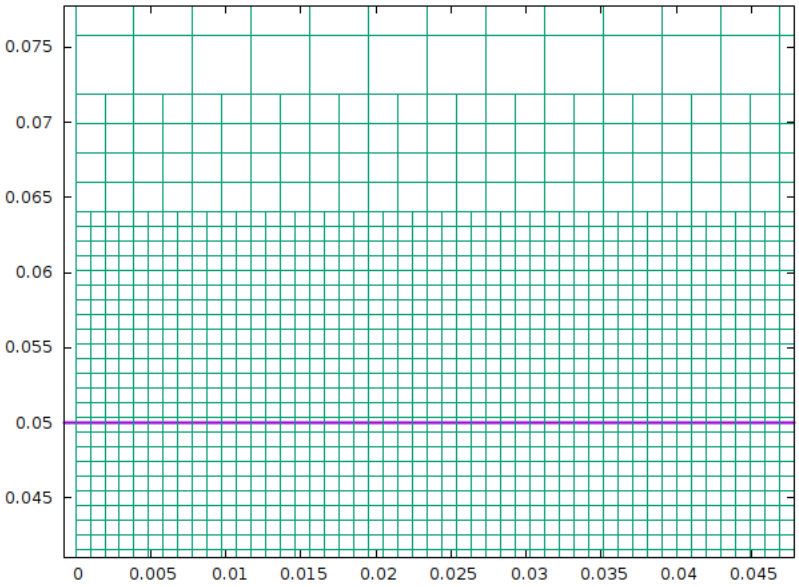


Figure 4.12: *Cartesian mesh around a flat plate.*

The plate is horizontal and the inlet flow angle is zero. The total pressure at the inlet of the domain is $p_{tin} = 1.03bar$ and the total temperature $T_{tin} = 300K$,while the static pressure on the outlet is $p_{sout} = 1.02bar$. The Reynolds number of the flow is $Re = 1093$, based on the length of the plate, and the Mach number is $Ma = 0.086$. On the upper side of the computational domain and far away from the boundary layer so it doesn't affect the flow, the Neumann boundary condition of the velocity is applied

$$\frac{\partial u}{\partial y} = 0 \tag{4.37}$$

The convergence of the governing equations and the velocity magnitude field is shown in figures 4.13 - 4.14

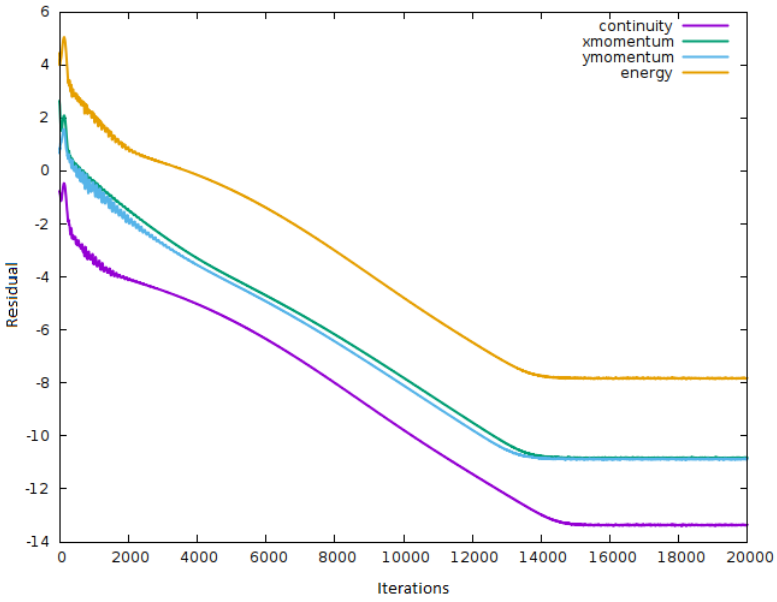


Figure 4.13: *Flow over a flat plate. Convergence of governing equations.*

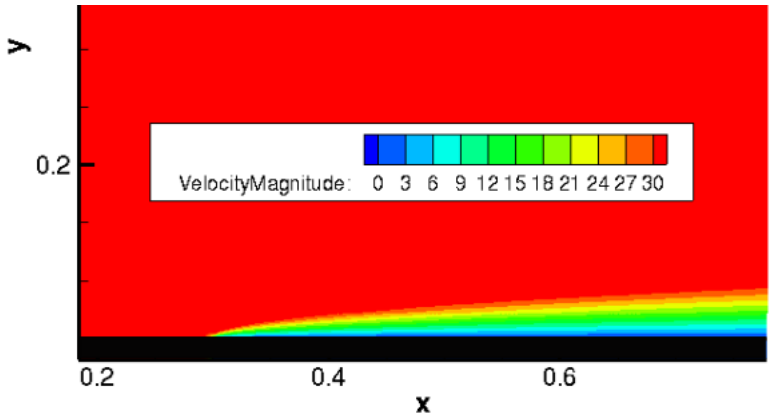


Figure 4.14: *Flow over a flat plate. Velocity Magnitude.*

To avoid the inlet boundary conditions interacting with the boundary conditions on the solid boundary, the plate is considered to start from the position $x = 0.3$ and not from the beginning of the domain. To validate the solution, the skin-friction coefficient of the plate is calculated as

$$C_f = \frac{\mu \left(\frac{du}{dy}\right)_w}{\frac{1}{2}\rho U_\infty^2} \tag{4.38}$$

As there are no values for the flow variables on the boundary of the plate, the term $\frac{du}{dy}$ needs to be interpolated to the geometry nodes.The interpolation of the flow variables at the solid boundary is described on Section 4.2.1. In this case, the flow can be considered incompressible ($Mach = 0.086$) so the Blasius analytical expression for the skin-friction coefficient can be used for comparison

$$C_f = \frac{0.664}{\sqrt{Re_x}} \tag{4.39}$$

Figure 4.15 shows the analytical and computed C_f distribution w.r.t. Re_x.

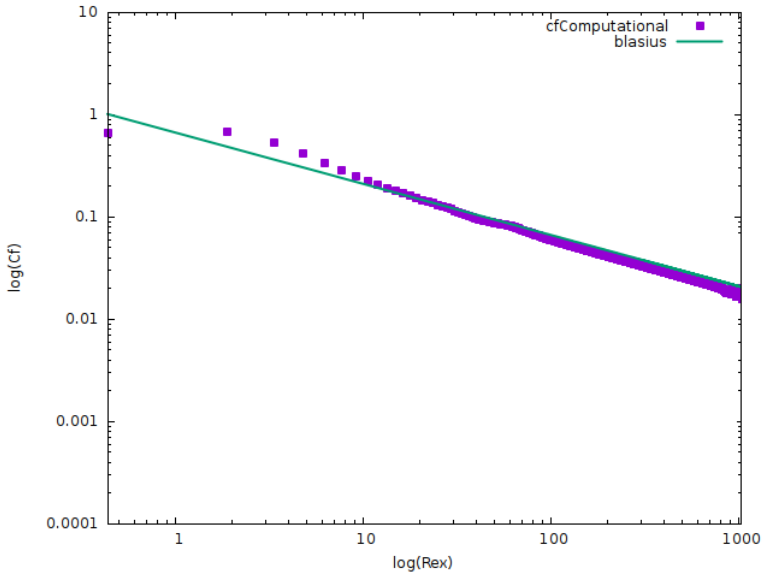


Figure 4.15: *Flow over a flat plate. Skin friction coefficient and comparison with the analytical solution.*

One can see that the CFD solution is aligned with the analytical one and the biggest difference between the two is less than 0.05.

4.3.2 Flow around NACA0012 airfoil

The second application to verify the software is the flow around a NACA0012 airfoil with a chord line of $c = 1m$. The infinite conditions of the flow have Reynolds number of $Re = 1000$, $Ma = 0.5$ and with zero angle of attack. The mesh around the airfoil consists of about 70000 cells, shown in figure 4.16

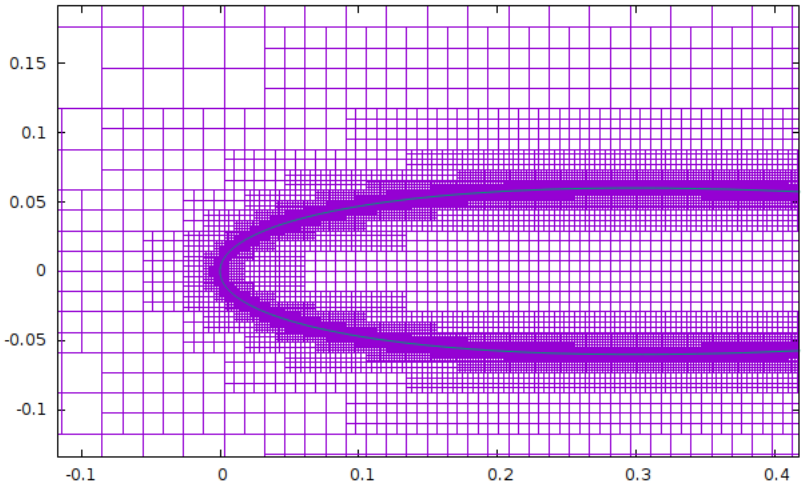


Figure 4.16: *Flow around $NACA0012$ airfoil. Part of the cartesian mesh.*

The convergence of the continuity, x-momentum, y-momentum and energy equations, as well as the velocity magnitude are shown in figures 4.17-4.19

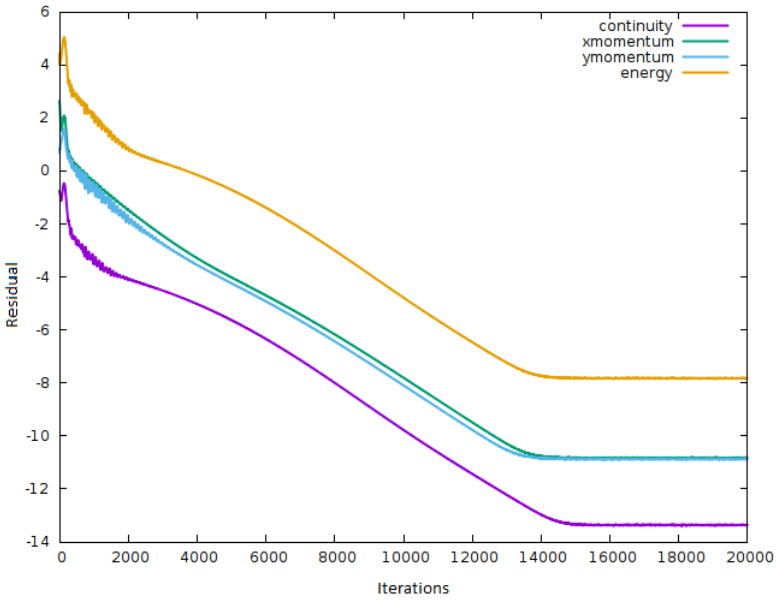


Figure 4.17: *Flow around NACA0012 airfoil. Convergence of the flow equations.*

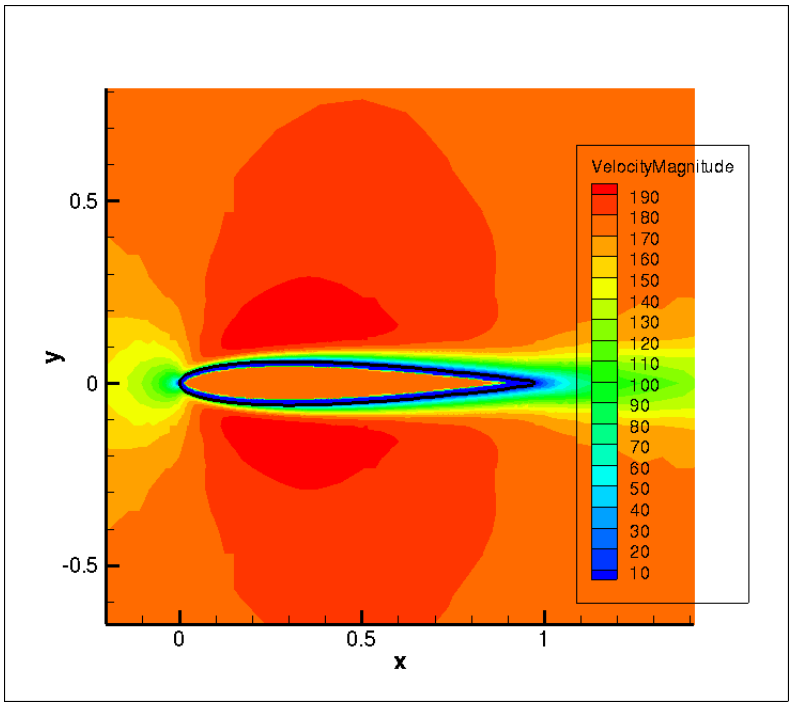


Figure 4.18: *Flow around NACA0012 airfoil. Velocity magnitude.*

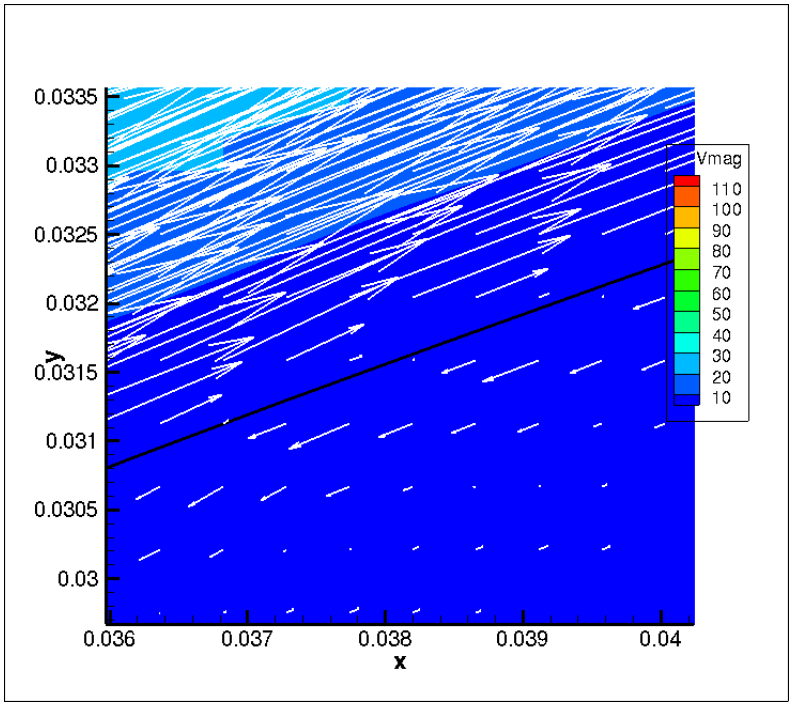


Figure 4.19: *Flow around NACA0012 airfoil. Velocity vector inside and outside of the fluid.*

Near the boundary and inside the solid geometry one can see the ghost cells having

a non-zero velocity value. To evaluate the solution, the pressure coefficient of the airfoil is calculated and compared with data from the literature [47]. Again the method described in subsection 4.3.1.1 is used to interpolate the flow pressure on the boundary. Figure 4.20 compares the two values for the pressure coefficient

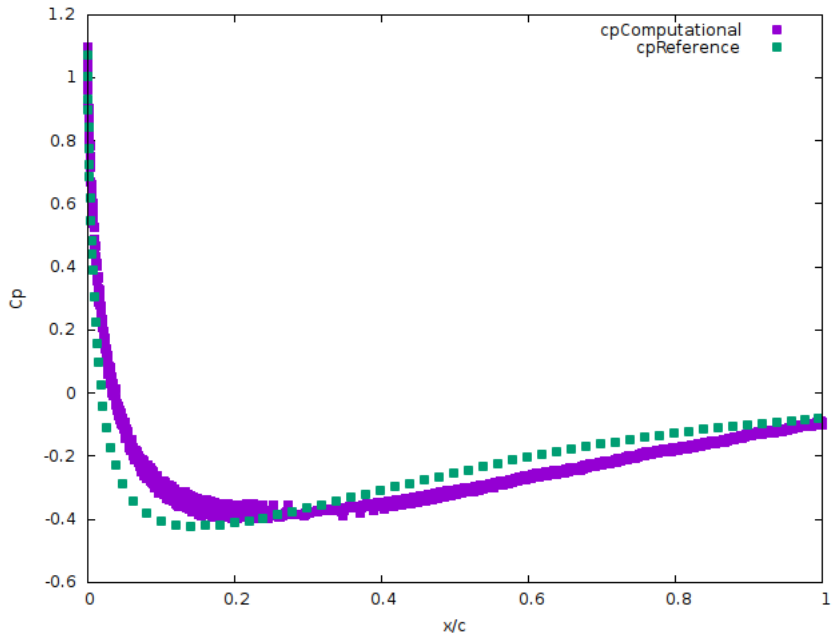


Figure 4.20: *Flow around NACA0012 airfoil. Calculated pressure coefficient (purple line) compared with literature data (blue line).*

The two curves appear to have a slight difference possibly because of the interpolation of the pressure on the boundary. The results are still valid though and prove that the solver is working correctly.

4.4 Parallelisation of CFD solver

In this section of the thesis, the process of making the software parallel is described. Simulating a flow and solving the governing equations can be a time consuming process. Especially if the computational domain is 3D and includes a complex geometry, such a simulation could need days or even weeks to be finished. For this reason, parallel software which runs simultaneously in various cores is becoming more and more popular within the CFD community.
For the purposes of this thesis, the distributed memory system (cluster) of the Parallel CFD & Optimization Unit of NTUA was used. Within a cluster there are different computational nodes connected to each other, and their function is controlled by a specific software. Every node has its own memory, which is shared by all its processors. This means that the system has distributed memory for each node, but shared memory for each local processor.

4.4.1 Communication between processors - MPI Protocol

The idea of parallel computing is that the processors are working and executing tasks independently. However, at times during the process, they need to communicate in order to exchange information. For this reason, for programming on shared and distributed memory systems, there are special software developed to control and coordinate the tasks on each processor. Herein, the MPI protocol is used.
MPI works based on the idea of Master & Workers. From the available processors, one is considered to be the master and is responsible for distributing tasks and information. Usually the Master doesn't execute any computations. The MPI protocol allows point-to-point and multipoint communications. Point-to-point communications are exclusively between two processors. This is achieved with two basic commands MPI_SEND and MPI_RECV. The message which is to be send has a unique identifier, known by both the sender and the receiver. Multipoint communication, on the other hand, is when a processor needs to send the same message to all the other processors. This is achieved by the command MPI_BCAST.

4.4.2 Mesh Partitioning in Parallel Computing

The generation of the computational mesh is executed only on a single processor, to be referred to as the Master. After the mesh is finished, the mesh nodes are distributed to the Workers, where the flow equations will be solved. Each Worker is responsible for only a part of the mesh, and the parts are interconnected. An example of how the mesh is divided in four processors is shown in figure 4.21.

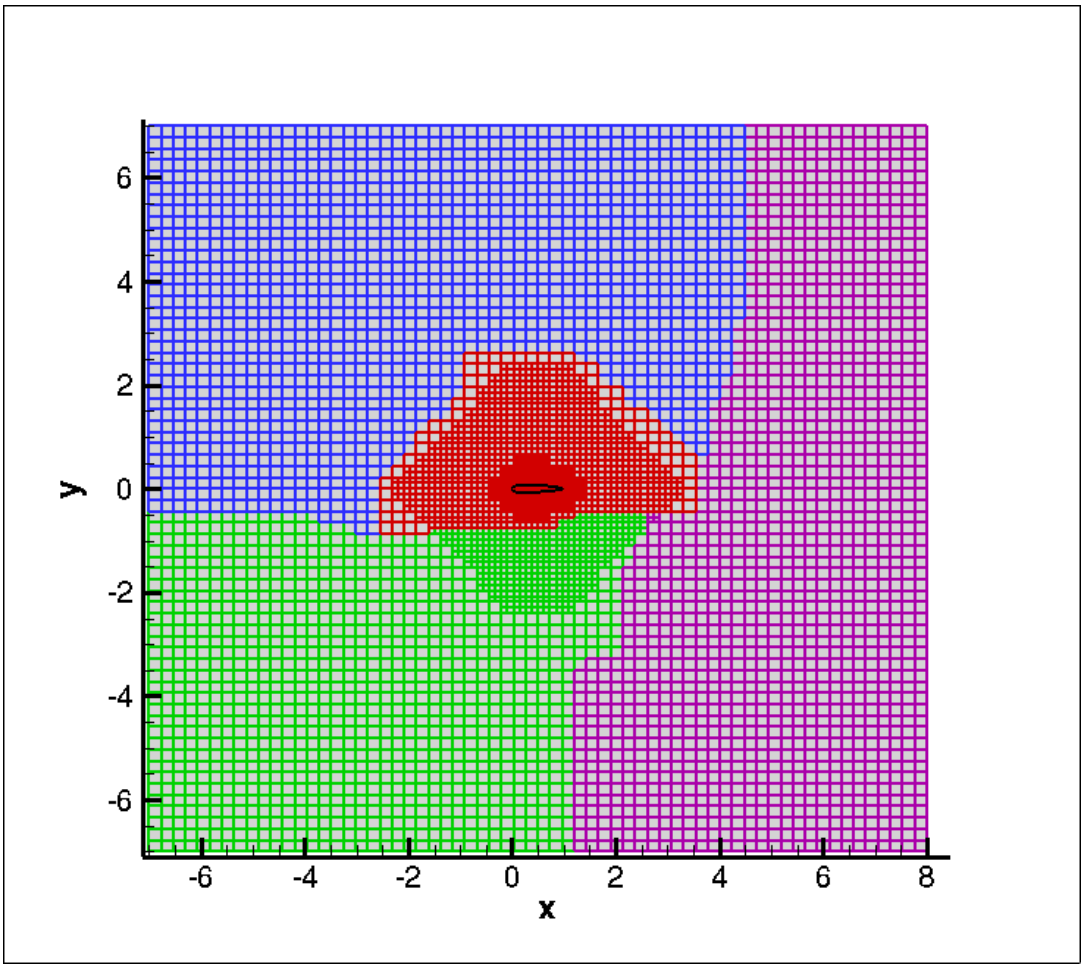


Figure 4.21: *Division of the mesh in four processors.*

The partitioning of the mesh in smaller parts is not an easy process. To make it as efficient as possible, each part of the mesh needs to contain the same amount of cells, which means that the processors are responsible for the same amount of tasks to execute. Second requirement is that the processors need to communicate the least possible times. Communication is time consuming and if they are too many, it could take more time to communicate than actually solving the equations. Herein, for the partitioning of the mesh the open software *METIS* [1] is used. At the end, each processor is assigned to one partition, where it will solve the flow equations.

4.4.3 Method and Frequency of Communication

As already described in chapter 4.1, in order to solve the equations at a cell, the values of the flow variables at all neighbouring cells are needed. However, as one processor only sees a part of the mesh, for the outer cells there is no information

about the flow variables of the neighbours, as they belong to a different partition. For this reason, communication zones are created as shown in figure 4.22

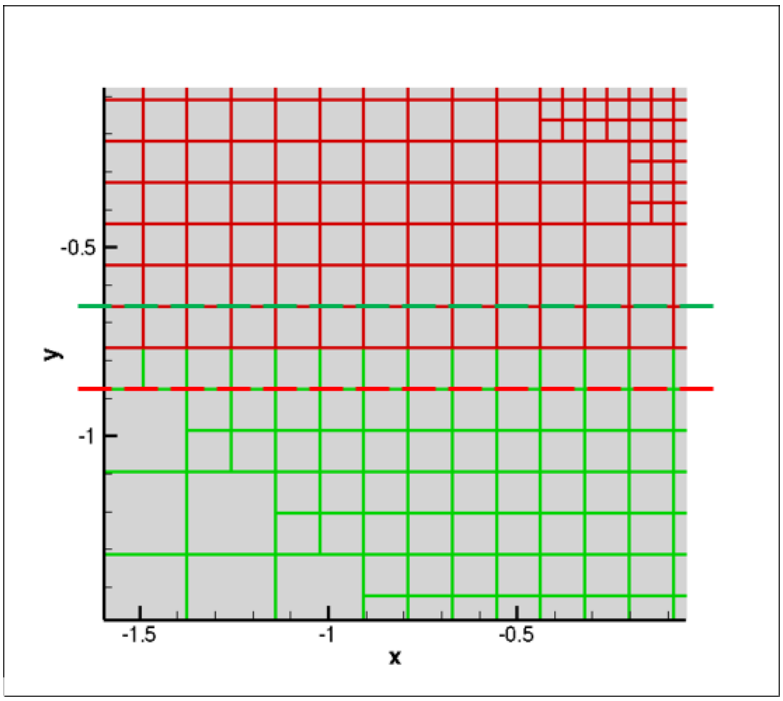


Figure 4.22: *Communication zones between processors.*

Each processor is solving the flow equations within one partition and gets information for the flow variables of the cells within the communication zone. After every solving iteration, each processor communicates with its neighbouring processor to exchange information about the cells in the communication zone. This kind of communication is done using the commands MPI_SEND and MPI_RECV.

4.4.4 Parallel Software Applications

The software is tested using different numbers of processors. For the assessment of the benefit of the parallelisation, two metrics were used; the parallel speed-up and the efficiency.
Parallel speed-up shows how much faster the software becomes with respect to the time of the serial execution and is given by

$$S = \frac{T_S}{T_P} \tag{4.40}$$

where T_S is the time of a serial execution and T_P the time of a parallel execution. The efficiency of the software shows how much does one processor contribute to the reduction of the overall execution time. The following equation is used for the calculation

$$E = S/p \tag{4.41}$$

where p is the number of processors used. Based on Amdahl's law [5], the parallel acceleration increases as the available processors increase, but it is always limited from the part of the software that cannot be parallelised. In the ideal case, where the whole software can be parallelised, the decrease in computational time would be proportional to the increase in the number of processors, i.e. for n processors the execution time would be n times smaller than the serial execution.
Here, for the assessment of the software, a simple case of a horizontal flow around an airfoil is used with three different sizes of the mesh. The first mesh consists of 30000 cells, the second one of 60000 cells and the last one of 100000 cells. An example of the coarser mesh is shown in figure 4.23

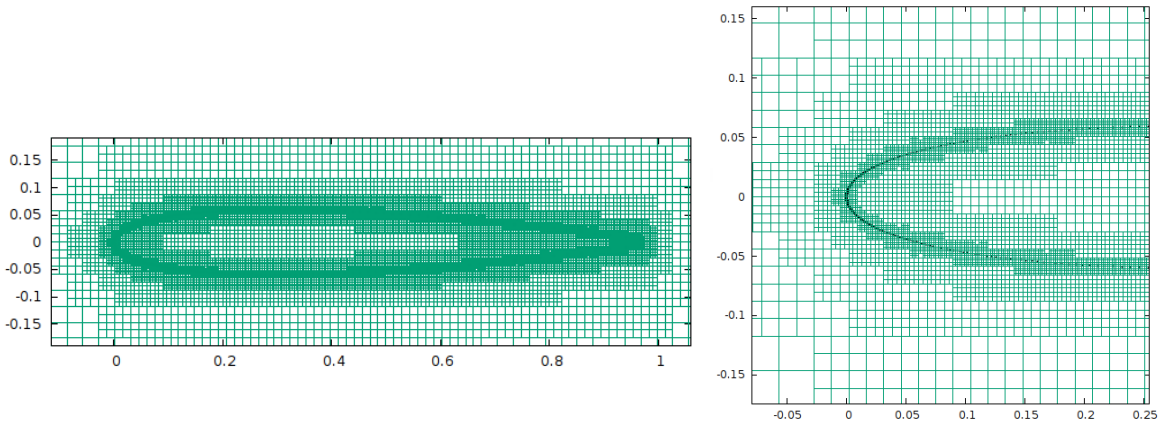


Figure 4.23: *Mesh of 30000 cells around an airfoil for the assessment of the parallel software.*

The flow is solved using an increasing number of processors and afterwards the speed-up and the efficiency are computed. Figure 4.24 shows the execution time w.r.t the number of processors.

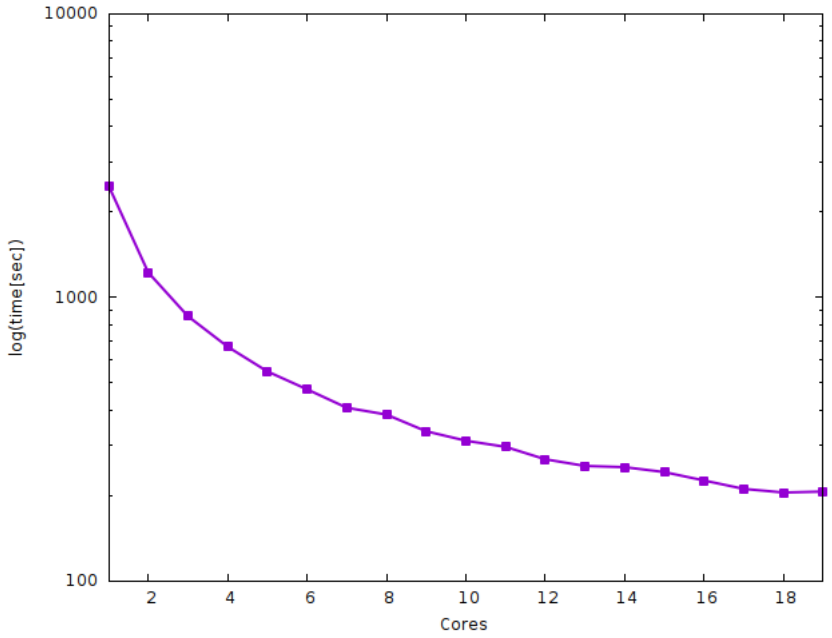


Figure 4.24: *Execution time w.r.t. number of processors.*

As the number increases, the benefit in execution time becomes smaller. This is because from one point on, the communications between the partitions become too many, and the addition of more processors adds to the communication burden. Figures 4.25 and 4.26 show the speed-up and the efficiency accordingly.

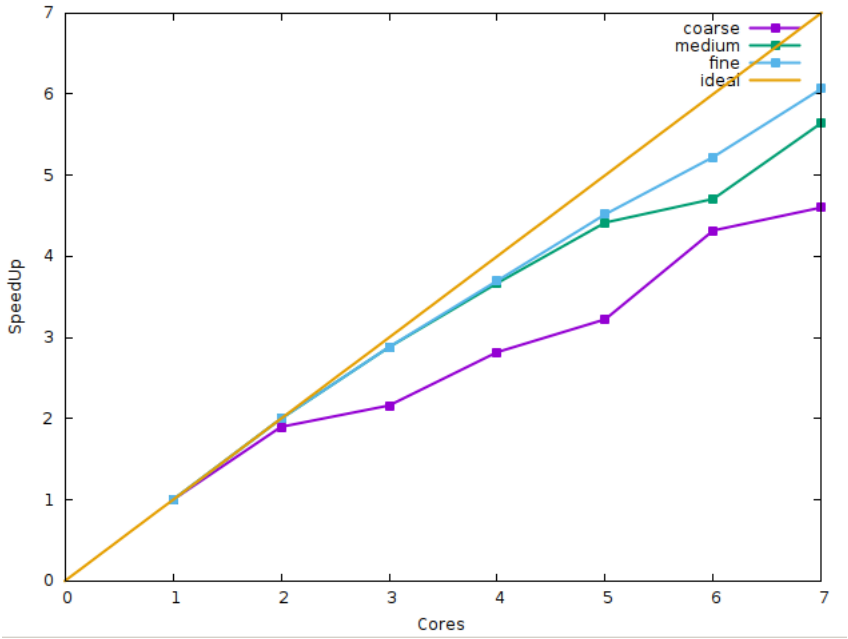


Figure 4.25: *Speed-up of the software for increasing number of processors.*

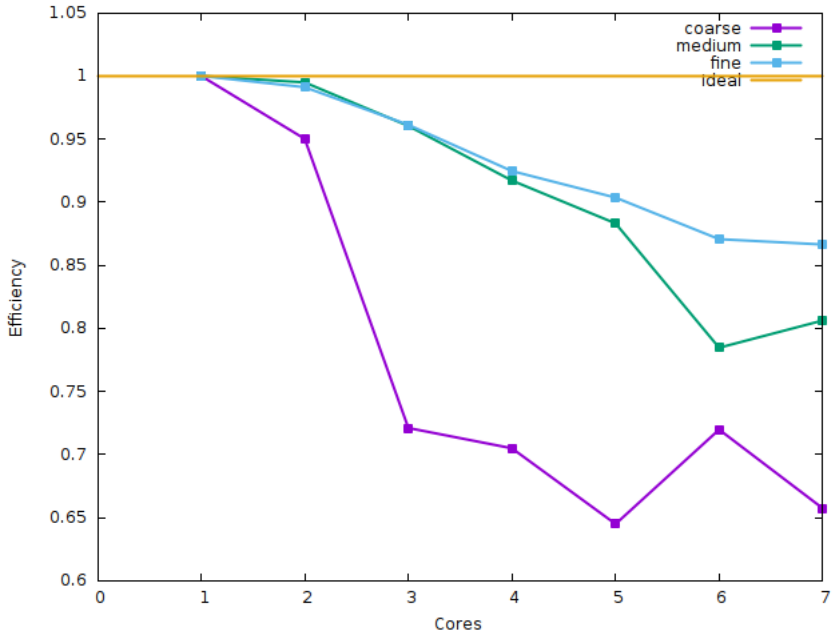


Figure 4.26: *Efficiency of the software from the addition of extra processors.*

One can see that with the addition of extra processors, the variables diverge from the ideal curve. However, by using a finer mesh, the parallel speed-up approaches the ideal curve. This is because in a finer mesh, the percentage of the cells in the communication zones is lower compared to a coarser mesh.

4.5 The Continuous Adjoint Method for Aerodynamic Shape Optimization

In this section, the continuous adjoint method is developed for the 2D steady inviscid flow. The primal equations are the Euler equations, i.e. equations 4.3 without the viscous terms. The adjoint method is implemented in the software in order to perform aerodynamic shape optimization. At first, the adjoint problem is derived from the primal equations and the ghost cell equation is also modified in order to apply the adjoint boundary conditions. Finally, the software is used to perform shape optimization of two airfoils with the objective to maximize the lift force exerted on them using different angles of attack.
The shape of the airfoils is derived using the Bezier curves and the coordinates of the Bezier points are the design variables of the optimization problem. The derivative of the objective function w.r.t the design variables is derived using the adjoint method, and afterwards, the steepest descent method is used to perform the optimization cycles.

4.5.1 Objective Function

The objective function for the optimization problems throughout this part of the thesis is the lift force over an airfoil,

$$F = \int_{S_w} p n_k r_k dS \tag{4.42}$$

where S_W the surface of the airfoil, p the static pressure, $\vec{r}$ the unit vector in the direction of the lift and $\vec{n}$ the unit vector vertical to the airfoil showing outwards. The design variables are updated using the steepest descent method

$$b_n^{new} = b_n^{old} - \eta \frac{\delta F}{\delta b_n} \tag{4.43}$$

where η is a coefficient which defines the step of the method.

4.5.2 Continuous Adjoint Method

To formulate the adjoint method, an augmented function is defined

$$F_{aug} = F + \int_{\Omega} \Psi_i R_i d\Omega \tag{4.44}$$

The derivative of this function w.r.t the design variables is the same as the derivative of F, as the additional term is zero.

$$\frac{\delta F_{aug}}{\delta b_n} = \frac{\delta F}{\delta b_n} + \frac{\delta}{\delta b_n}\int_{\Omega} \Psi_i R_i d\Omega \tag{4.45}$$

and using the $Leibnitz$ theorem the equation becomes

$$\frac{\delta F_{aug}}{\delta b_n} = \frac{\delta F}{\delta b_n} + \int_{\Omega} \frac{\partial\left(\Psi_i R_i\right)}{\partial b_n} d\Omega + \int_{S} \Psi_i R_i \frac{\delta x_k}{\delta b_n}\hat{n}_k dS \tag{4.46}$$

Each term of eq. (4.46) will be analysed separately

$$T1 = \frac{\delta F}{\delta b_n} = \underbrace{\int_{S_w} \frac{\delta p}{\delta b_n} n_k r_k dS}_{\text{ABC}} + \underbrace{\int_{S_w} p \frac{\delta}{\delta b_n}\left(n_k r_k dS\right)}_{\text{SD}}$$

The first term refers to the variation of the flow variables on the boundary of the airfoil and contributes to the adjoint boundary conditions. The second term includes the variation of the geometric characteristics of the solid w.r.t the design variables and contributes to the sensitivity derivatives. At this point is important to note again that the flow variables are not available on the boundary, so in order to calculate the surface integrals, interpolation of the values should be performed, as described in section 4.3.1.1. The variables to be interpolated are the conservative flow variables and their spatial derivatives. The derivatives of the geometric parameters is calculated analytically from the Bezier curve. The second integral of equation 4.46 becomes

$$T2 = \int_{\Omega} \frac{\partial\left(\Psi_i R_i\right)}{\partial b_n} d\Omega = \int_{\Omega} \frac{\partial \Psi_i}{\partial b_q} R_i d\Omega + \int_{\Omega} \Psi_i \frac{\partial R_i}{\partial b_q} d\Omega \quad = \int_{\Omega} \Psi_i \frac{\partial}{\partial b_q}\left(\frac{\partial f_{ik}}{\partial x_k}\right) d\Omega \tag{4.47}$$

The last equality is valid because the flow equations are satisfied $R_i = 0$. Going further

$$\begin{aligned} T2 &= \int_\Omega \frac{\partial}{\partial x_k}\left(\Psi_i \frac{\partial f_{ik}}{\partial b_n}\right) d\Omega - \int_\Omega \frac{\partial \Psi_i}{\partial x_k}\frac{\partial f_{ik}}{\partial b_n} d\Omega \\ &= \int_S \Psi_i \frac{\partial f_{ik}}{\partial b_n}\hat{n}_k dS - \int_\Omega \frac{\partial \Psi_i}{\partial x_k}\left(\frac{\partial f_{ik}}{\partial U_j}\frac{\partial U_j}{\partial b_n}\right) d\Omega \\ &= \underbrace{\int_S \Psi_i \frac{\partial f_{ik}}{\partial b_n}\hat{n}_k dS}_{\text{T3}} + \int_\Omega \underbrace{\left(-\frac{\partial \Psi_i}{\partial x_k}\mathbf{A}_{ijk}\right)}_{\text{FAE}} \frac{\partial U_j}{\partial b_n} d\Omega \end{aligned} \tag{4.48}$$

Computing the term $\frac{\partial U_j}{\partial b_n}$ is a very expensive process and needs to be avoided. To achieve that, its multiplier is set to zero, and this introduces the field adjoint equation (FAE). The T3 term is divided in the surface terms on the farfield boundaries of the domain and the ones on the solid walls.

$$\begin{aligned} T3 = \int_S \Psi_i \frac{\partial f_{ik}}{\partial b_n}\hat{n}_k dS &= \int_{S_\infty \cup S_w} \Psi_i \frac{\partial f_{ik}}{\partial b_n}\hat{n}_k dS \\ &= \underbrace{\int_{S_\infty} \Psi_i \frac{\partial f_{ik}}{\partial b_n}\hat{n}_k dS}_{\text{ABC}} + \underbrace{\int_{S_w} \Psi_i \frac{\partial f_{ik}}{\partial b_n}\hat{n}_k dS}_{\text{T4}} \end{aligned} \tag{4.49}$$

Term T4 is further expanded to include the geometric values on the boundary

$$\begin{aligned} T4 &= \int_{S_w} \Psi_i \frac{\delta f_{ik}}{\delta b_n}\hat{n}_k dS - \int_{S_w} \Psi_i \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\hat{n}_k dS \\ &= \int_{S_w} \Psi_i \frac{\delta}{\delta b_n}\left(f_{ik}\,\hat{n}_k\right) dS - \int_{S_w} \Psi_i\, f_{ik}\frac{\delta \hat{n}_k}{\delta b_n} dS - \int_{S_w} \Psi_i \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\hat{n}_k dS \\ &= \int_{S_w} \Psi_2 \frac{\delta}{\delta b_n}\left(p\hat{n}_x\right) dS + \int_{S_w} \Psi_3 \frac{\delta}{\delta b_n}\left(p\hat{n}_y\right) dS - \int_{S_w} \Psi_i f_{ik}\frac{\delta \hat{n}_k}{\delta b_n} dS - \int_{S_w} \Psi_i \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\hat{n}_k dS \\ &= \int_{S_w} \Psi_{k+1} \frac{\delta}{\delta b_n}\left(p\,\hat{n}_k\right) dS - \int_{S_w} \Psi_i\, f_{ik}\frac{\delta \hat{n}_k}{\delta b_n} dS - \int_{S_w} \Psi_i \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\hat{n}_k dS \\ &= \underbrace{\int_{S_w} \Psi_{k+1} \frac{\delta p}{\delta b_n}\hat{n}_k dS}_{\text{ABC}} + \underbrace{\int_{S_w} \Psi_{k+1}\, p \frac{\delta \hat{n}_k}{\delta b_n} dS}_{\text{SD}} \underbrace{- \int_{S_w} \Psi_i\, f_{ik}\frac{\delta \hat{n}_k}{\delta b_n} dS}_{\text{SD}} \underbrace{- \int_{S_w} \Psi_i \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\hat{n}_k dS}_{\text{SD}} \end{aligned} \tag{4.50}$$

After the analysis, the terms remaining, belong in three categories, as discussed in the next sections.

4.5.2.1 Field adjoint equations (FAE)

The equations that give the adjoint field are

$$-\frac{\partial \Psi_i}{\partial x_k}\ \mathbf{A}_{ijk} = 0 \Leftrightarrow -\mathbf{A}_{jik}\ \frac{\partial \Psi_j}{\partial x_k} = 0$$

$$\Leftrightarrow -\mathbf{A}_k^T\ \frac{\partial \vec{\Psi}}{\partial x_k} = 0 \tag{4.51}$$

and it's worth mentioning that are linear equations. The adjoint inviscid flow vector $\vec{f_A}^{PQ_i}$ is calculated as

$$\vec{f_A}^{PQ_i} = \frac{1}{2}(-A^{TP}\vec{\Psi}^P - A^{TQ}_i\vec{\Psi}^{Q_i}) - \frac{1}{2}|\tilde{A}^T|(\vec{\Psi}^L - \vec{\Psi}^R) \tag{4.52}$$

4.5.3 Sensitivity derivatives (SD)

The sensitivity derivatives are computed as

$$\begin{aligned}\frac{\delta F_{aug}}{\delta b_n} &= \int_S \Psi_i R_i \frac{\delta x_k}{\delta b_n}\hat{n}_k dS - \int_{S_w} \Psi_i\ f_{ik}\frac{\delta \hat{n}_k}{\delta b_n}dS + \int_{S_w} \Psi_{k+1}\ p\ \frac{\delta \hat{n}_k}{\delta b_n}dS \\ &+ \int_{S_w} p\ \frac{\delta}{\delta b_n}\left(\hat{n}_k \hat{r}_k dS\right) - \int_{S_w} \Psi_i\ \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\ \hat{n}_k dS\end{aligned} \tag{4.53}$$

The term $\left(\frac{\delta x_k}{\delta b_n}\right)$ describes the relation between the coordinates of the geometrical nodes and the design variables, and for this reason, for the cells that don't intersect with the geometry, the term is zero. So the first term of equation (4.53), based on the aforementioned observation, can be re-written only for the solid wall boundaries (S_w) as

$$\begin{aligned}\frac{\delta F_{aug}}{\delta b_n} &= \int_{S_w} \Psi_i R_i \frac{\delta x_k}{\delta b_n}\hat{n}_k dS + \int_{S_w} \left(\Psi_{k+1}\ p\ - \Psi_i\ f_{ik}\right)\ \frac{\delta \hat{n}_k}{\delta b_n}dS \\ &+ \int_{S_w} p\ \frac{\delta}{\delta b_n}\left(\hat{n}_k \hat{r}_k dS\right) - \int_{S_w} \Psi_i\ \frac{\partial f_{ik}}{\partial x_m}\frac{\delta x_m}{\delta b_n}\ \hat{n}_k dS\end{aligned} \tag{4.54}$$

4.5.3.1 Adjoint Boundary Conditions

The adjoint boundary conditions are formed from the terms on the boundaries of the computational domain i.e the terms noted with ABC, namely

$$\begin{aligned} ABC &= \int_{S_\infty} \Psi_i \, \frac{\partial f_{ik}}{\partial b_n} \, \hat{n}_k dS + \int_{S_w} \Psi_{k+1} \, \frac{\delta p}{\delta b_n} \hat{n}_k dS + \int_{S_w} \frac{\delta p}{\delta b_n} \, \hat{n}_k \hat{r}_k dS \\ &= \int_{S_\infty} \Psi_i \, \frac{\partial f_{ik}}{\partial b_n} \, \hat{n}_k dS + \int_{S_w} \frac{\delta p}{\delta b_n} \left(\Psi_{k+1} \;\; \hat{n}_k + \hat{n}_k \, \hat{r}_k \right) dS \end{aligned} \tag{4.55}$$

The first integral is on the outer boundaries of the domain and the second one on the boundary. In order to avoid computing the terms $\left(\frac{\partial f_{ik}}{\partial b_n}, \; \frac{\delta p}{\delta b_n} \right)$ their multipliers are set to zero, and so the adjoint boundary conditions are:
A. For the farfield boundaries (S_∞)

$$\overrightarrow{\Psi}|_{S_\infty} = \overrightarrow{0} \tag{4.56}$$

B. For the solid boundaries (S_w)

$$\Psi_{k+1}|_{S_w} \hat{n}_k + \hat{n}_k \, \hat{r}_k = 0 \tag{4.57}$$

Adjoint Ghost Cell Boundary Conditions

The adjoint boundary conditions are applied in the same way as the primal boundary conditions. By solving an equation as the eq. 4.30 the adjoint variables are copied at the ghost cells. After the copy, the variables need to be modified to impose the boundary condition. So, the final equation to be solved is

$$\Psi^s_{k+1} = \Psi_{k+1} - 2\Psi \cdot n_x - 2(\vec{n} \cdot \vec{r}) n_x \tag{4.58}$$

where $k = 1, 2$ and Ψ^s_{k+1} are the adjoint flow variables in the ghost cells.

4.5.4 Aerodynamic Shape Optimization

Application 1

The first application of the optimization software is on the airfoil shown in figure 4.27 whose shape is defined with 17 Bezier points.

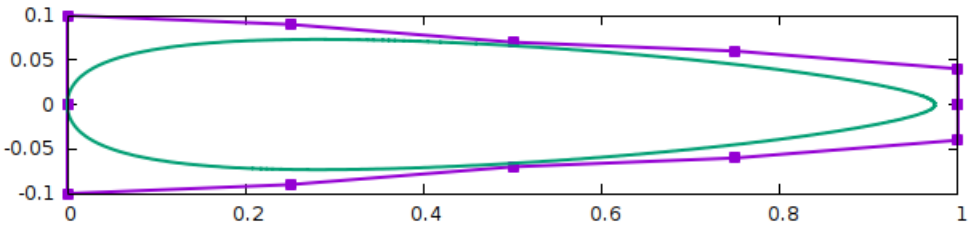


Figure 4.27: *Airfoil shape optimization. Initial shape.*

In order to keep the length of the airfoil the same, not all the points are allowed to move. The two points at the leading and trailing edge are fixed and the only points able to move are points 2, 3, 4, 12, 13, 14. The mesh around the airfoil is shown in figure 4.28. The airflow has velocity $V = 100\frac{m}{s}$ and zero angle of attack

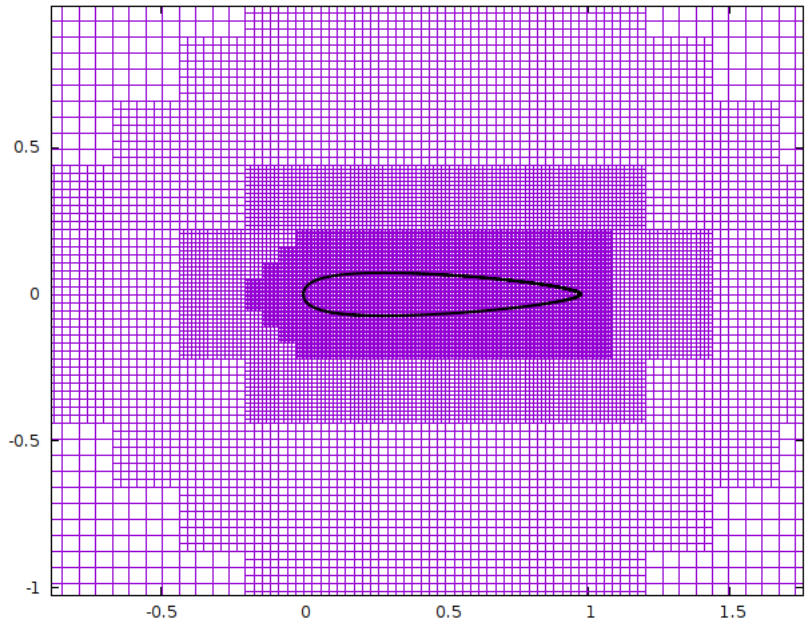


Figure 4.28: *Mesh around a NAcA0012 airfoil.*

Figure 4.29 shows the convergence graphs of the adjoint equations and figures 4.30 show the adjoint fields.

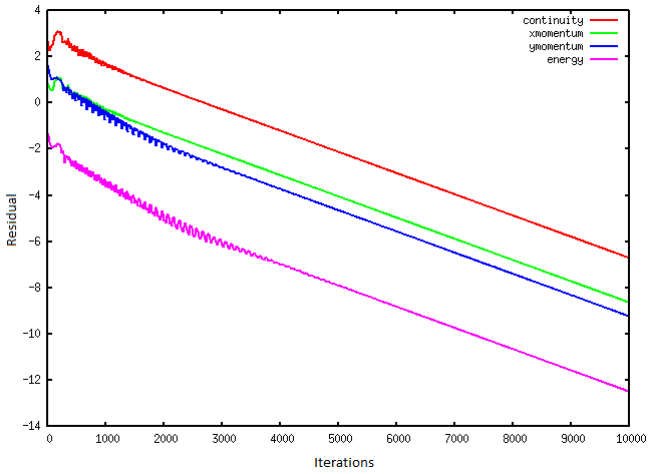


Figure 4.29: *Convergence of the adjoint equations*

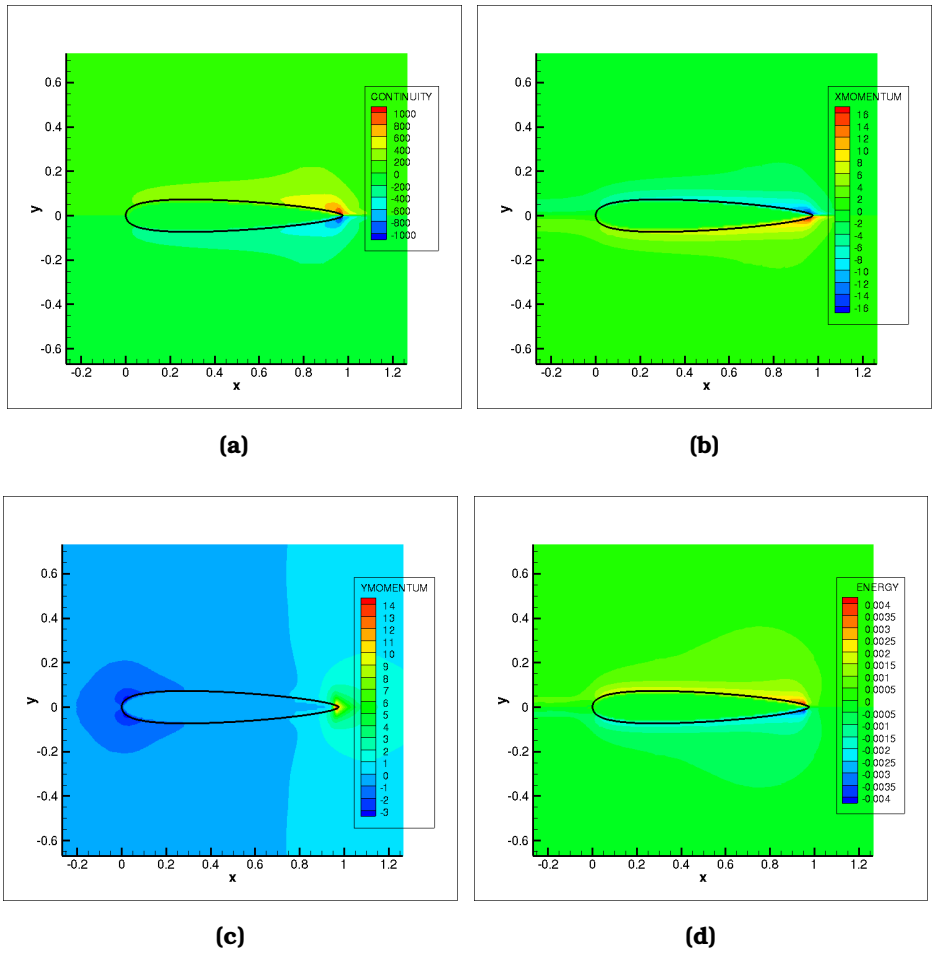


Figure 4.30: *Adjoint variables fields. a)adjoint density b)adjoint x-momentum c)adjoint y-momentum d)adjoint energy.*

Figure (4.31) shows the comparison between the starting and the optimized geometry.

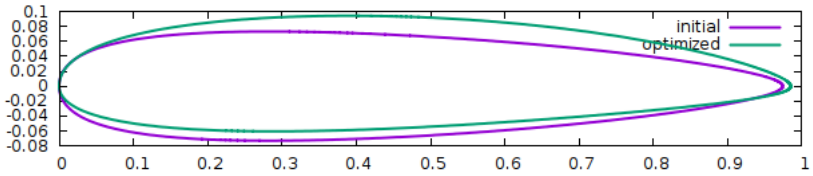


Figure 4.31: *Comparison of initial and new geometry.1*

The initial airfoil has Lift of $L_{init} = 0.0N$, end the optimized one $L_{opt} = 1529.01N$. The objective function is shown in figure 4.32.

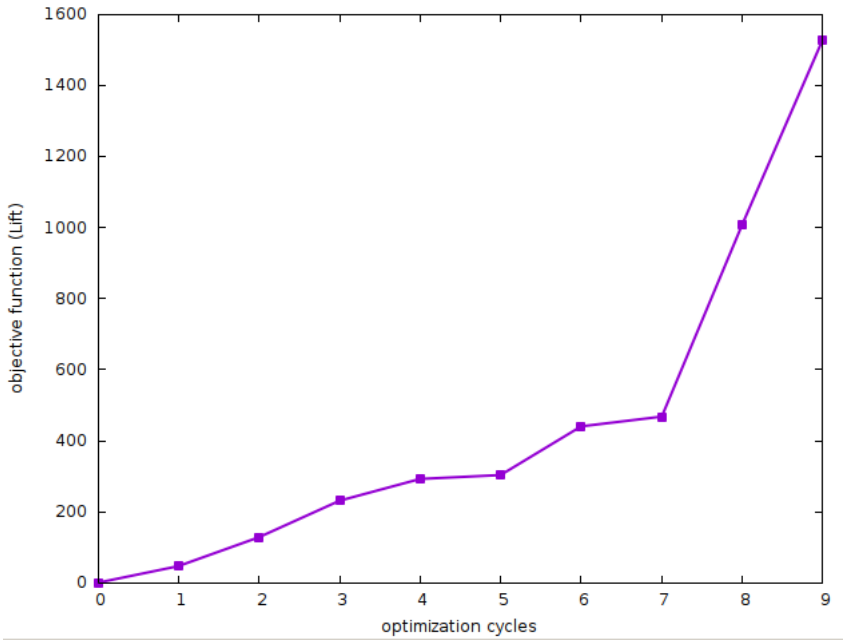


Figure 4.32: *Airfoil shape optimization. Change in the objective function during optimization.*

Because the flow is inviscid, the lift force will continue to increase as the optimization cycles continue, unless a constraint is imposed.

Application 2

The second application is on the airfoil shown in figure 4.33. The shape is again defined using 17 Bezier points.

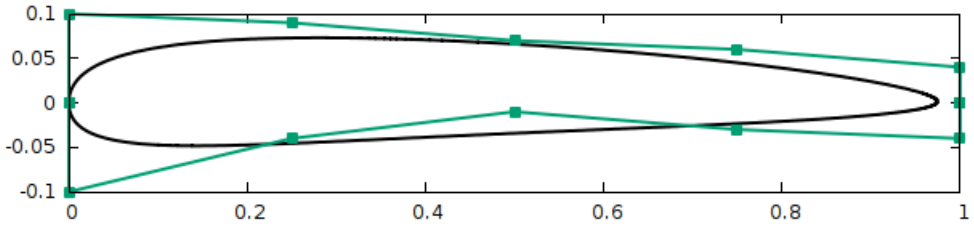


Figure 4.33: *Airfoil shape optimization. Initial shape.*

Same as before, only the points 2, 3, 4, 12, 13, 14 are allowed to move, while the length of the airfoil remains the same. Figure 4.34 shows the mesh around the airfoil..

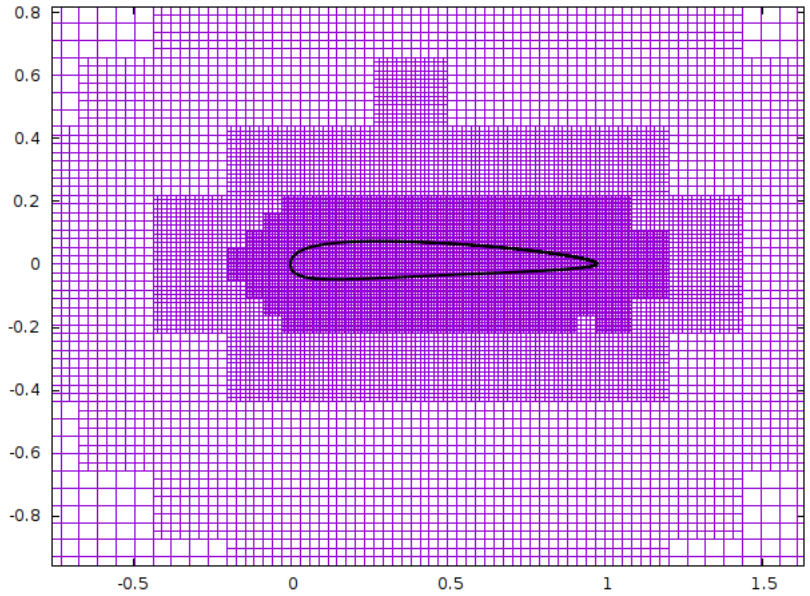


Figure 4.34: *Mesh around an airfoil.*

The flow has velocity $V = 100\frac{m}{s}$ and $a = 5^o$ angle of attack. Figure 4.35 shows the convergence of the adjoint equations and figures 4.30 show the adjoint flow variables.

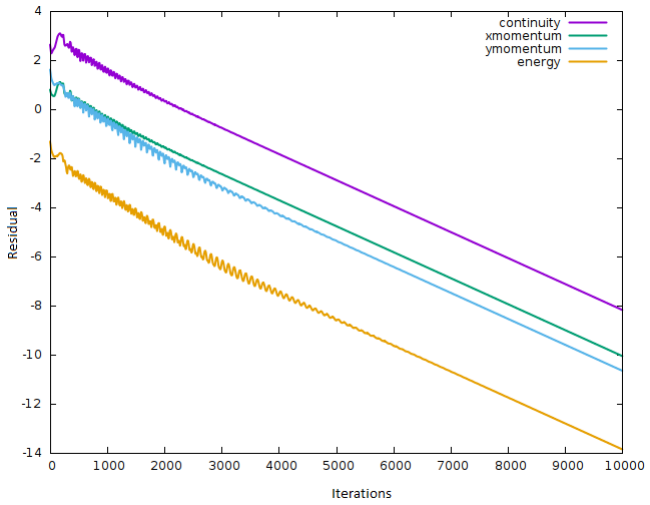


Figure 4.35: *Convergence of the adjoint equations.*

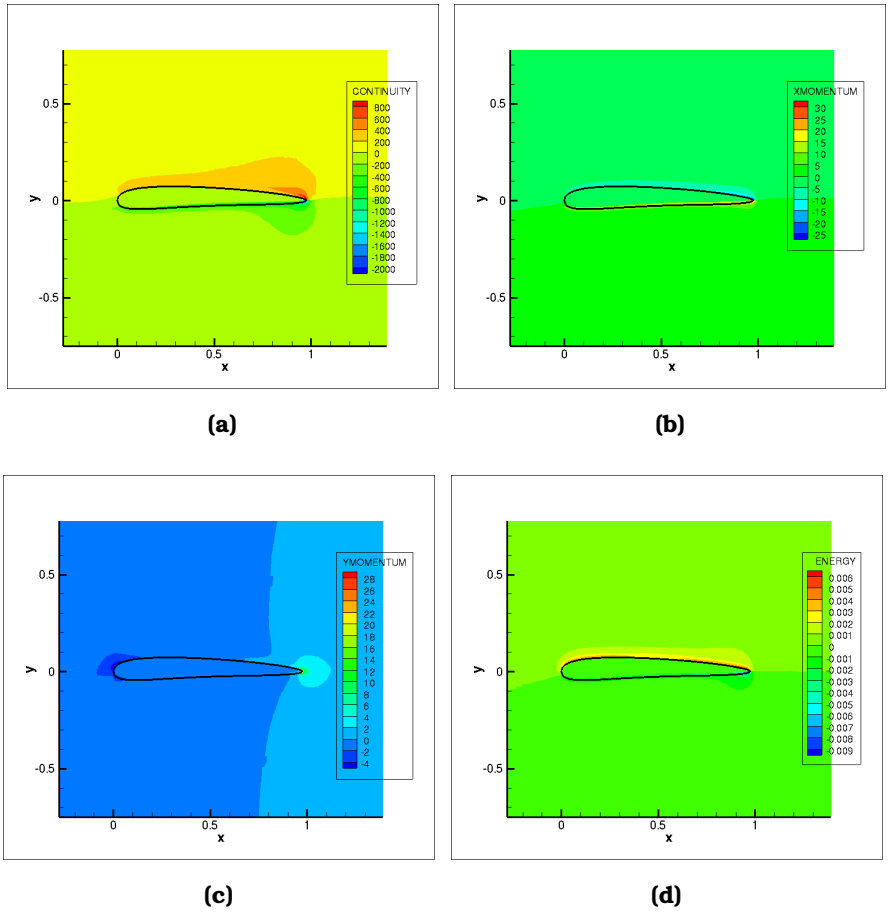


Figure 4.36: *Adjoint variables fields. a) adjoint density, b) adjoint x-momentum, c) adjoint y-momentum, d) adjoint energy.*

Figure 4.37) shows the comparison of the initial and the optimized geometry.

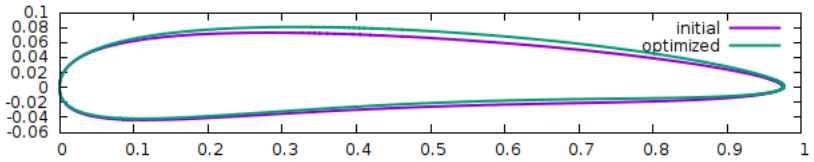


Figure 4.37: *Comparison of current and optimized.*

The lift of the initial airflow is $L_{init} = 1239.16N$, and the final $L_{opt} = 3613.87N$. The objective function during optimization is shown in figure 4.38.

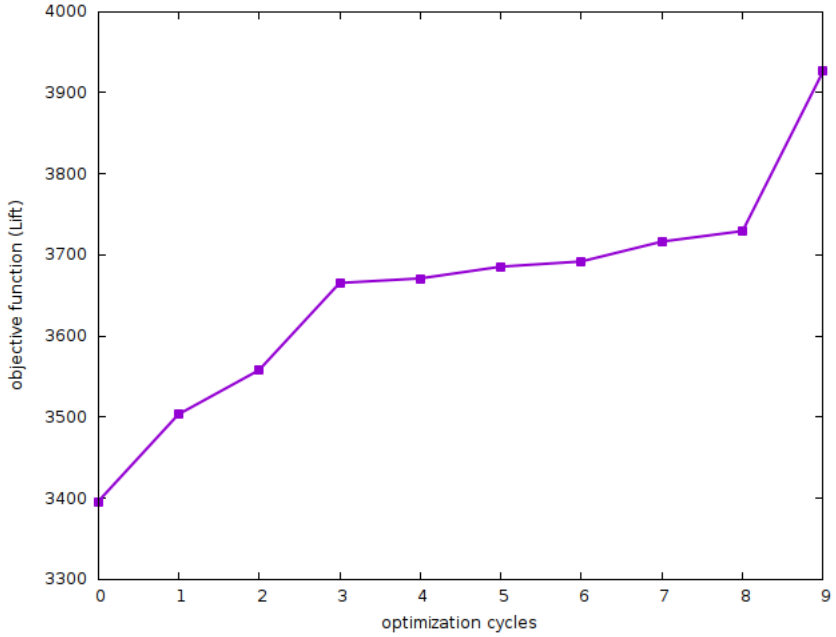


Figure 4.38: *Change in objective function during the optimization.*

as before, because the objective function is increasing, the optimization stops after 9 cycles.

Chapter 5

Summary and Conclusions

This chapter concludes the study by summarising the key points of the work presented herein together with the key findings. The main purpose of this work was to contribute to the development of primal and adjoint CFD solvers for application on aerodynamic shape optimization. For the first part of the thesis, conducted in RRD, the in-house CFD solver HYDRA was used. The main focus of the study was on cases where the primal equations were exhibiting limit cycle oscillations while converging. This lead to the adjoint equations diverging, which means that optimization for these cases was not possible. To solve this problem, a new source term was introduced into the primal equations, which is acting as a force, and forces the residual of the equations to converge to the mean value of the oscillations. The adjoint problem was formulated accordingly to include this source term. The software was tested on a case of flow around a compressor stator vane and, indeed, eliminated the oscillations from the primal solution but also made the adjoint equations to converge to a solution. This solution was used to derive the sensitivity derivatives.
For the second part of the thesis, an existing software of the NTUA was used. The software was initially able to solve inviscid flows, so as part of the thesis it was modified to solve viscous flows as well. It was then validated against analytical and experimental data found in the literature. Next, the program was parallelised usign the MPI protocol, in order to make the solving process faster. The parallel software was evaluated for its speed and efficiency in comparison to the serial software. Finally, the adjoint problem was derived using the continuous adjoint method. Two cases were used to evaluate the adjoint method, by performing aerodynamic shape optimization on two different airfoils.
The key conclusions from the entire study are the following:

- By adding an appropriate source term into the primal equations, the convergence of the primal problem can be optimized and achieve a more

stable solution in cases of LCO.

- The adjoint convergence is highly affected by the status of the primal solution. If the second one is oscillating and not stabilised, the adjoint problem cannot converge. The work here has proven that when the primal solution is stable, then adjoint convergence can be achieved.
- Making a software parallel improves the speed of the CFD solver significantly. Especially for industrial solvers, parallelisation is almost obligatory, otherwise the solving process is unsustainable. When using finer mesh for the computational domain, the benefit of the parallel software becomes bigger, because the communications between partitions are smaller.
- Generally, the ghost cell method imposes the boundary conditions indirectly, which means less accurately. However, the simulations herein show that the method can achieve very good results compared to other solvers which use body-fitted mesh or compared with literature data, at least for laminar flows.
- It is important to mention that the work here has set the ground for further development on both software. On HYDRA, there is a clear benefit on making the adjoint problem to converge, when before it wasn't possible. A sensitivity map was derived and optimization can now be perform on a case that couldn't converge in the past. In the second part of the thesis, an additional optimization software was created an added to the toolset of the LTT/NTUA, which handles cartesian grids and can be used to perform aerodynamic shape optimization.

Bibliography

[1] Metis - serial graph partitioning and fill-reducing matrix ordering.

[2] A partnership for next-generation computational fluid dynamics. https://www.airbus.com/en/newsroom/news/2018-01-a-partnership-for-next-generation-computational-fluid-dynam 2018.

[3] B. M. Aaron Vose and J. Levesque. Tri-hybrid computational fluid dynamics. Cray User Group, 41(2):363–386, 2014.

[4] S. R. Allmaras, F. T. Johnson, and P. R. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. 7th International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, Hawaii, 2012.

[5] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[6] W. K. Anderson and D. L. Bonhaus. Airfoil design on unstructured grids for turbulent flows. AIAA Journal, 37(2):185–191, 1999.

[7] H. Bishop Keller, N. Y New Yor, N. York, A. Mathematical Society, for Industrial, and S. aplied mathematis. Computational fluid dynamics / [edited by herbert b. keller]. SERBIULA (sistema Librum 2.0), 02 2019.

[8] H. Blasius. Grenrschichten in flussgkeiten mit kleiner reibung. Zeitschrift für Angewandte Mathematik und Physik, 41(2):363–386, 1908.

[9] J. V. Boussinesq. Théorie de l'écoulement tourbillonnant et tumultueux des liquides dans les lits rectilignes à grande section. Comptes Rendus des Séances de l'Académie des Sciences, 122:1290–1295, 1877.

[10] J. Chung, G. Martin, and K. D. Lee. Aerodynamic design of 3D compressor blade using an adjoint method. AIAA Paper 2004-0027, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, 2004.

[11] D. L. Davidson. The enterprise-wide application of computational fluid dynamics in the chemicals industry. Proceedings of the 6th World Congress of Chemical Engineering, 41(2):363–386, 1851.

[12] D.Chatzinikolaou. Development and Application of the Ghost-Cell Method for 2D Steady Inviscid Flows of Compressible Fluids (in Greek). Diploma Thesis. Laboratory of Thermal Turbomachines, N.T.U.A, 2017.

[13] L. Euler. Principes generaux de l'etat d'equilibre des fluides;" "principes generaux du mouvement des fluides;" "continuation des recherches sur la theorie du mouvement des fluides.". Histoire de l'Acacdernie de Berlin, 41(2):363–386, 1755.

[14] C. Frey, H. P. Kersken, and D. Nürnberger. The discrete adjoint of a turbomachinery RANS solver. ASME Paper GT2009-59062, ASME Turbo Expo. Orlando, Florida, 2009.

[15] M. B. Giles. On the use of Runge-Kutta time-marching and multigrid for the solution of steady adjoint equations. Technical Report NA00/10, Computing Laboratory, University of Oxford, UK, 2000.

[16] M. B. Giles, M. C. Duta, J. D. Müller, and N. A. Pierce. Algorithm developments for discrete adjoint methods. AIAA Journal, 41(2):198–205, 2003.

[17] M. B. Giles and N. A. Pierce. Adjoint equations in CFD-Duality, boundary conditions and solution behavior. AIAA Paper 1997-1850, 13th AIAA Computational Fluid Dynamics Conference, Snowmass Village, Colorado, 1997.

[18] H. H. Goldstine. A History of the Calculus of Variations from the 17th through the 19th Century. Springer New York, NY, 1980.

[19] A. Jameson. Multigrid algorithms for compressible flow calculations. In Multigrid Methods II. Lecture Notes in Mathematics, volume 1228, pages 166–201. Springer Berlin Heidelberg, 1986.

[20] A. Jameson, E. Schmidt, and E. Turkel. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper, 14th Fluid and Plasma Dynamics Conference, Palo Alto, California, 1981.

[21] A. Jameson, S. Shankaran, and L. Martinelli. Continuous adjoint method for unstructured grids. AIAA Journal, 46(5):1226–1239, 2008.

[22] D. J. Mavriplis. Discrete adjoint-based approach for optimization problems on three-dimensional unstructured meshes. AIAA Journal, 45(4):740–750, 2007.

[23] A. Milli and S. Shahpar. PADRAM: Parametric design and rapid meshing system for complex turbomachinery configurations. ASME Paper GT2012-69030, ASME Turbo Expo. Copenhagen, Denmark, 2012.

[24] C. Misev. Development and Optimization of an Implicit CFD Solver in Hydra. PhD thesis, University of Surrey, UK, 2017.

[25] M.Martin. Measurement technology for micro-scale aerodynamics. National Renewable Energy Laboratory, pages 1–6, 2007.

[26] S. K. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. AIAA Paper 2000-0667, 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, 2000.

[27] L. M. H. Navier. Memoire sur les lois du mouvement des fluides. Memoires de l'Academie Royale des Sciences vol.6, 41(2):363–386, 1827.

[28] G. Ntanakas. Unsteady Discrete Adjoint Method Formulated in the Time-Domain for Shape Optimization in Turbomachinery. PhD thesis, National Technical University of Athens, Greece, 2018.

[29] E. Padway and D. Mavriplis. Toward a pseudo-time accurate formulation of the adjoint and tangent systems. AIAA Paper 2019-0699, AIAA SciTech Forum, San Diego, California, 2019.

[30] D. Pan. An immersed boundary method for incompressible flows using volume of body function. International Journal for Numerical Methods in Fluids, 50(6):733–750, 2006.

[31] D. Pan and T. Shen. Computation of incompressible flows with immersed bodies by a simple ghost cell method. International Journal for Numerical Methods in Fluids, 60:1378 – 1401, 08 2009.

[32] D. I. Papadimitriou and K. C. Giannakoglou. Compressor blade optimization using a continuous adjoint formulation. ASME Paper GT2006-90466, ASME Turbo Expo. Barcelona, Spain, 2006.

[33] D. I. Papadimitriou and K. C. Giannakoglou. Total pressure loss minimization in turbomachinery cascades using a new continuous adjoint formulation. Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy, 221(6):865–872, 2007.

[34] C. Peskin. Flow patterns around heart valves: a digital computer method for solving the equations of motion. IEEE Transactions on Biomedical Engineering, BME-20(4):316–317, 1973.

[35] J. E. V. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. Computers & Fluids, 39(3):373–391, 2010.

[36] N. A. Pierce and M. B. Giles. Preconditioned multigrid methods for compressible flow calculations on stretched meshes. Journal of Computational Physics, 136(2):425–445, 1997.

[37] O. Pironneau. Optimal Shape Design for Elliptic Systems. Springer Series in Computational Physics, 1984.

[38] L. Prandtl. Über flüssigkeitsbewegung bei sehr kleiner reibungen. Verhandlungen des dritten internationalen Mathematiker- Kongresses in Heidelberg, 41(2):363–386, 1905.

[39] O. Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. Philosophical Transactions of the Royal Society of London, 41(2):363–386, 1895.

[40] G. R.Mittal. Immersed boundary methods. Annual review of fluid mechanics, 2005:1069–1082.

[41] P. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics, 43(2):357–372, 1981.

[42] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics, 43(2):357–372, 1981.

[43] A. Rohde. Eigenvalues and eigenvectors of the euler equations in general geometries. 15th AIAA Computational Fluid Dynamics Conference, pages 1–6, 2001.

[44] K. Samouchos. The Cut-Cell Method for the Prediction of 2D/3D Flows in Complex Geometries and the Adjoint-Based Shape Optimization. PhD thesis, National Technical University of Athens, Greece, 2022.

[45] STAR-CCM+ Solves Aerodynamics and Heat in Tesla Model S. https://www.digitalengineering247.com/article/star-ccm-solves-aerodynamics-and-heat-in-tesla-model-s. 2010.

[46] G. G. Stokes. On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids. On the Theories of the Internal Friction of Fluids in Motion, and of the Equilibrium and Motion of Elastic Solids, 41(2):363–386, 1851.

[47] R. Swanson and S. Langer. Steady-state laminar flow solutions for naca 0012 airfoil. Computers & Fluids, 126:102 – 128, 2016.

[48] R. C. Swanson, E. Turkel, and R. C. C. Convergence acceleration of Runge-Kutta schemes for solving the Navier-Stokes equations. Journal of Computational Physics, 224(1):365–388, 2007.

[49] R. C. Swanson, E. Turkel, and S. Yaniv. Analysis of a RK/implicit smoother for multigrid. In Computational Fluid Dynamics 2010, pages 409–417. Springer Berlin Heidelberg, 2011.

[50] The CFD Solver CODA and the Sparse Linear Systems Solver Spliss. https://elib.dlr.de/143414/1/2021-07-29_HLRN_CODAResults.pdf. 2018.

[51] The HYDRA code - Rolls Royce's standard aerodynamic tool. https://www.mpls.ox.ac.uk/research-section/the-hydra-code-rolls-royces-standard-aerodynamic-design-tool. 2012.

[52] I. Vasilopoulos. CAD-based and CAD-free Aerodynamic Shape Optimization of Turbomachinery Blade Rows using the Adjoint Method. PhD thesis, NTUA, 2020.

[53] D. X. Wang and L. He. Adjoint aerodynamic design optimization for blades in multistage turbomachines–Part I: Methodology and verification. Journal of Turbomachinery, 132(2):021011, 2010.

[54] H.-Y. Wu, F. Liu, and H.-M. Tsai. Aerodynamic design of turbine blades using an adjoint equation method. AIAA Paper 2005-1006, 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, 2005.

[55] S. Yang, H.-Y. Wu, F. Liu, and H.-M. Tsai. Aerodynamic design of cascades by using an adjoint equation method. AIAA Paper 2003-1068, 41st AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, 2003.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Ανάπτυξη Πρωτεύοντων και Συζυγών Επιλυτών για Χρήση στην Αεροδυναμική Βελτιστοποίηση Μορφής.

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
'Υπολογιστική Μηχανική'

Μεταπτυχιακή Εργασία

Δανάη Χατζηνικολάου

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2022

Κεφάλαιο 1

Περίληψη

Η μεταπτυχιακή εργασία ασχολείται με την ανάπτυξη πρωτεύοντων και συζυγών επιλυτών για χρήση στην αεροδυναμική βελτιστοποίηση μορφής. Χωρίζεται σε δύο κύρια μέρη: το πρώτο μέρος έγινε στην Rolls-Royce Deutschland με τη χρήση του επιλυτή CFD, HYDRA. Πρόκειται για έναν επιλύτη 3Δ μόνιμης συνεκτικής ροής, ο οποίος επιλύει τις εξισώσεις RANS, σε συνδυασμό με το μοντέλο τύρβης Spalart-Almaras, σε μη-δομημένο πλέγμα με τη μέθοδο πεπερασμένων όγκων. Το λογισμικό σε συνδυασμό με τη διακριτή συζυγή μέθοδο χρησιμοποιείται στην αεροδυναμική βελτιστοποίηση μορφής σε βιομηχανικές εφαρμογές στροβιλομηχανών. Η εργασία ασχολείται με περιπτώσεις κατά τις οποίες η σύγκλιση του πρωτεύοντος προβλήματος περιπίπτει σε ταλαντώσεις με συνέπεια το συζυγές πρόβλημα να αποκλίνει και να μην μπορούν να παραχθούν οι παράγωγοι ευαισθησίας. Εισάγεται ένας όρος πηγής στις εξισώσεις του πρωτεύοντος προβλήματος έτσι ώστε να μειωθούν οι ταλαντώσεις, και στη συνέχεια ο ίδιος όρος διαφορίζεται ώστε να εισαχθεί στο συζυγές πρόβλημα. Ο κώδικας εφαρμόζεται στη ροή γύρω από ένα πτερύγιο συμπιεστή με δύο διαφορετικές γωνίες ροής.
Το δεύτερο μέρος της εργασίας συντάχθηκε στο Εθνικό Μετσόβιο Πολυτεχνείο χρησιμοποιώντας τον κώδικα της ΜΠΥΡΒ/ΕΜΠ ο οποίος επιλύει 2Δ μόνιμες ροές σε καρτεσιανά πλέγματα με τη μέθοδο των ψευδο-κυψελών και ο οποίος συνδυασμένος με τη συνεχή συζυγή μέθοδο χρησιμοποιείται για την αεροδυναμική βελτιστοποίηση μορφής. Αρχικά, ένας υπάρχον επιλύτης ροής για ατριβείς ροές τροποποιείται ώστε να χειρίζεται και συνεκτικές ροές και, στη συνέχεια, πιστοποιείται με την προσομοίωση μιας ροής πάνω από μια επίπεδη πλάκα και γύρω από μια αεροτομή NACA0012. Τα αποτελέσματα συγκρίνονται με βιβλιογραφικά δεδομένα. Στη συνέχεια, το λογισμικό παραλληλοποιείται και αξιολογείται η απόδοσή του. Στο τέλος, διατυπώνεται το συζυγές πρόβλημα με τη χρήση της συνεχούς συζυγούς μεθόδου για τη χρήση σε αεροδυναμική βελτιστοποίηση μορφής και αξιολογείται κάνοντας βελτιστοποίηση μορφής σε δύο διαφορετικές αεροτομές, με συνάρτηση κόστους τη μεγιστοποίηση της άνωσης.

1.1 Βελτίωση σύγκλισης συζυγούς προβλήματος μόνιμων ροών

Για πρώτο μέρος της εργασίας χρησιμοποιείται το λογισμικό HYDRA της Rolls-Royce Deutschland. Το λογισμικό επιλύει τις εξισώσεις RANS για 3Δ μόνιμες συνεκτικές ροές στη μορφή

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{F}^{inv}(\vec{U}) + \nabla \cdot \vec{F}^{visc}(\vec{U}) = \vec{S} \tag{1.1}$$

όπου $\vec{U} = [\rho, \rho u, \rho v, \rho w, \rho E]$ είναι το διάνυσμα των συντηρητικών μεταβλητών. Τα διανύσματα ροής $\vec{F}^{inv}$ και $\vec{F}^{visc}$ περιλαμβάνουν τους μη-συνεκτικούς και συνεκτικούς όρους των εξισώσεων αντίστοιχα. Συγκεκριμένα

$$\vec{F}^{inv} = \begin{bmatrix} \rho\vec{v} \\ \rho\vec{v}u + p\vec{e}_x \\ \rho\vec{v}v + p\vec{e}_y \\ \rho\vec{v}w + p\vec{e}_z \\ \rho\vec{v}E + p\vec{v} \end{bmatrix}, \qquad \vec{F}^{visc} = -\begin{bmatrix} 0 \\ \vec{\tau}_1 \\ \vec{\tau}_2 \\ \vec{\tau}_3 \\ \vec{\tau}_k v_k + q_k \end{bmatrix}, \qquad \vec{S} = \begin{bmatrix} S_c \\ S_{m1} \\ S_{m2} \\ S_{m3} \\ S_e \end{bmatrix} \tag{1.2}$$

όπου $\vec{\tau}_k = [\tau_{k1}, \tau_{k2}, \tau_{k3}]^{\mathrm{T}}$ και από το νόμο του Fourier $\vec{q} = k\nabla T$. Οι εξισώσεις συμπληρώνονται από το μοντέλο τύρβης Spalart-Almaras (SA) που εισάγει την SA μεταβλητή $\tilde{\nu}$, η οποία υπολογίζεται από την εξίσωση

$$\begin{aligned} \frac{\partial \tilde{\nu}}{\partial t} + \underbrace{\frac{\partial(\tilde{\nu} v_k)}{\partial x_k}}_{Conv} &= \underbrace{\frac{1}{\sigma}\left[\frac{\partial}{\partial x_k}\left((\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_k}\right) + c_{b2}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right]}_{Diff} \\ &+ \underbrace{c_{b1}\tilde{\Omega}\tilde{\nu}}_{Prod} - \underbrace{(c_{w1} f_w)\left(\frac{\tilde{\nu}}{d}\right)^2}_{Destr} \end{aligned} \tag{1.3}$$

όπου ο όρος με *Conv* δηλώνει τη μεταφορά της μεταβλητής $\tilde{\nu}$, ο όρος *Diff* την μοριακή και τυρβώδη διάχυση, οι όροι *Prod*, *Destr* μοντελοποιούν την παραγωγή και καταστροφή της τύρβης. Η πλεγματοποίηση του υπολογιστικού χωρίου γίνεται με το λογισμικό PADRAM της RRD χρησιμοποιώντας μη-δομημένο οριόδετο πλέγμα. Οι εξισώσεις ροής διακριτοποιούνται στους κόμβους του πλέγματος με τη μέθοδο των πεπερασμένων όγκων και επιλύονται με τη μέθοδο Runge-Kutta τρίτης τάξης. Στη συνέχεια, η διακριτή συζυγής μέθοδος χρησιμοποιείται για τον υπολογισμό των συζυγών μεταβλητών και των παραγώγων ευαισθησίας με σκοπό την αεροδυναμική βελτιστοποίηση μορφής.

1.1.1 Μέθοδος της ¨ωμής¨ δύναμης (Brute Force Method) για περιπτώσεις σύγκλισης οριακού κύκλου

Όταν η σύγκλιση εμπίπτει σε οριακό κύκλο σημαίνει ότι παρατηρούνται ταλαντώσεις συγκεκριμένου πλάτους το οποίο δεν αποσβένει. Παρατηρείται ότι όταν η σύγκλιση του πρωτεύοντος προβλήματος έχει αυτήν τη συμπεριφορά, το συζυγές πρόβλημα αποκλίνει.
Για την αντιμετώπιση του προβλήματος, χρησιμοποιείται η ιδέα που αναπτύχθηκε εσωτερικά στην RRD και ονομάζεται μέθοδος ¨ωμής¨ δύναμης, Brute Force Method. Στόχος της μεθόδου είναι να εισάγει έναν όρο πηγής στις εξισώσεις ροής, ο οποίος θα αναγκάσει τη λύση να συγκλίνει στη μέση τιμή των ταλαντώσεων και ο οποίος όταν παραγωγιστεί και εισαχθεί στις συζυγείς εξισώσεις, θα τις αναγκάσει να συγκλίνουν. Για τον υπολογισμό του, αρχικά υπολογίζεται η μέση τιμή των μεταβλητών ροής

$$\bar{u} = \frac{\sum_{i=1}^{N} v_i}{N} \tag{1.4}$$

όπου N είναι ο αριθμός των τελευταίων επαναλήψεων που χρησιμοποιήθηκαν για να υπολογιστεί η μέση τιμή. Ο ακριβής αριθμός θα υπολογιστεί αργότερα, με βάση τις εκάστοτε εφαρμογές που γίνονται. Ο όρος πηγής έχει τη μορφή

$$\vec{S_f} = \frac{1}{\tau} \begin{bmatrix} \rho - \bar{\rho} \\ u - \bar{u} \\ v - \bar{v} \\ w - \bar{w} \\ E - \bar{E} \end{bmatrix} \tag{1.5}$$

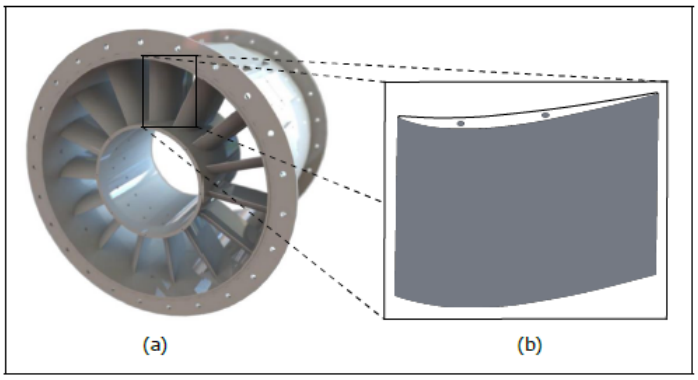
όπου η μπάρα δηλώνει τη μέση τιμή της μεταβλητής η οποία υπολογίστηκε από την εξίσωση 1.4 και τ είναι μια παράμετρος με μονάδες χρόνου η τιμή της οποίας επιλέγεται μετά από παραμετρική ανάλυση. Ο όρος πηγής διακριτοποιείται στους κόμβους του πλέγματος ως

$$S_{fi} = \frac{u_i - \bar{u}_i}{\tau} V_i \tag{1.6}$$

και εισάγεται στις εξισώσεις Runge-Kutta. Οι εξισώσεις ροής λύνονται αρχικά χωρίς τον όρο πηγής και κατά τη διάρκεια των N τελευταίων επαναλήψεων, αφού η λύση εμπίπτει σε οριακό κύκλο, υπολογίζεται η μέση τιμή των μεταβλητών ροής. Μετά το πέρας των N επαναλήψεων, ο όρος πηγής εισάγεται στις εξισώσεις οι οποίες συνεχίζονται να λύνονται μέχρι να επιτευχθεί μεγαλύτερη σύγκλιση.

1.1.2 Εφαρμογές

Η γεωμετρία που χρησιμοποιήθηκε για την εφαγμογή της μεθόδου είναι ο στάτορας ενός συμπιεστή όπως σχεδιάστηκε στο Τεχνικό Πανεπιστήμιο του Βερολίνου TUB και φαίνεται στο Σχήμα 1.1



Σχήμα 1.1: *Στάτορας συμπιεστή σχεδιασμένος στο Τεχνικό Πανεπιστήμιο του Βερολίνου TUB.*

.

Το υπολογιστικό πλέγμα αποτελείται από περίπου $2 \cdot 10^6$ κόμβους. Αρχικά το πτερύγιο εκτίθεται σε ροή 46^o το οποίο είναι 4^o περισσότερο από τη τιμή για την οποία έχει σχεδιαστεί. Το πρωτεύον και το συζυγές πρόβλημα της εφαρμογής αυτής συγκλίνουν ακόμα και χωρίς την εισαγωγή του όρου πηγής. Η εφαργμογή γίνεται για πιστοποίηση της μεθόδου και για να χρησιμοποιηθούν τα αποτελέσματα για σύγκριση. Η δεύτερη περίπτωση, για την οποία αυτή τη φορά το συζυγές πρόβλημα χωρίς τον όρο πηγής δε συγκλίνει, είναι η περίπτωση όπου το πτερύγιο εκτίθεται σε ροή 5^o μεγαλύτερη από τη γωνία σχεδιασμού. Η μέθοδος Βρυτε Φορσε χρησιμοποιείται εδώ έτσι ώστε να επιτευχθεί σύγκλιση των συζυγών εξισώσεων. Τα αποτελέσματα συγκρίνονται με εκείνα της πρώτης εφαρμογής ($+4^o$).

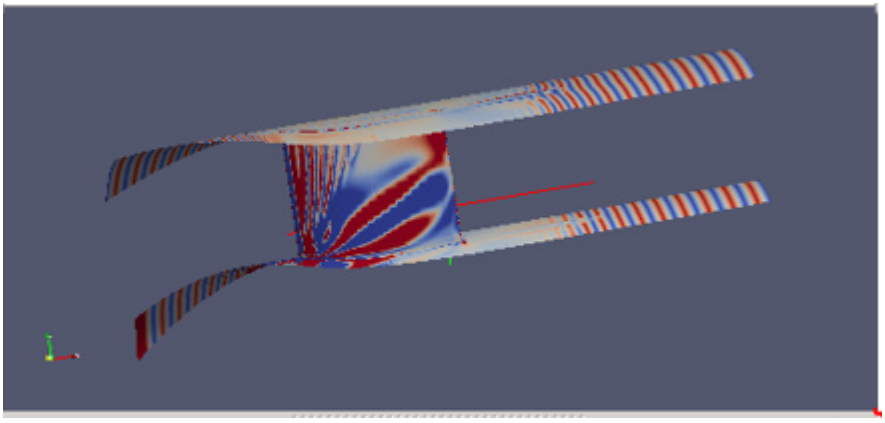
1.1.3 Η περίπτωση $+4^o$.

Αρχικά, χωρίς την εισαγωγή του όρου πηγής, οι πρωτεύουσες και συζυγείς εξισώσεις συγκλίνουν χωρίς ταλαντώσεις. Μετά την εισαγωγή του όρου πηγής, οι εξισώσεις συνεχίζουν να συγκλίνουν με γρηγορότερο ρυθμό και επιτυγχάνεται βαθύτερη σύγκλιση. Στη συνέχεια, επιλύεται το συζυγές πρόβλημα και υπολογίζονται οι παράγωγοι ευαισθησίας. Η συνάρτηση-στόχος προς

ελαχιστοποίηση είναι ο συντελεστής απωλειών ολικής πίεσης

$$\omega = \frac{p_{t,inlet} - p_{t,outlet}}{p_{t,inlet}} \tag{1.7}$$

Χρησιμοποιώντας το συζυγές πεδίο υπολογίζονται οι παράγωγοι ευαισθησίας, οι οποίες φαίνονται στο Σχήμα 1.2

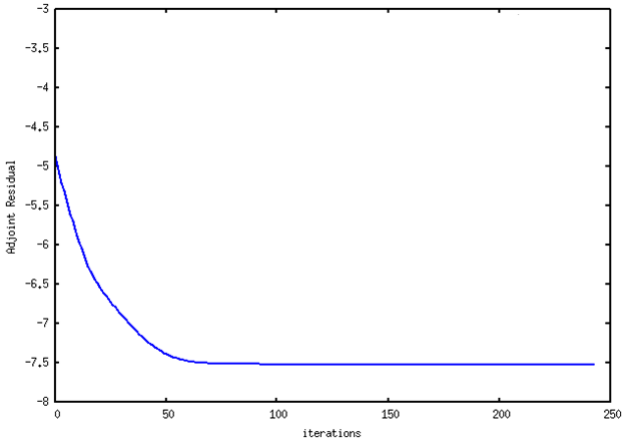


Σχήμα 1.2: *Εφαρμογή* $+4^o$. *Παράγωγοι ευαισθησίας.*

Στο Σχήμα φαίνεται ποιες περιοχές πρέπει να μετακινηθούν ώστε να μειωθεί ο συντελεστής απωλειών. Με κόκκινο χρώμα φαίνονται οι περιοχές που πρέπει να μετακινηθούν προς τα έξω, ενώ με μπλε φαίνονται εκείνες που πρέπει να μετακινηθούν προς τα μέσα.

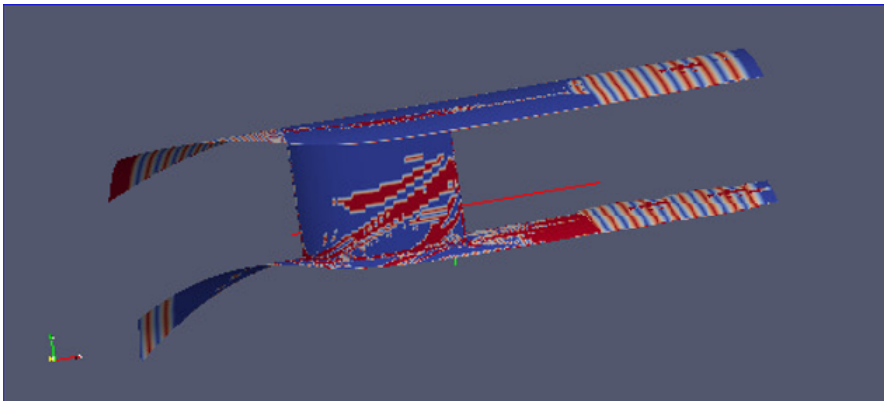
1.1.4 Η περίπτωση $+5^o$.

Σε αυτήν την εφαρμογή η σύγκλιση του πρωτεύοντος προβλήματος εμπίπτει σε οριακό κύκλο και κατά συνέπεια το συζυγές πρόβλημα αποκλίνει. Μετά την εφαρμογή της μεθόδου απότομης δύναμης, το πρωτεύον πρόβλημα συγκλίνει χωρίς ταλαντώσεις στην ίδια λύση που συγκλίνουν και οι εξισώσεις χωρίς τον όρο πηγής. Το συζυγές πρόβλημα επίσης φαίνεται να συγκλίνει και το υπόλοιπο των εξισώσεων μειώνεται κατά 2.5 τάξεις μεγέθους όπως φαίνεται στο Σχήμα 1.3.



Σχήμα 1.3: *Σύγκλιση RMS συζυγών εξισώσεων με τη μέθοδο της ¨ωμής¨ δύναμης.*

Οι ισογραμμές των παραγώγων ευαισθησίας φαίνονται στο Σχήμα 1.4.



Σχήμα 1.4: *Εφαρμογή* $+5^o$. *Παράγωγοι ευαισθησίας.*

Παρατηρείται ότι τιμές των παραγώγων είναι σχετικά χαμηλές, σχεδόν μηδενικές σε κάποια σημεία. Αυτό θα μπορούσε να οφείλεται στο γεγονός ότι παρόλο που το συζυγές πρόβλημα συγκλίνει, το υπόλοιπο μειώνεται κατά 2.5 τάξεις μεγέθους έως την τάξη του 10^{-14}, ενώ το υπόλοιπο του πρωτεύοντος προβλήματος έχει μειωθεί κατά 9 τάξεις έως την τάξη του 10^{-14}.
Το σημαντικότερο όμως όσο αφορά στις παραγώγους ευαισθησίας είναι το πρόσημό τους και όχι τόσο το μέτρο τους. Συγκρίνοντας τις δύο περιπτώσεις +4 και +5 γωνια φαίνεται ότι οι 2 περιπτωσεις έχουν παρόμοιες παραγώγους όσο αφορά στο πρόσημο.

1.2 Προγραμματισμός Παράλληλου Λογισμικού Πρόλεξης Ροών και της Συνεχούς Συζυγούς Μεθόδου με τη Μέθοδο των Ψευδο-κυψελών. Εφαρμογές στην Αεροδυναμική Βελτιστοποίηση Μορφής.

Για το δεύτερο μέρος της εργασίας χρησιμοποιήθηκε ο κώδικας υπολογιστικής ρευστοδυναμικής του εργαστηρίου ΜΠΥΡΒ/ΕΜΠ. Ο κώδικας επιλύει 2Δ μόνιμες ατριβείς ροές σε καρτεσιανά πλέγματα, με τη μέθοδο των ψευδο-κυψελών. Αρχικά, οι όροι των τάσεων εισάγονται στις εξισώσεις ροής ώστε ο κώδικας να διαχειρίζεται και συνεκτικές ροές. Οι εξισώσεις ροής ισχύουν στη μορφή

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{f_x}}{\partial x} + \frac{\partial \vec{f_y}}{\partial y} - \frac{\partial \vec{f_x^v}}{\partial x} - \frac{\partial \vec{f_y^v}}{\partial y} = 0 \tag{1.8}$$

όπου

$$f_k^v = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ u_i \tau_{ik} + q_k \end{bmatrix} \tag{1.9}$$

Ο τανυστής των τάσεων υπολογίζεται από τη σχέση

$$\tau_{ij} = \mu \left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \delta_{ij} \frac{\partial u_k}{\partial x_k} \right] \tag{1.10}$$

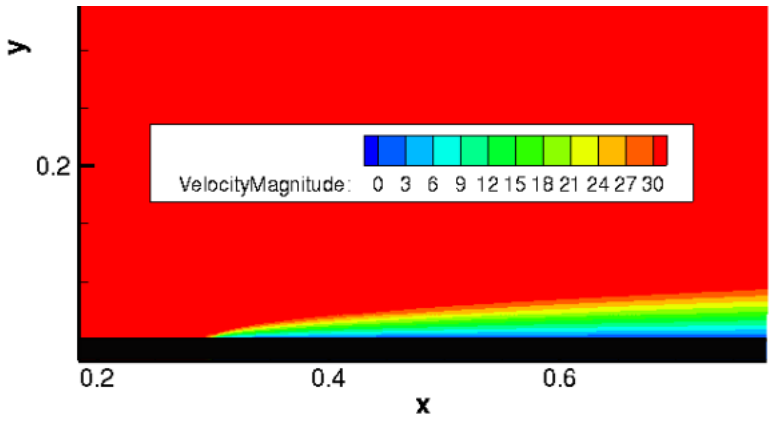
όπου το δ_{ij} είναι το δέλτα του $Kronecker$ και το μ ο συντελεστής δυναμικής συνεκτικότητας. Για την επιβολή της οριακής συνθήκης μη εισχώρησης χρησιμοποιείται η μέθοδος των ψευδο-κυψελών, κατά την οποία, στις κυψέλες που βρίσκονται μέσα στο στερεό και κοντά στο όριο δίνονται κατάλληλα μεγέθη ροής ώστε να προσομοιωθεί η επίδραση του στερεού στη ροή. Αρχικά, η ταχύτητα κοντά στο στερό όριο αντιγράφονται μέσα στις ψευδο-κυψέλες (εσωτερικά του στερεού) και στη συνέχεια αντιστρέφεται η κατεύθυνσή της. Έτσι, ακριβώς πάνω στο στερεό όριο, η ταχύτητα είναι μηδενική. Η εξίσωση που λύνεται μέσα στις ψευδο-κυψέλες για την αντιγραφή των μεγεθών ροής είναι

$$U_i^{n+1} = U_i^n - \Delta t \frac{\partial U_i^n}{\partial x_k} \frac{\partial \Phi}{\partial x_k} \tag{1.11}$$

όπου U_i τα μεγέθη ροής στην ψευδο-κυψέλη i και Φ η απόσταση της από το στερεό όριο.

1.2.1 Αξιολόγηση προγραμματισθέντος

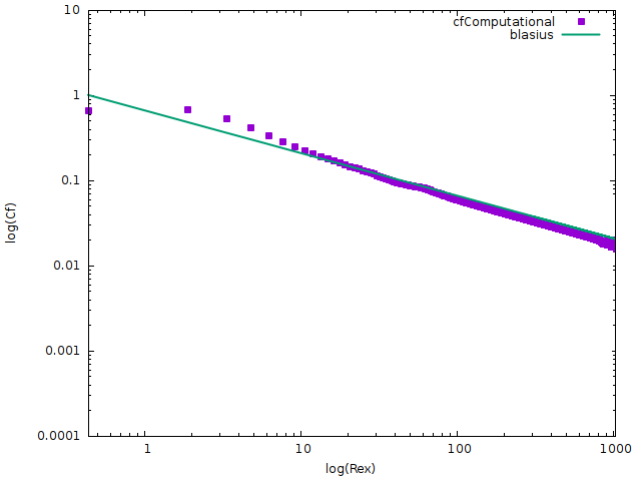
Ο κώδικας αξιολογείται σε δύο εφαρμογές: στη ροή πάνω σε επίπεδη πλάκα και στη ροή γύρω από συμμετρική αεροτομή. Εδώ παρουσιάζονται τα αποτελέσματα μόνο της πρώτης εφαρμογής. Για ροή πάνω σε επίπεδη πλάκα με γωνία ροής μηδενική, ολική πίεση εισόδου $P_{tin} = 1.03bar$ και θερμοκρασία $T_{tin} = 300K$ και στατική πίεση εξόδου $P_{sout} = 1.02bar$, αριθμό Reynolds $R = 1093$ και $Mach = 0.086$ το πεδίο ταχύτητας που προκύπτει είναι



Σχήμα 1.5: *Ροή σε Επίπεδη Πλάκα. Μέτρο ταχύτητας.*

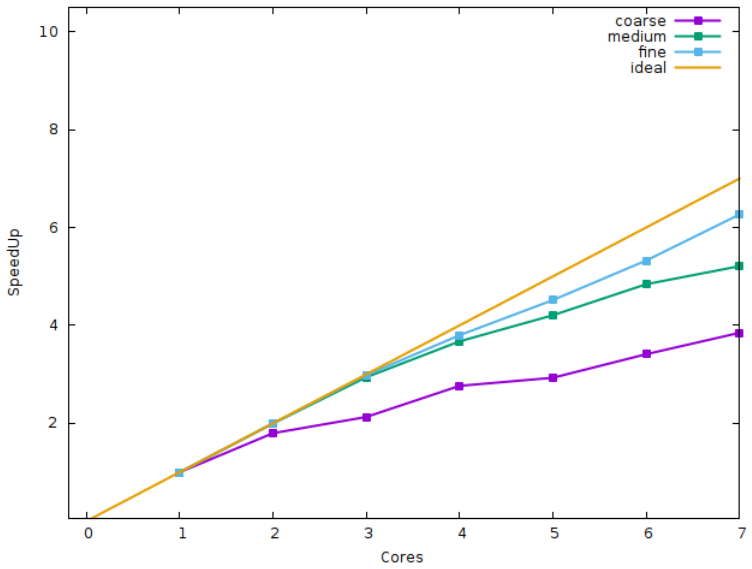
Ο συντελεστής τριβής της πλάκας συγκρίνεται με την αναλυτική λύση του Blasius όπως φαίνεται στο Σχήμα 1.6

$$C_f = \frac{0.664}{\sqrt{Re_x}} \tag{1.12}$$

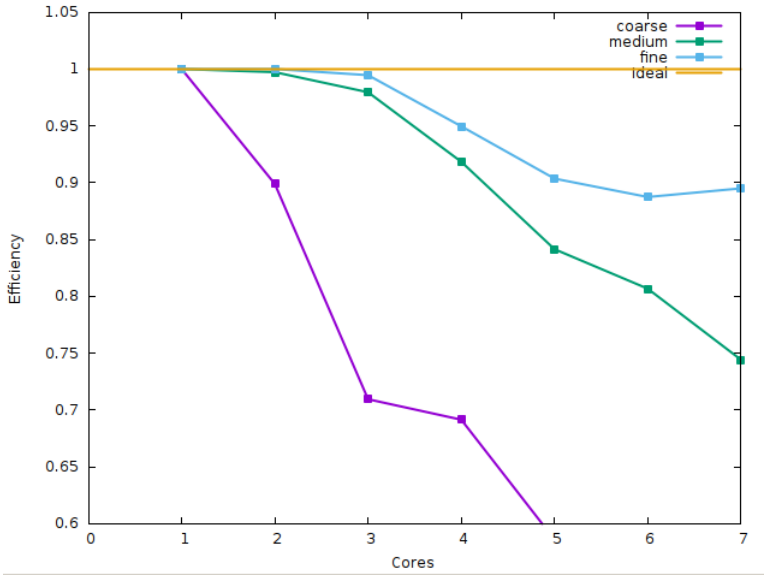


Σχήμα 1.6: *Ροή σε Επίπεδη Πλάκα. Καμπύλη συντελεστή τριβής και σύγκριση με τη λύση Blasius.*

Παρατηρείται ότι η αριθμητικη και η αναλυτική λύση είναι πολύ κοντά η μια στην άλλη, με μεγαλύτερη διαφορά μεταξύ αναλυτικής και αριθμητικής λύσης του συντελεστή τριβής να είναι 0.05. Στη συνέχεια, το λογισμικό παραλληλοποιείται και αξιολογείται η απόδοση και η επιτάχυνσή του. Η εφαρμογή έγινε πάλι για τη ροή γύρω από επίπεδη πλάκα χρησιμοποιώντας τρία διαφορετικά πλέγματα. Στα Σχήματα (1.7)-(1.8) φαίνονται τα αποτελέσματα μετά από επίλυση με 3 διαφορετικά πλέγματα που αποτελούνται από 36000, 70000 και 100000 κυψέλες αντίστοιχα.



Σχήμα 1.7: *Ροή σε Επίπεδη Πλάκα. Επιτάχυνση λογισμικού για διαφορετικό πλήθος επεξεργαστών και 3 διαφορετικά πλέγματα.*



Σχήμα 1.8: *Ροή σε Επίπεδη Πλάκα. Απόδοση λογισμικού για διαφορετικό πλήθος επεξεργαστών και 3 διαφορετικά πλέγματα.*

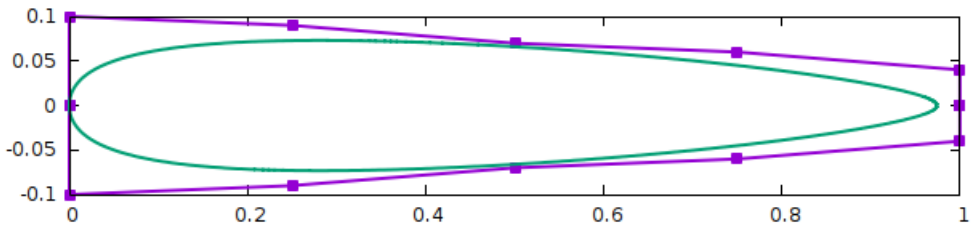
Παρατηρείται ότι με την αύξηση των επεξεργαστών υπάρχει απόκλιση από την ιδανική καμπύλη διότι αυξάνονται οι επικοινωνίες μεταξύ τους. Με την πύκνωση του πλέγματος επιτυγχάνεται καλύτερη προσέγγιση της ιδανικής καμπύλης, διότι πλέον το ποσοστό των επικοινωνιών είναι μικρότερο σε σχέση με αυτό των υπολογισμών που πρέπει να γίνουν.

1.3 Βελτιστοποίηση Μορφής με Χρήση της Συνεχούς Συζυγούς Μεθόδου

Στο τέλος της εργασίας αναπτύσσεται η συνεχής συζυγής μέθοδος για εφαρμογή σε αεροδυναμική βελτιστοποίση μορφής. Η συνάρτηση-κόστους του προβλήματος (άνωση) δίνεται από τη σχέση

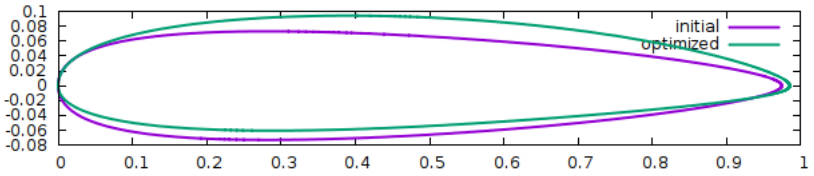
$$F = \int_{S_w} p n_k r_k dS \tag{1.13}$$

όπου S_W η επιφάνεια της αεροτομής, p η στατική πίεση, $\vec{r}$ το μοναδιαίο διάνυσμα στην κατεύθυνση της άνωσης και $\vec{n}$ το μοναδιαίο διάνυσμα κάθετο στην αεροτομή με κατεύθυνση προς τα έξω. Αφού επιλυθούν οι συζυγείς εξισώσεις, το συζυγές πεδίο χρησιμοποιείται για τον υπολογισμό των παραγώγων ευαισθησίας οι οποίες στη συνέχεια χρησιμοποιούνται στη βελτιστοποίηση με τη μέθοδο της απότομης καθόδου. Το λογισμικό βελτιστοποίησης εφαρμόστηκε σε 2 αεροτομές και εδώ θα παρουσιαστεί η μία από τις δύο. Η αεροτομή φαίνεται στο Σχήμα (1.9) και παραμετροποιείται με 17 σημεία $Bezier$. Η αεροτομή αποτελείται από μια συνεχόμενη καμπύλη.



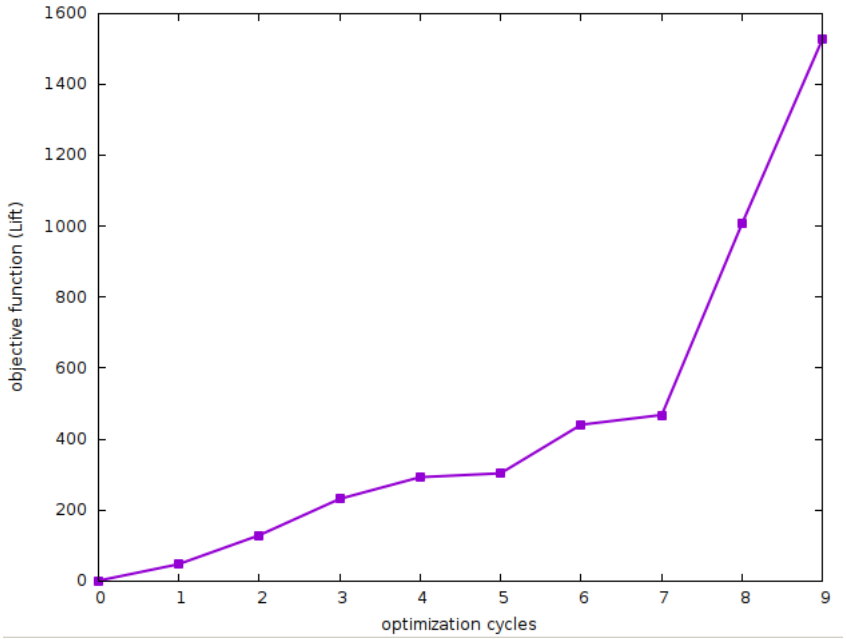
Σχήμα 1.9: *Βελτιστοποίηση Μορφής Αεροτομής. Αρχικό σχήμα αεροτομής μαζί με τα σημεία* $Bezier$.

Οι μεταβλητές σχεδιασμού είναι οι συντεταγμένες των σημείων $Bezier$. Μετά την επίλυση γίνεται σύγκριση της αρχικής και της τελικής αεροτομής στο Σχήμα 1.10



Σχήμα 1.10: *Βελτιστοποίηση Μορφής Αεροτομής. Σύγκριση αρχικής και βελτιστοποιημένης γεωμετρίας.*

Η αρχική γεωμετρία έχει μηδενική τιμή άνωσης $L_{init} = 0.0N$, ενώ η τελική $L_{opt} = 1529.01N$. Η πορεία της συνάρτησης-κόστους κατά τη βελτιστοποίηση φαίνεται στο Σχήμα (1.11).



Σχήμα 1.11: *Βελτιστοποίηση Μορφής Αεροτομής. Μεταβολή συνάρτησης-κόστους κατά τη βελτιστοποίηση.*

Επειδή η ροή είναι ατριβής, η άνωση θα συνεχίσει να αυξάνει όσο προχωρά η βελτιστοποίηση εκτός και αν επιβληθεί κάποιος περιορισμός. Έτσι, κριτήριο για τον τερματισμό του αλγορίθμου επιλέγεται να είναι οι 9 κύκλοι βελτιστοποίησης.

1.4 Συμπεράσματα

Τα βασικά συμπεράσματα που προέκυψαν από την έρευνα είναι τα εξής:

- Με την εισαγωγή ενός όρου πηγής στις εξισώσεις του πρωτεύοντος προβλήματος που η σύγκλισή του περιπίπτει σε ταλάντωση, οι ταλαντώσεις μπορούν να εξαλειφθούν και η λύση να συγκλίνει περισσότερο.
- Η σύγκλιση του συζυγούς προβλήματος εξαρτάται σε πολύ μεγάλο βαθμό από τη σύγκλιση του πρωτεύοντος προβλήματος. Στην περίπτωση που το δεύτερο έχει ταλαντώσεις, οι συζυγείς εξισώσεις μπορεί να αποκλίνουν.
- Η παραλληλοποίηση ενός κώδικα επιφέρει τεράστια εξοικονόμηση χρόνου για την επίλυση προβλημάτων υπολογιστικής ρευστοδυναμικής. Ειδικά όταν πρόκειται για μεγάλες εφαρμογές, η χρήση παράλληλου λογισμικού είναι απαραίτητη. Μετά από παραμετρική μελέτη παρατηρήθηκε ότι σε πυκνότερα πλέγματα, ένα παράλληλο λογισμικό είναι πιο αποδοτικό, δηλαδή μπορεί να επιφέρει μεγαλύτερη επιτάχυνση στην επίλυση. Αυτό οφείλεται στο γεγονός ότι το ποσοστό των κυψελών που πρέπει να επικοινωνήσουν είναι αισθητά μικρότερο από τους υπολογισμούς που πρέπει να κάνει κάθε επεξεργαστής.
- Η μέθοδος των ψευδο-κυψελών είναι εν γένει μια μέθοδος που δεν επιβάλλει με ακρίβεια τις οριακές συνθήκες στο όριο ενός στερεού. Παρ' όλα αυτά παρατηρήθηκαν πολύ καλά αποτελέσματα σε προβλήματα στρωτής ροής, κάνοντας σύγκριση με αναλυτικές λύσεις και με αποτελέσματα που προέκυψαν με χρήση οριόδετων πλεγμάτων.
- Η εργασία έβαλε βάσεις για περαιτέρω ανάλυση και έρευνα. Αρχικά στο λογισμικό HYDRA, ο συζυγής κώδικας συγκλίνει και παράγει αποτελέσματα σε περιπτώσεις που πριν ήταν αδύνατο. Από την άλλη μεριά, στη βιβλιογραφία υπάρχουν λίγες αναφορές στη χρήση της μεθόδου των ψευδο-κυψελών σε συζυγείς μεθόδους, ιδιαίτερα στη συνεχή συζυγή μέθοδο. Το λογισμικό που αναπτύχθηκε έδωσε πολύ καλά αποτελέσματα στα προβλήματα που εφαρμόστηκε και πλέον υπάρχει ένα ακόμα διαθέσιμο εργαλείο βελτιστοποίησης στη ΜΠΥΡΒ/ΕΜΠ.