12-2018

Towards Adaptive and Grid-Transparent Adjoint-Based Design Optimization Frameworks

Seyedreza Djeddi
University of Tennessee, djeddi@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Seyedreza Djeddi entitled "Towards Adaptive and Grid-Transparent Adjoint-Based Design Optimization Frameworks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mechanical Engineering.

Kivanc Ekici, Major Professor

We have read this dissertation and recommend its acceptance:

Vasilios Alexiades, Jay Frankel, Majid Keyhani

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Towards Adaptive and Grid-Transparent Adjoint-Based Design Optimization Frameworks

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Seyedreza Djeddi

December 2018

to my parents, Farrokh and Farah

to my sister, Sahar, and to my brother, Amir

and to my lovely wife, Roshanak, for her encouragement and support

in memory of my late grandfather

who always believed in me but did not get to see me graduate with a PhD

Acknowledgments

I am deeply indebted to my advisor, Dr. Kivanc Ekici, for his continuous support during my years at the University of Tennessee. Dr. Ekici provided me with every bit of assistance, patience, guidance, and expertise since my first day as a Ph.D. student. He gave me freedom and courage to branch out into new research areas, while continuously providing me with his valuable feedback, advice, encouraging words, and great ideas. Every time I stumbled upon an issue or a puzzling problem in my research, he provided me with not one but many suggestions and ideas to help me tackle those challenges while at the same time helping me become a better problem solver. The enthusiasm he has for CFD has been contagious and his depth of knowledge is awe-inspiring. In addition to his academic support, Dr. Ekici has been a great role model, a caring mentor, and has been tremendously helpful in preparing me for my career. He is without a doubt the most brilliant professor and kind-hearted advisor I could ever ask for.

I would like to thank the rest of my doctoral committee members for their valuable comments and suggestions. Special thanks go to Professor Majid Keyhani and Professor Vasilios Alexiades for their insightful courses on numerical heat transfer, conservation laws, and parallel computing. These courses not only provided me with the fundamental knowledge to carry out my Ph.D. research, but they also taught me how to tackle scientific problems. I would also like to express my gratitude to Professor Jay Frankel for reviewing my dissertation and for his extremely valuable feedback, comments, and encouraging words.

I am also grateful for the financial support that I have received throughout my Ph.D. studies from the National Science Foundation (under grant No: CBET-1150332), Oak Ridge National Laboratory, the R.N. Lyons Engineering Graduate Fellowship, James A. Euler

Scholarship, and William S. Johnston Scholarship, as well as the Department of Mechanical, Aerospace, and Biomedical Engineering.

I would like to thank my colleague, Dr. Jason Howison, for his valuable suggestions, encouragement, guidance, and support. I also want to thank fellow graduate students and lab mates, Mr. Andrew Kaminsky and Mr. Hang Li, for their suggestions, support and encouragement even when I stumbled upon problems in my research. I cherish the times we spent together, even the "buggy" ones!

I owe a great debt of gratitude to my Dad, Farrokh, and to my Mom, Farah, for supporting me through all these years in any way possible. They have provided me with unconditional love, opportunity, and constant encouragement throughout the years and have never given up on me. I hope that I have at least made them proud or even simply happy as I can never thank them enough for what they have done for me.

I would like to acknowledge my sister, Sahar, and my brother, Amir, who always believed in me even when I doubted myself. They always tried to encourage me to work harder and do better in every step of the way. I feel fortunate to have them in my life and I am proud to call them my siblings.

Last but not least, I would like to thank my lovely wife, Roshanak, for constantly reminding me of all the beautiful things in life and the true meaning of love. She is simply the most loving, supportive, encouraging, and patient wife I could ever ask for. Roshanak has made countless sacrifices to help me get to where I am today. None of this would have been possible without the love and support that I have received from her. My love, Thank you for being my voice of reason and my heart of the matter.

Abstract

With the growing environmental consciousness, the global perspective in energy production is shifting towards renewable resources. As recently reported by the Office of Energy Efficiency & Renewable Energy at the U.S. Department of Energy, wind-generated electricity is the least expensive form of renewable power and is becoming one of the cheapest forms of electricity from any source. The aeromechanical design of wind turbines is a complex and multidisciplinary task which necessitates a high-fidelity flow solver as well as efficient design optimization tools. With the advances in computer technologies, Computational Fluid Dynamics (CFD) has established its role as a high-fidelity tool for aerodynamic design.

In this dissertation, a grid-transparent unstructured two- and three-dimensional compressible Reynolds-Averaged Navier-Stokes (RANS) solver, named UNPAC, is developed. This solver is enhanced with an algebraic transition model that has proven to offer accurate flow separation and reattachment predictions for the transitional flows. For the unsteady time-periodic flows, a harmonic balance (HB) method is incorporated that couples the sub-time level solutions over a single period via a pseudo-spectral operator. Convergence to the steady-state solution is accelerated using a novel reduced-order-model (ROM) approach that can offer significant reductions in the number of iterations as well as CPU times for the explicit solver. The unstructured grid is adapted in both steady and HB cases using an r-adaptive mesh redistribution (AMR) technique that can efficiently cluster nodes around regions of large flow gradients.

Additionally, a novel toolbox for sensitivity analysis based on the discrete adjoint method is developed in this work. The **F**ast automatic **D**ifferentiation using **O**perator-overloading **T**echnique (FDOT) toolbox uses an iterative process to evaluate the sensitivities of the cost function with respect to the entire design space and requires only minimal modifications to

the available solver. The FDOT toolbox is coupled with the UNPAC solver to offer fast and accurate gradient information. Ultimately, a wrapper program for the design optimization framework, UNPAC-DOF, has been developed. The nominal and adjoint flow solutions are directly incorporated into a gradient-based design optimization algorithm with the goal of improving designs in terms of minimized drag or maximized efficiency.

Table of Contents

List of Tables

List of Figures

Nomenclature

Acronyms

AD Automatic Differentiation

AGARD Advisory Group for Aerospace Research and Development

ALE Arbitrary Lagrangian-Eulerian

AMR Adaptive Mesh Refinement/Redistribution

BC Baş and Çakmakçıoğlu

CFD Computational Fluid Dynamics

CPU Central Processing Unit

CV Control Volume

DNS Direct Numerical Simulations

DOF Design Optimization Framework or Degree-of-Freedom

FDOT Fast Automatic Differentiation using Operator Overloading Technique

FVM Finite Volume Method

GCL Geometric Conservation Law

GWEC Global Wind Energy Council

HAWT Horizontal Axis Wind Turbine

HB Harmonic Balance

L-BFGS-B Limited-memory Broyden-Fletcher-Goldfarb-Shanno Bounded

LES Large-Eddy Simulations

NACA National Advisory Committee for Aeronautics

NLFD Non-Linear Frequency Domain

NREL National Renewable Energy Laboratory

ONERA Office National d'Etudes et de Recherches Aérospatiales

OO Operator Overloading

POD Proper Orthogonal Decomposition

RAE Royal Aircraft Establishment

RAM Random Access Memory

RANS Reynolds-Averaged Navier-Stokes

RBF Radial Basis Function

ROM Reduced-Order Model

SA Spalart-Allmaras

SCT Source Code Transformation

SGS Sub-Grid Scale

UNPAC Unstructured Parallel Compressible

Greek Symbols

α Angle of attack or the optimizer step-size

χ Turbulent viscosity to laminar viscosity ratio

δ_{ij}	Kronecker delta
ϵ	Smoothing coefficient or Machine Zero
γ	Gas constant
γ_{trans}	Laminar-Turbulent transition intermittency function
Λ	Spectral radii
λ	Mean free path or second viscosity coefficient
$\mathcal{V}$	Control volume
μ	Dynamic viscosity
μ_L	Laminar viscosity
μ_T	Turbulent (eddy) viscosity
ν_L	Laminar kinematic viscosity
Ω	Vorticity magnitude or rotational frequency or fundamental frequency of excitation
ω	Frequency
$\overline{\overline{\tau}}$	Viscous stress tensor
$\partial\mathcal{V}$	Control surface
ϕ	Limiter function
ρ	Density
τ	Pseudo-time step
$\tilde{\nu}$	Spalart-Allmaras turbulence model working variable
$\tilde{\nu}_{\text{cr}}$	Critical eddy viscosity
$\vec{\nabla}$	Gradient vector

$\vec{\omega}$ Angular velocity vector

$\vec{\psi}$ Adjoint vector

$\vec{\Theta}$ Work of viscous stresses and heat conduction

$\vec{\xi}$ Expansion coefficients

Roman Symbols

$\mathbf{A}$ Jacobian of the flux vector

$\mathbf{D}$ Pseudo-spectral operator

$\mathbf{E}$ Discrete Fourier transformation matrix

$\mathbf{E}^{-1}$ Inverse discrete Fourier transformation matrix

$\mathbf{H}$ Hessian matrix

Kn Knudsen number

Pr Prandtl number

Re Reynolds number

Re_ν Vorticity Reynolds number

Tu_∞ Free-stream turbulence intensity

$\tilde{K}$ Averaged turbulent kinetic energy

$\tilde{S}_{ij}$ Averaged strain rate

$\vec{F}$ Force vector

$\vec{f}_{\mathrm{cent}}$ Centrifugal force

$\vec{f}_{\mathrm{Cor}}$ Coriolis acceleration force

$\vec{F}_C$ Convective flux

$\vec{F}_D$	Diffusive flux
$\vec{f}_e$	Body (external) force vector
$\vec{n}$	Normal vector
$\vec{R}$	Residual vector
$\vec{r}$	Radial position vector
$\vec{r}_{ij}$	Vector connecting node i to node j
$\vec{S}$	Surface normal vector
$\vec{S}_{ij}$	Metric of edge ij
$\vec{U}$	Vector of conservation variables
$\vec{v}$	Velocity vector
$\vec{v}_{\text{abs}}$	Absolute velocity vector
$\vec{v}_{\text{grid}}$	Grid velocity vector
$\vec{v}_{\text{rel}}$	Relative velocity vector
$\vec{v}_{\text{rot}}$	Rotational velocity vector
c	Speed of sound
c_p	Specific heat coefficient at constant pressure
c_v	Specific heat coefficient at constant volume
d	Wall distance
dS	Surface element
E	Total energy per unit mass of a fluid
e	Internal energy

H Total enthalpy

h Specific enthalpy

I Rothalpy or Objective function

K Spring stiffness

k Thermal conductivity coefficient or spring stiffness (AMR) or reduced frequency, $k = \omega c/2U_\infty$

L Characteristic length

L_{ij} Length of edge ij

M Mach number

N_H Number of harmonics retained in the Fourier series

N_T Total number of sub-time levels ($N_T = 2N_H + 1$)

p Static pressure

Q_V Volumetric source term

R Specific gas constant

T Temperature

t Physical time

V Contravariant velocity ($\vec{v} \cdot \vec{n}$)

V_r Contravariant relative velocity

$x,\ y,\ z$ Cartesian coordinates

Superscripts

$'$ Fluctuating part

$\bar{}$	Mean part or adjoint variable or non-dimensional values
$\tilde{}$	Roe averages
n	Values at physical or pseudo time level n
T	Transpose

Subscripts

∞	Free-stream values
ref	Reference values
i	Node i
ij	Edge connecting node i to node j
$R,\ L$	Right and left states (Roe fluxes)

Chapter 1

Introduction

1.1 Motivation

In recent decades, renewable energy started playing a more important role in energy policies in most of the developed countries. While the US is second among countries in terms of the amount of energy produced from sustainable sources, renewables constitute about 20% of the total energy produced [212]. Among other renewable energy resources, wind energy has the second highest share after hydropower, as shown in Figure 1.1.

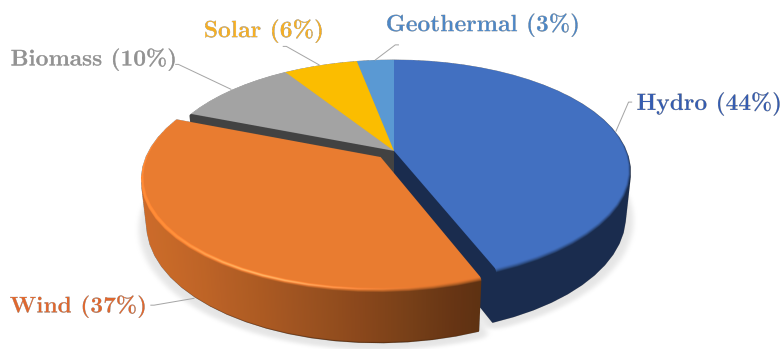


Figure 1.1: Renewable energy consumption by source (data from the U.S. Energy Information Administration [212]).

The increased interest in wind energy among all the other renewable energy resources can be related to a few different factors. Current data shows that there is a steady growth in wind energy capacity. In a report published by the Global Wind Energy Council (GWEC), it

is indicated that with the current and some conservative new policies, it is possible to see the wind energy capacity getting tripled by the year 2050 [68]. However, this report also shows that by investing in advanced technologies, it is possible to achieve an eight-fold increase in wind energy capacity in the same amount of time [68] (see Figure 1.2). On the other hand, a 2012 study by the National Renewable Energy Laboratory (NREL) reports that the levelized cost of the wind power can go down by about 25% in the next 10-15 years [130].

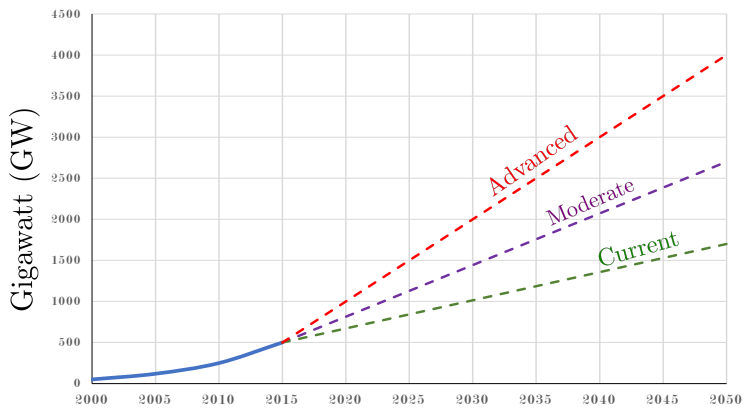


Figure 1.2: Wind energy capacity trend and forecast (data from Global Wind Energy Council [68]).

Unfortunately, the increase in wind energy capacity can be limited due to cost considerations as most of the current capacity improvement efforts are focused on increasing the size of the blades which can exponentially increase the production costs. There has been a steady increase in the size of the modern commercial wind turbines with the most recent multi-MW horizontal-axis wind turbines (HAWT) having a rotor diameter of about 140 meters. With the goal of reducing the cost of electric power, this trend is forecast to continue. However, the aerodynamic efficiency of the turbine can be improved as an alternative for harnessing more energy from the wind. In fact, by designing optimized wind turbine blades, the efficiency and capacity can be increased without increasing the size of the wind turbines. While this is the main goal here, the development of an advanced and robust design optimization framework can have broader impacts. As an example, techniques developed in this work can be readily extended for designing aircraft wings and fuselage that are much more efficient than what is currently available, which can be translated into billions of dollars of savings in fuel costs as well as significant reductions in carbon footprint.

The aerodynamic shape optimization of a wind turbine involves the determination of an optimized blade topology that satisfies certain objectives subject to a set of aerodynamic and/or structural constraints. Traditionally, "inverse" design methods have been used with the goal of obtaining an optimal shape considering a prescribed target distribution of aerodynamic quantities. More recently, design optimization techniques have shifted towards more "direct" methods based on searching design space for optimum topologies. This approach may lead to novel unconventional designs subject to challenging off-design conditions.

As wind turbines continue to gain more popularity and take on a greater role in energy production, their design has also become a more complex and multidisciplinary task involving rather conflicting requirements such as increased performance, reduced acoustic signature, and reduced blade loads. This requires advanced modeling tools, such as Computational Fluid Dynamics (CFD), that can cover diverse aspects of the flow features around the turbine blades while building a platform for design optimization to further enhance the state-of-the-art. Currently, most designs rely on low-fidelity or semi-empirical computational tools such as Blade Element Momentum (BEM) theory [78]. These methods are fast computational tools but they also rely on the existence and accuracy of the available airfoil data. On the other hand, Navier-Stokes solvers which have been mostly avoided in the past due to their high computational requirements, are becoming widely used with the increasing computational power available. Nonetheless, time-accurate transient solutions of HAWT flows can take a significant amount of time even with the aid of high performance computing resources.

Due to the fact that most of the fundamental wind turbine unsteady problems can be viewed as periodic, frequency-domain techniques can be utilized to avoid long wall-clock times and prohibitively costly computational effort. The **high-dimensional harmonic balance (HDHB) method** [85], in particular, takes advantage of the temporal periodicity to convert the solution of the flowfield from an unsteady time-accurate approach into a mathematically-steady time-frequency-coupled approach. Although the use of the HDHB method can greatly reduce the computational cost, the overall efficiency of CFD modeling used in the design optimization frameworks can be further enhanced in several ways.

Since CFD solvers involve iterative numerical schemes, their convergence behavior can be improved by novel **acceleration** techniques. On the other hand, any CFD-based simulation and modeling requires extensive amount (well over 30%) of time spent on the generation of the computational grid (mesh). In fact, the traditional CFD simulations require a continuous improvement of the mesh which relies heavily on the experience of the design engineers. Therefore, a **solution-based grid adaptation** technique can be greatly advantageous, especially during the design stage, as unconventional geometries and off-design conditions become involved.

In general, there are two main families of aerodynamic optimization techniques that can be categorized as (1) gradient-based and (2) non-gradient-based approaches, where in the latter, repeated cost function evaluations are required. Using a CFD-based design approach, the computational burden of evaluating the objective (cost) function can be very high even for today's high performance computing resources. Therefore, it is desirable to use gradient-based algorithms in the framework of aerodynamic design optimization. In this approach, however, derivatives (sensitivities) of a cost function with respect to all design variables (that can number in the hundreds to thousands) are required.

While "optimum" configurations can be determined after a small number of optimization cycles using the gradient-based approach, the cost and complexity associated with the gradient evaluations are overwhelming. These issues are even more pronounced in high-fidelity CFD solvers and in the framework of a multidisciplinary design optimization process. Therefore, a **fast and efficient sensitivity analysis tool** would be a leap forward that can greatly reduce the time and cost of the CFD-based design and topology optimization. Such an advanced and effective tool would result in the development of a **robust optimization platform** for not only wind turbine designs but any aerodynamic or aerostructural design problem involving aircraft and rotorcraft, to name a few.

Last but not least, the development of CFD solvers is a demanding task involving an extensive amount of man-hour implementing various numerical techniques. With the steady increase in the amount of flow features that are required to be captured in a numerical flow solver, high-order methods are becoming popular in the CFD community. Additionally, unconventional grid topologies are being considered in order to ease up the grid generation

process for complex geometries. Implementing new high-order methods as well as the non-standard element definitions in an in-house or a commercial CFD solver requires a great deal of additional code development. Therefore, a **grid-transparent** solution process, that treats different grid topologies identically, would result in significant time-savings for advancing the available modeling tools while improving the flexibility of the CFD solvers for future developments.

1.2 Background

Advances in airfoil analysis, rotor development, material technology and stress analysis have made wind turbines suitable alternatives to fossil fuels in terms of cost incurred per energy unit [228]. The aerodynamic effects on the wind turbines are mostly known but the details of the flow are still not well-understood. On the other hand, in order to improve the wind turbine aerodynamic and aerostructural characteristics, design optimization tools have been developed and various efforts have been made to address the issues associated with each tool. As far as the CFD solvers and their efficiency and numerical capabilities are concerned, efforts have been made in developing convergence acceleration as well as grid adaptation techniques. Overall, there have been numerous studies carried out in these different areas over a long period of time. This section aims to highlight the previous efforts done in the field concerning this dissertation.

1.2.1 Wind Turbine Modeling and Design

The size of commercial wind turbines has increased dramatically during the last 25 years. This development has forced the design tools to evolve from simple static calculations (with the assumption of constant wind speed) to dynamic simulation techniques that can predict not only the unsteady aerodynamic loads, but also the aeroelastic response of the entire wind turbine.

Different methods with various levels of complexity are used to calculate the aerodynamic loads on a wind turbine rotor. Boundary Element Momentum (BEM) method is the most common tool for calculating the aerodynamic loads on wind turbine rotors since it is

computationally cheap. Furthermore, BEM provides satisfactory results provided that good airfoil data are available for lift and drag coefficients as a function of the angle of attack and the Reynolds number. The method was introduced by Glauert [78] as a combination of one-dimensional momentum theory and blade element consideration to determine the loads locally along the blade span. This method is proven to be successful but, as stated before, it depends solely on reliable airfoil data for different blade sections. Additionally, for airfoils subjected to temporal variations of the angle of attack, the dynamic response changes the static airfoil data and dynamic stall models need to be included [89].

The first applications of Computational Fluid Dynamics (CFD) to wings and rotor configurations date back to late seventies and early eighties in connection with airplane wings and helicopter rotors [7, 24, 178, 170, 37] using potential flow solvers. To overcome some of the limitations of potential flow solvers, a shift towards unsteady Euler solvers was seen through the eighties [179, 115, 169, 3]. When computational resources increased, the solution of the full Reynolds-Averaged Navier-Stokes (RANS) equations including the viscous effects were obtained in the late eighties and early nineties [195, 196, 19]. As a result, application of these simulation tools to flows over wind turbine blades became of practical interest. Subsequently, full Navier-Stokes computations of rotor aerodynamics were carried out [88, 191, 232, 57, 192].

For the CFD approaches in wind turbine flow applications, the computational cost can easily become very high for large scale problems, such as unsteady wind turbine computations. Furthermore, the traditional time-accurate method is not suitable in an adjoint optimization platform, which is pursued in this study. This is due to the fact that adjoint optimization requires storage of all intermediate flow variables that can make it extremely expensive. Considering that most of the unsteady problems of interest in wind turbine applications are periodic in time, the frequency domain techniques can be suitable alternatives to these time-accurate methods. In the time-linearized methods [84, 42], the unsteadiness is assumed to be harmonic in time and the deviations are small comparing to the mean flow. Thus, the nonlinear unsteady governing equations can be decoupled into a set of nonlinear steady and linear unsteady equations which would then be solved in sequence one after the other. This way, the computational cost would be approximately three times

that of the steady computation [106]. However, the nonlinear effects cannot be resolved due to the fundamental assumption of small perturbations.

In order to be able to capture the nonlinear effects, Hall et al. [85] applied the classical frequency domain technique of harmonic balance to Euler equations. In this approach, they represented the flow by a Fourier series in time and the Fourier coefficients were defined by conservative flow variables. The harmonic balance (HB) method is capable of handling both linear and nonlinear unsteady problems and for linear unsteady problems, the HB solutions are identical to those of the time-linearized method [106]. While the classical HB method can be applied to Euler equations in a straightforward fashion, the method does not work well for the RANS equations with complex turbulence models. With the aim of circumventing this problem, Hall et al. [85] introduced the high-dimensional harmonic balance (HDHB) method in which the conservative variables were computed and subsequently stored at equally-spaced sub-time levels over the span of a single period. These solutions would then be coupled using a pseudo-spectral operator which approximates the physical time derivative in the governing equations [85]. Moreover, the entire numerical solution would become mathematically steady-state which makes it possible for the convergence acceleration techniques such as local time-stepping and residual smoothing to be utilized in order to further enhance the computational speed. Time-periodic flows about wind turbines that are driven by the rotation are special in the sense that the fundamental frequency is predefined by an external forcing function, i.e., the rotation of the turbine in the case of wind turbine flows. This characteristic along with the many benefits of the HDHB method, have made it a perfect choice in conjunction with RANS solvers to tackle the complex wind turbine flow problems.

It is worth mentioning that all of the previous efforts have been limited to flows with time-periodicity while a number of interesting physical flows are *aperiodic*. As an example, a turbine may be subjected to wake excitation at one frequency while the blades are vibrating at another frequency. If the ratio of these excitation frequencies is irrational, then the flow will be called *aperiodic*. As originally proposed and demonstrated by Ekici and Hall [60], it is possible to use the harmonic balance method to tackle unsteady aperiodic flows. In the framework of wind turbine flows and in the recent years, Campobasso and

Baba-Ahmadi [35] have implemented a harmonic balance compressible RANS solver with low-speed preconditioning for wind turbine unsteady aerodynamics. They have shown that using a harmonic balance solver, periodic wind turbine flows can be simulated more than 10 times faster than time-domain solvers. Recently, Howison and Ekici [102] have used a robust HB-RANS solver to study the unsteady aerodynamics of a pitching S809 wind turbine airfoil using the one-equation Spalart-Allmaras turbulence model and low-speed preconditioning. More recently, Howison [105] has shown that the coupling a transition model with HB-RANS solvers can further increase the fidelity of the numerical results for cases involving highly separated flows.

Despite various efforts that have been made for the design optimization of wind turbines, achieving the optimal design of a wind energy system is yet to be realized. Due to the multidisciplinary nature of the physics that the wind turbines experience as well as the stochastic nature of the wind, the design optimization problem is complex and difficult to tackle [82].

Nevertheless, various design methods have been made available in the literature that can be readily applied to the aerodynamic shape optimization of the rotor blades [78, 99]. With the advancements in CFD simulations, multidisciplinary design optimization techniques have gained popularity [70, 181, 69, 95]. Development of the Eppler code [65] enabled the design of new airfoil sections, which ultimately lead to the design of new HAWT turbines. This code uses a prescribed velocity distribution as well as a conformal mapping technique to solve for the potential flow around the airfoils using a panel method. Due to its inverse design capabilities [176], the Eppler code has been favored among other airfoil design codes available [87].

In the years that followed, BEM methods have become the driving force for the wind turbine design and many efforts have been made to maximize the annual energy production by optimizing airfoil geometries as well as blade twist [74, 131, 223]. With the Combined Experiment Rotor (CER) project [30, 158] being commissioned at the National Renewable Energy Laboratory (NREL), the Phase VI blade [73] was introduced. In the process of developing this new blade, several codes and software packages have been developed and utilized. These include a BEM-based flow solver named PROP [96], an inverse design method

code named PROPID [72, 182], and finally a blade geometry optimization toolbox based on the genetic algorithm named PROPGA [181]. These tools have been used collectively ever since for the design optimization of horizontal-axis wind turbines [41, 94].

With the pioneering works of Pironneau [167] and Jameson [111] and the introduction of the "adjoint method" for aerodynamic shape optimization, focus has been recently shifted towards CFD-based continuous and discrete adjoint methods for the design optimization in aerospace and wind turbine applications. These efforts are discussed in more detail in the next section.

1.2.2 Adjoint-Based Sensitivity Analysis

In the past several decades, efficient gradient-based aerodynamic optimization methods have been developed. Traditionally, the gradient information has been computed using the finite difference method. However, these approximations not only suffer from truncation and cancellation errors, but they also require evaluation of the objective function – a fully converged CFD solution – for each perturbed input during each design cycle. Therefore, the computational cost using this approach grows linearly with the number of design variables which can be computationally expensive.

As an alternative to the finite difference approach, Pironneau [167] introduced the adjoint method to fluid dynamics problems which was later extended to aerodynamic design of three-dimensional wings by Jameson [111] using the continuous approach. A few years later, Elliot and Peraire [64] presented the discrete method that allowed investigators to perform large-scale, multi-point/multi-disciplinary optimization studies for aircraft design. The main advantage of adjoint methods is their computational efficiency since the cost of gradient evaluation is independent of the number of design variables in both continuous [6, 124] and discrete [163, 75] approaches. In the continuous approach, the governing flow equations are linearized first and then discretized which makes it not only computationally- but also memory-efficient. However, development of a continuous adjoint solver normally requires even more effort than the nominal CFD solver. Especially, for complex flow configurations involving turbulence and transition, code development becomes extremely difficult and cumbersome. The issue becomes even more pronounced because of the difficulties that

arise in developing proper boundary conditions. In contrast, the discrete adjoint methods are based on the linearization of the discretized form of the governing flow equations, i.e., the flow solver. In this approach, the inclusion of turbulence and transition models and the treatment of boundary conditions is greatly simplified. A more in-depth comparison of the continuous and discrete adjoint approaches and their advantages and disadvantages can be found in a paper authored by Giles and Pierce [76]. To this day, the preference of one approach over the other is still largely debated.

In the framework of discrete adjoint methods, algorithmic or automatic differentiation (AD) tools can be used to substantially simplify the development of the complementary adjoint solver. Automatic differentiation is a non-approximative method similar to symbolic differentiation that is based on the systematic application of the chain rule of differentiation. The AD method allows fast and highly accurate evaluation of derivatives that are exact up to machine precision with no round-off or truncation errors. The AD algorithm can be carried out in forward or reverse modes based on the direction in which the derivatives are being propagated. The forward mode is very easy to implement as it simply propagates the derivatives along the expression tree. Therefore, after each forward solution, the sensitivity of the objective function with respect to one input variable is obtained. Apparently, the process has to be repeated for each individual design variable, thus increasing the computational cost linearly with the number of design variables. The reverse mode of AD, on the other hand, has the advantage of having much improved efficiency since the computational cost is independent of the number of input variables. However, this requires the reversal of the expression tree of the nominal solver as well as the reversal of the propagation of gradient information which makes it more challenging to implement.

As far as programming is concerned, AD can be performed using two different approaches: (1) source code transformation (SCT) and (2) operator overloading (OO). Using the former approach, a preprocessor reads in the computer code (nominal solver) and parses it by applying differentiation rules to each expression and arithmetic operation and finally generates a new source code (adjoined) that can be compiled and run to calculate the derivatives. The SCT approach is popular among researchers since the resulting adjoint code can be aggressively optimized to achieve fast run-times. However, since the code has to

be parsed, adjoined, and extensively debugged after each minor modification to the primal code such as changing the cost function or the design variables, investigators continue to seek other alternatives. Currently, there are many SCT tools developed for C/C++ and Fortran programming languages. Some examples include OpenAD [165], TAPENADE [91], TAF [71], and ADIFOR [20]. In comparison, the operator overloading approach is a direct result of the language capabilities of object-oriented programming. In the OO approach, new derived types or classes are used for storing the values of each variable together with an index for that variable in the expression tree. The entire expression tree is then recorded in another class called the *tape* that stores the operation type, the indices of the input arguments, the resulting value after the operation and the resulting partial derivative (adjoint) for each individual entry. There are many OO/AD tools such as ADOL-C [80], Adept [100], CppAD [17] and more recently CoDiPack [4] for C/C++ programs. As the use of object-oriented programming paradigms has gained popularity in Fortran programming, many OO/AD tools have been developed specifically for Fortran. ADF [199], ADOL-F [183], AUTO_DERIV [197], DNAD [234] and more recently dco/fortran [164] are some examples of these tools that are available today. The main challenge in using existing OO/AD tools is that the memory requirements for recording the tape can easily exceed the available resources as the expression tree evolves. This issue becomes even more prominent for explicit iterative schemes where a relatively slow convergence rate requires many iterations of the primal solver to reach a fully converged solution. To the best of the author's knowledge, none of the existing OO/AD packages have so far addressed issues with inherently large memory footprint.

1.2.3 Solution-Based Grid Adaptation[1]

In general, there are three different classes of adaptive mesh schemes. These can be categorized as (1) h-adaptive, (2) p-adaptive, and (3) r-adaptive techniques. The first and the most widely used is h-adaptive also known as "Adaptive Mesh Refinement" in which

[1]This section, in part, is a reprint of the material as it appears in AIAA Paper 2018-3245 titled "*An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers*" (2018). Authors: **Reza Djeddi** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi** and Kivanc Ekici.

the grid cells are locally refined with the addition of new grid nodes. A similar approach can also lead to coarsening by removing unnecessary grid nodes. The local refinement or coarsening can be triggered using a coloring scheme, which mostly depends on the gradient information of a specific flow variable. The second class, which is closely related to the h-adaptive technique and that is generally categorized under the same family of mesh refinement schemes, is called the p-adaptive refinement. This approach is widely used in the framework of finite element (FE) techniques where the order of elements is adjusted locally. Once again a similar coloring scheme can mark the elements for refinement where a higher order shape function is used to increase the accuracy of the FE approximation. While both of these approaches provide a lot of flexibility and have been extensively improved over the years, they suffer from some major disadvantages. First of all, h-adaptive techniques require complex data structures with evolving nodal/elemental connectivity information that can create significant implementation challenges. Moreover, the mesh refinement technique is completely localized that can lead to irregular and highly stretched elements with poor global structures [27].

The third class of adaptive mesh techniques known as r-adaptive is one of the the focal points of this dissertation. Also known as "Adaptive Mesh Redistribution," which is referred to as AMR in this work, the r-adaptive approach in general is a moving mesh technique. In this approach, the grid nodes are redistributed or relocated based on a forcing field while keeping the number of nodes and the nodal connectivities unchanged. Although the r-adaptive approach is not as mature as the other two adaptive mesh techniques, it provides certain advantages that can make this technique more attractable. A very important feature of the r-adaptive technique is the fact that the data structure remains constant throughout the adaptive redistribution process which can make the implementation of the technique straightforward in any numerical solver. This feature becomes particularly advantageous for harmonic balance/time-spectral solvers where it is desirable to have exactly the same number of grid nodes/cells for each sub-time level grid. That means, using the r-adaptive approach the nodal connectivity is preserved for grids at different sub-time levels. Furthermore, solution interpolation, which introduces additional errors is also avoided.

The goal of the r-refinement approach like any other adaptive mesh technique is to reduce the discretization errors by modifying the grid topology. However, as stated earlier, unlike the h-adaptive method, the r-adaptive technique keeps the number of grid nodes and their connectivities constant but relocates and clusters them around the regions where the gradients and/or curvatures of a certain flow variable are high. It is worth noting that in CFD calculations, the large discretization errors can be typically associated with the high gradients and high curvature regions. Therefore, having smaller cells in these regions can lead to significant accuracy improvements without the need to refine the entire grid. This principal idea has been originally exploited by de Boor [49] via efforts to achieve an optimal mesh that can guarantee an equi-distribution of the discretization error throughout the computational domain.

Due to the fact that the node relocation has been extensively studied for the moving mesh applications, the r-adaptive AMR approach is often referred to as a "Moving Mesh Method (MMM)". In fact, a very attractive feature of the r-refinement technique is its dual use in mesh deformation applications as well as minimization of discretization errors. Tang [204], and Budd et al. [28] provide a more-in-depth review of the moving mesh techniques and their direct application in r-adaptive AMR problems.

Generally speaking, the r-adaptive techniques are less mature compared to the h-adaptive and p-adaptive approaches. In fact, the r-adaptive AMR technique can sometimes lead to mesh entanglement with negative volumes or collapsed cells being formed in the computational domain. Moreover, cells with poor quality can be introduced in some cases during the post adaptation process. These drawbacks led to the introduction of truss networks based on the "spring analogy" to reduce the chances of mesh entanglement. This idea was initially used by Batina [15] for mesh deformation in unsteady aerodynamics applications, and was later improved by Blom [22] to include springs with zero equilibrium length on each grid edge. Farhat et al. [67] introduced the idea of torsional springs placed at each grid node with the torsional stiffness being defined based on the angle between the two connecting edges. Originally developed for two-dimensional triangular cells, this approach was later extended to three-dimensional problems with tetrahedral elements [50, 29]. The use of torsional springs has shown to improve the quality of the deformed grid at the price of

increased computational demand. This issue was successfully addressed by Blom [22] using a semi-torsional approach where the opposing angle for each linear spring is used to define a torsional stiffness. However, all of these approaches were only applicable to triangular (in two-dimensions) or tetrahedal (in three-dimensions) elements [235].

Since the structural stability of the truss structures is only valid for triangular geometries, the standard system needs to be enhanced to include the variety of cell types in a hybrid and grid-transparent CFD solver. In this regard, Bottasso et al. [25], and Acikgoz and Bottasso [1] introduced a new approach called "ball-vertex," which tries to remedy these limitations. In the ball-vertex method, a new linear spring is added between each grid node and its opposing face (virtual diagonal edges in quadrilateral cells or virtual opposing faces connecting the neighboring nodes in prism, pyramid and hexahedral elements). It is worth noting that if the node of interest passes through the opposing face, the element would be tangled up which results in cell inversion. Therefore, the additional linear spring can efficiently avoid mesh entanglement and compared to the torsional spring method, the ball-vertex approach can offer higher quality cells with lower chance of entanglement even for large deformations [25, 137].

It must also be pointed out that most of the work in the literature based on the spring analogy and ball-vertex approaches are focused on mesh deformation and dynamic mesh motion for fluid-structure interaction (FSI) applications. However, the general idea is directly applicable to r-adaptive AMR techniques with the driving force being defined not based on the boundary movements but the gradients/curvatures of a certain flow variable [151, 119] or an error measure [23]. Additionally, some of the work in the literature is limited to r-adaptive AMR for structured grids. These methods mostly rely on error minimization, variational adaptation [26] or center of mass [126, 18] with a good review of error-based r-adaptive techniques presented by Tyson et al. [211].

As discussed earlier, despite the fact that the r-adaptive AMR techniques offer significant advantages compared to the h-adaptive methods, this class of AMR is still not widely adopted by the scientific community. Moreover, the use of r-adaptive techniques in unsteady aerodynamics applications is generally missing in the literature. In this work, the adaptive mesh redistribution (AMR) technique based on the ball-vertex method and spring analogy

is applied to unsteady periodic flow cases. The harmonic balance (HB) method that was originally introduced by Hall et al. [85] is used to cast the time-periodic unsteady flow equations into a set of mathematically-steady equations governing the fluid flow at equally-spaced sub-time levels over a single period. These equations are then coupled together using a source term that is basically the approximation to the time-derivative of the conservation variables defined based on a pseudo-spectral operator [51]. The use of the HB technique has proven to substantially reduce the computational cost of modeling periodic and aperiodic fluid flows in turbomachinery and wind energy applications [63, 107, 109, 103, 104, 102, 53]. To the best of the author's knowledge, this is the first work in which the r-adaptive AMR has been used in conjunction with the harmonic balance technique for unsteady aerodynamic applications involving time-periodic flows.

1.2.4 ROM-Based Convergence Acceleration[2]

While there are many classical approaches that are primarily used for accelerating the convergence to steady state such as local time stepping, implicit residual smoothing [115] and multigrid [110], there has also been a great interest in the estimation and minimization of convergence errors in the framework of iterative solvers. Almost all of these efforts are based on the fundamental assumption of a linearly convergent iterative procedure. These methods are categorized into two main families: (1) epsilon algorithms and (2) polynomial methods. The epsilon algorithms are the oldest and can be based on scalar or vector sequences. Both scalar epsilon algorithm (SEA) and vector epsilon algorithm (VEA) were introduced by Wynn [231], and they transform a slowly convergent or even divergent sequence into a rapidly convergent one with the aid of the method of summability and some intricate inversion formulas. The mathematical complexity of this class of methods is prohibitively high, and therefore, the popularity of the epsilon algorithms has declined, especially for complex CFD problems.

[2]This section, in part, is a reprint of the material as it appears in AIAA Journal 55 (9), 3059-3071 titled "*Convergence Acceleration of Fluid Dynamics Solvers Using a Reduced–Order Model*" (2017). Authors: **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici.

Compared to epsilon algorithms, the polynomial methods have a simpler definition and can be extended to higher order formulations. Minimal polynomial extrapolation (MPE) and reduced rank extrapolation (RRE) were derived from the pioneering works of Cabay and Jackson [32], Mešina [157] and Skelboe [187] which are all reviewed by Smith et al. [189]. These methods were later developed with the goal of convergence acceleration for mostly inviscid CFD solvers. Hafez et al. [83] used the power method and the minimal residual method to estimate and minimize the convergence error, which is then used to extrapolate to a solution closer to the "exact" (fully converged) solution. In a similar approach, Dagan [47] used the power method to develop a convergence acceleration algorithm, and Sidi [185] used various extrapolation methods to accelerate the convergence of iterative solvers. On the other hand, Theofilis [207] has used a linear instability-based "residual algorithm" approach to achieve significant reductions in the total simulation cost by devising a stop criteria for the time-integration process.

The fundamental idea of all polynomial methods is to approximate the eigenvalues of the flow solver. These eigenvalues can then be used in the characteristic polynomial to extrapolate the solution that drives the convergence error to machine accuracy [83, 157, 47, 185, 66]. Alternatively, the approximated eigenvalues can be used to determine the unstable modes of the flow solver. Jespersen and Bunning [118] worked on the convergence acceleration of an iterative process for the Euler equations by annihilating the dominant unstable eigenvalues. Also, Ekici et al. [62] extended the same idea to stabilize a Navier-Stokes solver by modifying the eigenvalues of the unstable modes that may drive the system into a divergent or nonconvergent limit-cycle. In that work, Ekici et al. [62] used a proper orthogonal decomposition (POD) method to approximate the eigenvalues of the unstable modes. The POD technique has been applied to various problems in the literature to obtain approximate, low-dimensional descriptions such as those that arise in turbulent fluid flows and structural vibrations, to name a few [101]. In general, data analysis using POD is often conducted to extract "mode shapes", or basis functions, from experimental data or detailed simulations of high-dimensional systems, for subsequent use in Galerkin projections that yield low-dimensional dynamical models. In mechanical engineering and CFD applications, there are numerous studies in the literature that incorporate linear/non-linear Reduced Order

Models (ROMs) [140, 208, 86, 59, 36, 172] as well as stabilization of explicit time-marching solvers [62].

In the framework of POD-based convergence acceleration, Tromeur-Dervout and Vassilevski [210] used proper orthogonal decomposition to obtain a reduced-order based initial guess for the inexact backtracking method in order to accelerate the convergence of a fully implicit solver applied to nonlinear unsteady boundary value problems. In a similar but more recent approach, Shterev [184] used Lagrange interpolation as an extrapolation tool to approximate in time the initial state required by the iterative solver in simulation of unsteady flow problems for the purpose of convergence acceleration. Markovinović and Jansen [144] have used POD-based reduced-order models to accelerate the solution of systems of equations using iterative solvers in time stepping schemes for large-scale numerical simulations. The acceleration is achieved by determining an improved initial guess for the iterative process based on information in the solution vectors from previous time steps. However, it should be noted that the application of POD-based techniques to convergence acceleration is mainly limited to iterative implicit solvers [210, 184, 81].

1.3 Objectives and Contributions

Based on the motivations and the related works in wind turbine design optimization presented earlier in this Chapter, the main objective of this dissertation is to develop a computationally-efficient framework for unsteady wind turbine blade shape optimization. The specific aims of this work as well as the contributions to the state-of-the-art include:

1. Development and validation of a **grid-transparent** two- and three-dimensional unstructured RANS solver. This **UNsturctured PArallel Compressible (UNPAC)** solver is enhanced with (1) the HB method for simulating time-periodic flows as well as (2) the Spalart-Allmaras turbulence model and an algebraic transition model for increased fidelity. Parallel computing capability is also added to the UNPAC solver using a non-overlapping domain decomposition approach and the Message Passing Interface (MPI) standard.

2. Development of a **Fast automatic Differentiation toolbox based on Operator-overloading Technique (FDOT)**. This advanced toolbox can efficiently and accurately calculate the sensitivity information of a cost function with respect to any design variable based on the discrete adjoint method with minimal changes required to be made to the available codes. The novel FDOT toolbox advances the state-of-the-art in discrete adjoint sensitivity analysis based on operator-overloading (OO) automatic differentiation (AD) by utilizing an iterative approach along with a variable flagging technique that can greatly enhance the computational and memory efficiency. To the best of the author's knowledge, none of the existing OO/AD packages have so far addressed issues with inherently large memory footprint. The FDOT toolbox can then be used in tandem with the UNPAC solver as well as a design optimization algorithm to perform aerodynamic shape optimization of the wind turbine blades.

3. Development of **a novel convergence acceleration technique based on the reduced-order-modeling (ROM)** that can significantly increase the performance of the UNPAC solver and further improve the accuracy and efficiency of the design optimization framework.

4. Incorporating a robust ***r*-Adaptive Mesh Redistribution (AMR)** technique in the UNPAC solver. This grid adaptation technique has the potential of being solution-based or adjoint-based although the former approach is sought in this work. Also, the implementation of the AMR tool in the framework of the HB solver would be a direct improvement to the state-of-the-art in grid adaptation for unsteady periodic flows.

5. Developing a design optimization framework, called **UNPAC-DOF**, by coupling the UNPAC solver and the FDOT toolbox for robust aerodynamic shape optimization of airfoils and wind turbine blade cross-sections.

1.4 Outline

The structure of the dissertation is as follows. Governing equations of the fluid dynamics (Euler, Navier-Stokes, and Reynolds-Averaged Navier-Stokes) are presented in Chapter 2

and details regarding the rotating frame of reference and cases involving moving grids are discussed. Additionally, details of the harmonic balance method used in this work are also presented. Next, the numerical procedure involved in the UNPAC solver including the spatial and temporal discretization, boundary treatments, and parallelization is detailed in Chapter 3. Moreover, the novel convergence acceleration technique and the implementation of the AMR approach in the UNPAC solver are presented. In Chapter 4, the FDOT toolbox developed in this work is described. Validation results for the flow solver including the convergence acceleration, AMR, steady, and HB test cases are presentd in Chapter 5 followed by the sensitivity analysis results in Chapter 6. Ultimately, the development of the design optimization framework, UNPAC-DOF, and the aerodynamic shape optimization results using this framework are presented in Chapter 7. This dissertation closes with a summary and the recommendations for future work in Chapter 8.

1.5 Related Published Works

The following journal and conference papers have been authored/co-authored by SeyedReza Djeddi during his PhD:

1. **Reza Djeddi** and Kivanc Ekici, *An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers*, 2018 Fluid Dynamics Conference, AIAA AVIATION Forum, Atlanta, Georgia. (AIAA 2018-3245)

2. Andrew Kaminsky, **Reza Djeddi**, and Kivanc Ekici. *Convergence Acceleration of Continuous Adjoint Solvers Using a Reduced-Order Model.*, International Journal for Numerical Methods in Fluids 86.9 (2018): 582-606.

3. **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici, *Convergence Acceleration of Fluid Dynamics Solvers Using a Reduced–Order Model*, AIAA Journal, 55 (9), 3059-3071 (2017).

4. **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici, *Convergence Acceleration of Fluid Dynamics Solvers Using a Reduced-Order-Model.*, 55th AIAA Aerospace Sciences Meeting. 2017.

5. Andrew Kaminsky, **Reza Djeddi**, and Kivanc Ekici, *An Efficient Reduced-Order-Model for Accurate Projection of Adjoint Sensitivities.*, 55th AIAA Aerospace Sciences Meeting. 2017.

6. **Reza Djeddi** and Kivanc Ekici, *Resolution of Gibbs Phenomenon Using a Modified Pseudo-Spectral Operator in Harmonic Balance CFD Solvers*, International Journal of Computational Fluid Dynamics 30.7-10 (2016): 495-515.

7. **Reza Djeddi**, Jason Howison, and Kivanc Ekici. *A Fully Coupled Turbulent Low-Speed Preconditioner for Harmonic Balance Applications.*, Aerospace Science and Technology 53 (2016): 22-37.

8. **Reza Djeddi**, and Kivanc Ekici. *Modified Spectral Operators for Time-Collocation and Time-Spectral Solvers.*, 54th AIAA Aerospace Sciences Meeting. 2016.

9. **Reza Djeddi**, Jason Howison, and Kivanc Ekici. *A Turbulent Low-Speed Preconditioner for Unsteady Flows About Wind Turbine Airfoils.*, 22nd AIAA Computational Fluid Dynamics Conference. 2015.

It must be noted that items 6 through 9 in the above list include the results of author's earlier PhD research although the findings of those publications are not included in this dissertation. Moreover, regarding the FDOT toolbox developed in this work, an invention disclosure has been filed at the University of Tennessee Research Foundation (UTRF) under invention number 18109-03. Additionally, there is a manuscript related to grid adaptation technique that was explored in this work that is currently under-review:

- **Reza Djeddi** and Kivanc Ekici. *Solution-Based Adaptive Mesh Redistribution Applied to Harmonic Balance Solvers* Manuscript submitted to the Journal of Aerospace Science and Technology.

Chapter 2

Governing Equations and Mathematical Formulation

This chapter describes the governing equations for inviscid, laminar, and turbulent flows. First, the conservation laws for mass, momentum, and energy are presented in Section 2.1. Next, the Navier-Stokes equations are defined in the integral form followed by assumptions and empirical constants in Section 2.2. Using the eddy-viscosity hypothesis and the one-equation Spalart-Allmaras model, the turbulence effects are modeled. Moreover, an algebraic transition model is utilized to address the laminar to turbulent transition in the boundary layer. This leads to the Reynolds-Averaged Navier-Stokes equations described in Section 2.3. Finally, for the unsteady periodic flows, the harmonic balance (HB) method is described and the HB equations are presented in Section 2.4.

2.1 Conservation Laws for a Finite Control Volume

The science of investigating the interactive motion within a large number of individual particles is called "*fluid dynamics*" [21]. These particles are in fact molecules and atoms of the fluid. In order to define density, velocity, pressure, temperature, and other quantities at each point in the fluid, mean velocity and mean kinetic energy must be specified for an element of the fluid (however infinitesimally small). To do so, certain assumptions must be made which are described below:

1. The density of the fluid is high enough that it can be treated as a *continuum*. This requires that the Knudsen number, Kn, to be [93]

$$\mathrm{Kn} \equiv \frac{\lambda}{L} \ll 1,$$

where λ is the mean free path and L is a characteristic length defined locally in the fluid. In a more general sense, the Knudsen number can be approximated as

$$\mathrm{Kn} \approx \frac{M}{\mathrm{Re}} \ll 1,$$

where M is the Mach number and Re is the Reynolds number [93]. It must be noted that this assumption obviously holds for inviscid flows where $\mathrm{Re} \to \infty$.

2. By neglecting the intermolecular forces, the fluid is treated as an ideal gas. In this work, air is assumed to be the working fluid which is treated as thermally and calorically perfect.

3. The fluid is assumed to be Newtonian where the viscous stress is linearly proportional to the strain rate.

Based on these assumptions, the conservation of a scalar quantity, U, per unit volume can be determined in a finite control volume. This control volume is defined as an arbitrary finite region of the flow, $\mathcal{V}$, as shown in Figure 2.1. The control volume is enclosed by the boundary surface $\partial\mathcal{V}$ which, for now, is assumed to be fixed in space (no boundary velocities). A surface element dS and its corresponding unit vector $\vec{n}$ are defined on $\partial\mathcal{V}$ where the normal vector is always pointing outward. Thus, the conservation law for the scalar quantity is written as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} U \; d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[(\vec{F}_C - \vec{F}_D)\cdot\vec{n}\right] \; dS = \int_{\mathcal{V}} Q_V \; d\mathcal{V} \tag{2.1}$$

where F_C and F_D are the convective and diffusive fluxes, respectively, and Q_V is the volumetric source term.

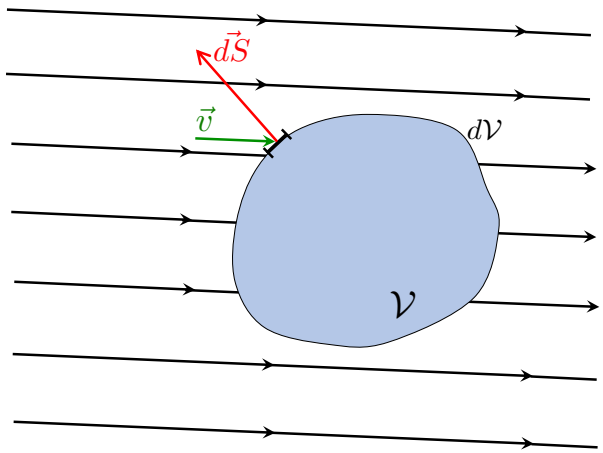


Figure 2.1: Finite control volume with fixed boundaries in space.

The conservation law given in Eq. (2.1) states that the time rate of change of the scalar quantity within the control volume $\mathcal{V}$ plus the balance of the convective and diffusive fluxes across the boundaries of the control volume, $\partial\mathcal{V}$, are equal to the changes due to the volumetric source term. Here, the convective flux determines the amount of the scalar quantity leaving the control volume through the boundaries and the diffusive flux is determined based on Fick's gradient law [98].

While the conservation law described in Eq. (2.1) is for a scalar quantity, a similar conservation law can be written for a vector quantity, $\vec{U}$. However, in such cases, the convective and diffusive flux vectors will turn into convective and diffusive flux tensors and the volumetric source term, Q_V, will become a volumetric source vector, $\vec{Q}_V$. It is worth noting that this conservation law holds throughout the flow field even at the discontinuities such as shocks. Also, in the absence of a source term, the time variations of the conservation variable, $\vec{U}$, will only depend on the fluxes across the boundaries of the control volume.

In the framework of fluid dynamics, three conservation laws are defined. These conservation laws lead to continuity, momentum, and energy equations which will be presented in the following sections.

2.1.1 Continuity Equation

In order to describe the continuity equation, the fluid is assumed to be single-phase which means while the fluid may consist of different chemical species, it is chemically inert. Therefore, the conservation of mass states that mass cannot be produced or destroyed. Assuming that the flow velocity at an arbitrary point on the boundary of the control volume is $\vec{v}$, the continuity equation is written for the conservation variable which is the fluid density, ρ, that is

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho \; d\mathcal{V} + \oint_{\partial\mathcal{V}} \rho(\vec{v}\cdot\vec{n}) \; dS = 0 \tag{2.2}$$

It is worth noting that there is no diffusive flux nor any volumetric source term for the continuity equation.

2.1.2 Momentum Equation

It is known that the momentum (mass times velocity) in a fraction of a control volume, $d\mathcal{V}$, is defined by $\rho\vec{v}d\mathcal{V}$. Thus, the conservation variables for the momentum equation are $\rho\vec{v} = [\rho u, \rho v, \rho w]^T$ in the Cartesian coordinate system. According to Newton's second law of motion, changes in momentum are due to the net force acting on the mass element. Therefore, using the conservation law defined in Section 2.1, the momentum equation within an arbitrary control volume, $\mathcal{V}$, can be written in the integral form as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho\vec{v} \; d\mathcal{V} + \oint_{\partial\mathcal{V}} \rho\vec{v}(\vec{v}\cdot\vec{n}) \; dS + \oint_{\partial\mathcal{V}} p\vec{n} \; dS - \oint_{\partial\mathcal{V}} (\overline{\overline{\tau}}\cdot\vec{n}) \; dS = \int_{\mathcal{V}} \rho\vec{f_e} \; d\mathcal{V} \tag{2.3}$$

In the above equation, the third and fourth terms on the left-hand side (LHS) are in fact source terms representing the external forces acting on the control surface. As such those forces are related to the isotropic pressure, p, and viscous stress tensor, $\overline{\overline{\tau}}$. While the latter is also known as the diffusive flux, the pressure term is usually grouped with the momentum flux (second integral on LHS) to give the total convective flux. Therefore, a more common form of the momentum equation is given by

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho\vec{v}\ d\mathcal{V} + \underbrace{\oint_{\partial\mathcal{V}} [\rho\vec{v}(\vec{v}\cdot\vec{n}) + p\vec{n}]\ dS}_{\text{convective}} - \underbrace{\oint_{\partial\mathcal{V}} (\overline{\overline{\tau}}\cdot\vec{n})\ dS}_{\text{diffusive}} = \int_{\mathcal{V}} \rho\vec{f_e}\ d\mathcal{V} \tag{2.4}$$

The volumetric source term on the right-hand side (RHS) is related to the external body forces which include gravitational, buoyancy, Coriolis, and centrifugal forces that act directly on the mass of the finite control volume. For flows that are modeled in a rotating frame of reference (such as those seen in wind turbine and turbomachinery applications), the inclusion of the Coriolis and centrifugal forces as the source terms of the momentum equation is required, and this will be explained in more detail in the later parts of this document.

2.1.3 Energy Equation

While the Newton's second law is the bedrock of the momentum equation, the first law of thermodynamics is used to derive the energy equation. This law states that the time rate of change of the total energy in a control volume, $\mathcal{V}$, is caused by the collective sum of the rate of work due to forces acting on the control volume as well as the net heat flux across its boundaries. The total energy per unit mass of a fluid is defined as [98]

$$E = e + \frac{|\vec{v}|^2}{2} \tag{2.5}$$

which is the sum of the internal energy, e, and the kinetic energy. Therefore, the conservation variable for the energy equation is defined as the total energy per unit volume, ρE. With the inclusion of the surface source terms and the body forces as the volumetric source terms, the integral form of the conservation of energy equation is written as [21]

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho E\ d\mathcal{V} + \oint_{\partial\mathcal{V}} \rho E(\vec{v}\cdot\vec{n})\ dS + \oint_{\partial\mathcal{V}} p(\vec{v}\cdot\vec{n})\ dS - \oint_{\partial\mathcal{V}} (\overline{\overline{\tau}}\cdot\vec{v})\cdot\vec{n}\ dS - \oint_{\partial\mathcal{V}} k(\nabla T\cdot\vec{n})\ dS = \int_{\mathcal{V}} (\rho\vec{f_e}\cdot\vec{v})\ d\mathcal{V} \tag{2.6}$$

Once again, the surface source term due to pressure forces (third integral on LHS) is combined with the energy flux to give the total convective flux for the energy equation. Based on Fick's law, one must include the effect of a diffusive flux in the governing equation,

and it is defined as the gradient of the conservation variable per unit mass. In principle, this diffusive flux is based on the heat diffusion due to thermal conduction at the molecular level. Therefore, Fourier's law of heat conduction is used in order to relate the thermal conduction to the temperature gradient. It must be noted that in the presence of a volumetric heat source, the volume integral on the RHS of Eq. (2.6) will be augmented by the rate of the heat transfer per unit mass due to this heat source (e.g. absorption or emission of radiation [21]).

In practice, the energy equation is written in a slightly modified form using the total enthalpy which is related to the total energy and pressure via [98]

$$H = h + \frac{|\vec{v}|^2}{2} = E + \frac{p}{\rho}. \tag{2.7}$$

Thus, using Eq. (2.7) and combining convective and diffusive fluxes, the final energy equation can be written in the integral form as below.

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho E \; d\mathcal{V} + \underbrace{\oint_{\partial\mathcal{V}} \rho H(\vec{v}\cdot\vec{n}) \; dS}_{\text{convective}} - \underbrace{\oint_{\partial\mathcal{V}} \left[(\overline{\overline{\tau}}\cdot\vec{v})\cdot\vec{n} + k(\nabla T\cdot\vec{n})\right] \; dS}_{\text{diffusive}} = \int_{\mathcal{V}} (\rho\vec{f_e}\cdot\vec{v}) \; d\mathcal{V} \tag{2.8}$$

With the derivation of the three conservation laws governing the fluid dynamics completed, the system of the Navier-Stokes equations can now be expressed in integral form.

2.2 Navier-Stokes Equations

Named after the French physicist Claude-Louis Navier and the Irish mathematician George Stokes, the Navier-Stokes equations govern the dynamics of the fluid flow. These equations are defined based on the three main conservation laws of mass, momentum, and energy that are collected into a single system of equations. For the sake of brevity, the sum of the convective transport of the conservation variables in the fluid flow and the pressure terms are cast into a vector of convective fluxes, $\vec{F_c}$ while the sum of viscous stresses and the heat diffusion is grouped into a vector of diffusive or viscous fluxes, $\vec{F_v}$. Also, all volumetric source terms (mainly the body forces) are cast into a source term vector, $\vec{Q}$. Thus, the integral

form of the compressible Navier-Stokes equations for a finite control volume, $\mathcal{V}$, with control surface, $\partial\mathcal{V}$ and a surface element, dS, is given by

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \vec{U}\ d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[\vec{F}_c - \vec{F}_v\right]\ dS = \int_{\mathcal{V}} \vec{Q}\ d\mathcal{V} \tag{2.9}$$

where $\vec{U}$ is the vector of conservation variables defined as

$$\vec{U} = \begin{bmatrix} \rho \\ \rho\vec{v} \\ \rho E \end{bmatrix} \tag{2.10}$$

where in Cartesian coordinate system, the velocity vector is defined as $\vec{v} = [u, v, w]^T$. Next, the contravariant velocity term, V, is defined as the velocity normal to the surface element, i.e.,

$$V = \vec{v} \cdot \vec{n} = u\ n_x + v\ n_y + w\ n_z \tag{2.11}$$

Therefore, the convective flux vector can be written as

$$\vec{F}_c = \begin{bmatrix} \rho V \\ \rho u V + p\ n_x \\ \rho v V + p\ n_y \\ \rho w V + p\ n_z \\ \rho H V \end{bmatrix} \tag{2.12}$$

Additionally, the viscous flux vector is defined in terms of the viscous stress tensor and the heat diffusion such that

$$\vec{F}_v = \begin{bmatrix} 0 \\ \vec{\tau}_{x_1} \cdot \vec{n} \\ \vec{\tau}_{x_2} \cdot \vec{n} \\ \vec{\tau}_{x_3} \cdot \vec{n} \\ \vec{\Theta} \cdot \vec{n} \end{bmatrix} \tag{2.13}$$

where $\vec{\tau}_{x_i}$ $(i = 1, 2, 3)$ is the i-th row of the viscous stress tensor given by

$$\overline{\overline{\tau}} = \begin{bmatrix} \vec{\tau}_{x_1} \\ \vec{\tau}_{x_2} \\ \vec{\tau}_{x_3} \end{bmatrix} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} \tag{2.14}$$

Also, the elements of the $\vec{\Theta}$ vector which describes the work of the viscous stresses as well as the heat conduction in the fluid flow, are defined as

$$\Theta_i = \vec{v} \cdot \vec{\tau}_{x_i} + k\frac{\partial T}{\partial x_i}; \quad i = 1, 2, 3 \tag{2.15}$$

The diagonal terms in the viscous stress tensor are the normal stresses while the rest are shear stresses. In general, the viscous stress tensor includes the dynamic viscosity, μ, and a second viscosity coefficient, λ, which are related according to the Stokes hypothesis [198] via

$$2\mu + 3\lambda = 0. \tag{2.16}$$

As stated earlier, for a Newtonian fluid, the shear stresses are proportional to the velocity gradients. Thus, with the assumption of a Newtonian fluid and using Eq. (2.16), the viscous stresses (both normal and shear components) can be written as [98]

$$\tau_{ij} = \mu \left[\left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j} \right) - \frac{2}{3} \left(\vec{\nabla} \cdot \vec{v} \right) \delta_{ij} \right] \tag{2.17}$$

where δ_{ij} is the Kronecker delta.

Finally, in the absence of a volumetric heat source, the vector of source terms is only described by the external body forces (including Coriolis and centrifugal terms) such that

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho f_{e,y} \\ \rho f_{e,z} \\ \rho \vec{f}_e \cdot \vec{v} \end{bmatrix} \tag{2.18}$$

As can be seen, the Navier-Stokes equations consist of five equations for the vector of conservation variables, $\vec{U}$. However, these equations contain seven unknown primitive variables, $\vec{W} = [\rho, u, v, w, p, T, E]^T$. In order to complete the system of equations, additional expressions are required.

Based on the assumptions described earlier in Section 2.1, in this work the fluid is considered to be a calorically perfect gas. Therefore, the equation of state for a perfect gas [98]

$$p = \rho R T \tag{2.19}$$

can be utilized, where R is the specific gas constant. The changes in the specific enthalpy are also related to the changes in the absolute temperature via [159]

$$dh = c_p \, dT. \tag{2.20}$$

Thus, according to

$$R = c_p \; - c_v \; , \quad \gamma \; = \frac{c_p}{c_v}, \tag{2.21}$$

with γ being the gas constant (ratio of the specific heat coefficients), the pressure can be expressed in terms of the conservation variables such that

$$p = (\gamma - 1) \left[\rho E - \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho} \right] \tag{2.22}$$

Additionally, the dynamic viscosity, μ, is related to the absolute temperature via the Sutherland formula [200] (named after the Australian physicist William Sutherland) given by

$$\mu = \frac{C_{\text{suth}} T^{3/2}}{T + T_{\text{suth}}} \tag{2.23}$$

where T_{suth} is the Sutherland temperature and C_{suth} is a constant based on the reference temperature and reference viscosity. In this work, air is considered to be the working fluid

and therefore, various empirical constants and necessary coefficients for air at standard temperature and pressure (STP) are given in Table 2.1.

Table 2.1: Empirical constants and closure coefficients for air (with an ideal gas assumption).

constant	value	units
R	287.04	J/kg-K
γ	1.4	-
c_p	1004.64	J/kg-K
Pr	0.72	-
T_{suth}	110.4	K
C_{suth}	1.458×10^{-6}	K

While the thermal conductivity coefficient, k, is almost constant throughout the fluid for liquids, it varies with temperature in gases and can be indirectly related to the dynamic viscosity. Therefore, the thermal conductivity coefficient can be written for air as [21]

$$k = c_p \frac{\mu}{\text{Pr}} \tag{2.24}$$

It is worth noting that in some cases, the heat flux in Eq. (2.15) can be written in terms of gradient of the specific enthalpy according to Eq. (2.20) and Eq. (2.24) to have

$$\Theta_i = \vec{v} \cdot \vec{\tau}_{x_i} + \left(\frac{\mu}{\text{Pr}}\right) \frac{\partial h}{\partial x_i}; \quad i = 1, 2, 3 \tag{2.25}$$

It must be noted that the dynamic viscosity that is used throughout this section will be later replaced by an effective dynamic viscosity in the framework of the Reynolds-Averaged Navier-Stokes equations which will be discussed in Section 2.3.

2.2.1 Rotating Frame of Reference

For the simulation of fluid flow about wind turbine blades and helicopter rotors, and in turbomachinery, where the computational domain undergoes a steady rotation about an arbitrary axis, it is suitable to write the Navier-Stokes equations in a rotating frame of reference. As opposed to the inertial frame of reference, rotating or moving reference frame

(MRF) allows one to study an unsteady problem with a steady rotational frequency as a steady problem in rotating frame of reference.

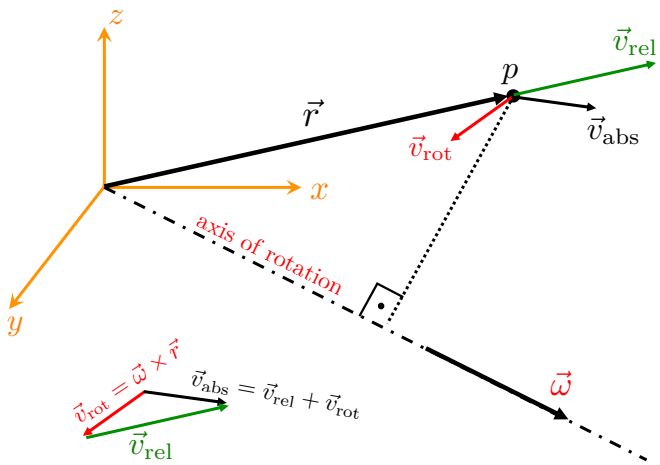


Figure 2.2: Steady rotation with angular velocity vector, $\vec{\omega}$, around an arbitrary axis. Note the relation between the absolute and rotating frames of reference as well as the relative and entrainment (rotating) velocity vectors.

Let us assume that there is a constant rotation with angular velocity vector, $\vec{\omega}$, about an arbitrary axis, as depicted in Figure 2.2. It is easy to infer that the absolute velocity, $\vec{v}_{\text{abs}}$, is a collective sum of the relative, $\vec{v}_{\text{rel}}$, and entrainment (rotating), $\vec{v}_{\text{rot}}$, velocity vectors such that

$$\vec{v}_{\text{abs}} = \vec{v}_{\text{rel}} + \vec{v}_{\text{rot}} = \vec{v}_{\text{rel}} + \vec{\omega} \times \vec{r}. \tag{2.26}$$

where $\vec{r}$ is the position vector for an arbitrary point, p, in space (see Figure 2.2).

As discussed earlier, rotating frame of reference necessitates the inclusion of two external body forces due to Coriolis acceleration and centrifugal force. According to Figure 2.2, for an arbitrary point in space with position vector $\vec{r}$, these forces (per unit mass) are given by

$$\vec{f}_{\text{Cor}} = -2\left(\vec{\omega} \times \vec{v}_{\text{rel}}\right) \tag{2.27}$$

$$\vec{f}_{\text{cent}} = -\vec{\omega} \times \left(\vec{\omega} \times \vec{r}\right) \tag{2.28}$$

Therefore, the Navier-Stokes equations can be recast in terms of relative velocity components as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \vec{U}_{\text{rel}}\ d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[\vec{F}_{c_{\text{rel}}} - \vec{F}_{v_{\text{rel}}}\right]\ dS = \int_{\mathcal{V}} \vec{Q}_{\text{rel}}\ d\mathcal{V} \tag{2.29}$$

where $\vec{U}_{\text{rel}}$ is the vector of conservation variables in relative frame of reference defined as

$$\vec{U}_{\text{rel}} = \begin{bmatrix} \rho \\ \rho\vec{v}_{\text{rel}} \\ \rho E_{\text{rel}} \end{bmatrix} \tag{2.30}$$

where in Cartesian coordinate system, the relative velocity vector is defined as $\vec{v}_{\text{rel}} = [u_{\text{rel}}, v_{\text{rel}}, w_{\text{rel}}]^T$. Here, the relative total energy is given by

$$E_{\text{rel}} = e + \frac{|\vec{v}_{\text{rel}}|^2}{2} - \frac{|\vec{v}_{\text{rot}}|^2}{2} = e + \frac{u_{\text{rel}}^2 + v_{\text{rel}}^2 + w_{\text{rel}}^2}{2} - \frac{|\vec{v}_{\text{rot}}|^2}{2} \tag{2.31}$$

Once again, with the definition of a contravariant relative velocity term, V_{rel}, as the relative velocity normal to the surface element with unit normal vector $\vec{n}$, i.e.,

$$V_{\text{rel}} = \vec{v}_{\text{rel}} \cdot \vec{n} = u_{\text{rel}}\ n_x + v_{\text{rel}}\ n_y + w_{\text{rel}}\ n_z, \tag{2.32}$$

the convective flux vector in relative frame of reference can be defined as

$$\vec{F}_{c_{\text{rel}}} = \begin{bmatrix} \rho V_{\text{rel}} \\ \rho u_{\text{rel}} V_{\text{rel}} + p\ n_x \\ \rho v_{\text{rel}} V_{\text{rel}} + p\ n_y \\ \rho w_{\text{rel}} V_{\text{rel}} + p\ n_z \\ \rho I V_{\text{rel}} \end{bmatrix} \tag{2.33}$$

where I is the *rothalpy* representing the total energy content in a steadily rotating frame of reference [21] defined as

$$I = h + \frac{|\vec{v}_{\text{rel}}|^2}{2} - \frac{|\vec{v}_{\text{rot}}|^2}{2} = H_{\text{rel}} - \frac{|\vec{v}_{\text{rot}}|^2}{2} \tag{2.34}$$

where H_{rel} is the relative total enthalpy. The viscous flux vector will have the same form as in Eq. (2.13) with the only difference that the velocity components in the viscous stress tensor (2.17) are now replaced by relative velocities.

Finally, the Coriolis and centrifugal forces are now included as the external body forces in the source term vector. Therefore, in the absence of any other body force and heat source in Eq. (2.18), the source term vector in relative frame of reference is given by

$$\vec{Q}_{\text{rel}} = \begin{bmatrix} 0 \\ \rho\left(\vec{f}_{\text{Cor}} + \vec{f}_{\text{cent}}\right) \\ 0 \end{bmatrix} \tag{2.35}$$

Once again, the closure problem is addressed using thermodynamic relations between the state variables. Therefore, the static pressure in Eq. (2.33) is defined for a calorically perfect gas using

$$p = (\gamma - 1)\left[\rho E - \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho} + \rho\,\frac{(u_{\text{rot}})^2 + (v_{\text{rot}})^2 + (w_{\text{rot}})^2}{2}\right] \tag{2.36}$$

where u_{rot}, v_{rot}, and w_{rot} are the components of the entrainment or rotational velocity vector given by Eq. (2.26). For special cases, where the rotation axis is aligned with one of the main axes in the Cartesian coordinate system, the governing equations and extra details are provided in Appendix A.

2.2.2 Navier-Stokes Equations for Moving Grids

With the advancements in computational resources, the simulation of fluid flows involving deforming or moving geometries has become more relevant. These applications include flutter analysis, oscillating rotorcraft, aerodynamic shape optimization, to name a few. For these cases, a robust mesh motion technique is required to conform the geometry deformations and movements. In particular, during an aerodynamic shape optimization process, the geometry is deformed during the design cycles which requires the body-fitted mesh to deform in order to accommodate the changes in the topology.

Grid movement during a simulation can be challenging since in some cases it can lead to the violation of the mass conservation which can result in significant accuracy degradation. In order to remedy these issues, an *Arbitrary Lagrangian Eulerian* (ALE) approach [55] is used which relies on a continuous switching between Lagrangian and Eulerian points of view. This means that while the Eulerian point of view is considered for satisfying the conservation laws of fluid dynamics in a finite control volume, relative velocities are considered for convective flux computations across the boundaries. This mimics the Lagrangian point of view where the frame of reference is attached to the control surfaces and hence no movement is considered and only fluxes across the boundaries are integrated. Therefore, the Navier-Stokes equations can be modified to reflect this effect and are re-written as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \vec{U}\ d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[\vec{F}_c^{\mathrm{ALE}} - \vec{F}_v\right]\ dS = \int_{\mathcal{V}} \vec{Q}\ d\mathcal{V} \tag{2.37}$$

where the ALE formulation of the convective fluxes, $\vec{F}_c^{\mathrm{ALE}}$, is as an augmented form of the original convective flux vector, and includes the effect of moving grid velocities such that

$$\vec{F}_c^{\mathrm{ALE}} = \vec{F}_c - (\vec{v}_{\mathrm{grid}} \cdot \vec{n})\,\vec{U}. \tag{2.38}$$

where $\vec{n}$ is, once again, the unit normal vector that is pointing outward on the control surface $\partial\mathcal{V}$. Here, the control surface is assumed to be moving according to the grid velocity vector, $\vec{v}_{\mathrm{grid}}$, that is defined as

$$\vec{v}_{\mathrm{grid}} = [\ \dot{x}, \dot{y}, \dot{z}\ ]^T \tag{2.39}$$

where $\dot{x}$, $\dot{y}$, and $\dot{z}$ are the time variations of the mesh defined in Cartesian coordinates. Thus, using the definition of the contravariant flow velocity, V, and the contravariant grid velocity, V_{grid}, the ALE form of the convective flux can be written as

$$\vec{F}_c^{\text{ALE}} = \begin{bmatrix} \rho V \\ \rho u V + p\ n_x \\ \rho v V + p\ n_y \\ \rho w V + p\ n_z \\ \rho H V \end{bmatrix} - \begin{bmatrix} \rho V_{\text{grid}} \\ \rho u V_{\text{grid}} \\ \rho v V_{\text{grid}} \\ \rho w V_{\text{grid}} \\ \rho E V_{\text{grid}} \end{bmatrix} \tag{2.40}$$

where $V_{\text{grid}} = \vec{v}_{\text{grid}} \cdot \vec{n}$. Using Eq. (2.7), the two vectors can be combined to give

$$\vec{F}_c^{\text{ALE}} = \begin{bmatrix} \rho V_r \\ \rho u V_r + p\ n_x \\ \rho v V_r + p\ n_y \\ \rho w V_r + p\ n_z \\ \rho H V_r + p\ V_{\text{grid}} \end{bmatrix} \tag{2.41}$$

where V_r is the contravariant velocity relative to the grid motion, i.e.,

$$V_r = (\vec{v} \cdot \vec{n}) - (\vec{v}_{\text{grid}} \cdot \vec{n}) = V - V_{\text{grid}} \tag{2.42}$$

While the viscous fluxes and the source terms remain the same, the contravariant flow velocity must be replaced by the contravariant relative velocity in the calculation of the Jacobian of the convective flux (and hence the spectral radii) according to the ALE formulation described here. These modifications due to grid motion will be discussed in more detail in Chapter 3.

In order to avoid numerical errors produced by the deformation of the median-dual control volumes, an additional conservation law must be solved simultaneously with the rest of the governing equations. This additional equation that was first developed by Thomas and Lombard [209] is referred to as the Geometric Conservation Law (GCL). If not accounted for, the violation of the GCL can lead to significant degradation of accuracy especially in aeroelastic computations [135]. The GCL is derived by first formulating the conservation of mass or the continuity equation for moving grids such that

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho \; d\mathcal{V} + \oint_{\partial\mathcal{V}} \rho(V - V_{\text{grid}}) \; dS = 0 \tag{2.43}$$

The ALE form of the continuity equation described by Eq. (2.43) must hold even for cases with a constant density and a uniform flow velocity. Therefore, it can be shown that in such cases the density terms can be cancelled and the integral of the contravariant velocity will be zero over a closed control volume. Thus, the GCL equation in integral form can be written as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \; d\mathcal{V} + \oint_{\partial\mathcal{V}} V_{\text{grid}} \; dS = 0 \tag{2.44}$$

which states that the rate of the total change of the control volume (CV) must be equal to the rate of incremental volume change due to the movement of the boundaries of the CV.

The geometric conservation equation given as Eq. (2.44) is usually solved in the framework of unsteady time-accurate solvers using the same numerical scheme that was used to discretize and solve the rest of the governing equations. This is necessary for obtaining a consistent solution method [21] and was also extended to non-linear frequency domain (NLFD) techniques [205]. Alternatively, Ma et al. [142] have shown that a source term approach can also be used to preserve the GCL condition eliminating the need to solve an additional conservation equation. In this work, the source term approach of Ma et al. [142] is used and the details of this method are presented in Chapter 3.

2.3 Reynolds-Averaged Navier-Stokes Equations

The Navier-Stokes equations presented in the previous section can be solved to simulate inviscid and laminar flows. In fact, in the absence of the effects of viscosity or for cases where the ratio of the kinematic to viscous forces is too large, the Navier-Stokes equations result in the well-known Euler equations that are used for the simulation of inviscid flows. However, many fluid flow problems in engineering applications involve turbulent flows whose direct simulations require considerable and prohibitively costly computational resources. In fact, the computational cost of the direct simulation of the turbulent flow features, known

as *direct numerical simulations* (DNS), can easily grow out of hand due to the fact that the required number of grid nodes is proportional to $\mathrm{Re}^{9/4}$ [21]. Therefore, the application of the DNS is limited to small scale problems at low to moderate Reynolds numbers. Despite its significant computational demand, DNS is used for calibrating turbulence models as well as advanced studies of laminar to turbulent transition phenomenon, to name a few.

As an alternative, *large-eddy simulations* (LES) have been introduced. In the framework of LES, only large scale eddies are resolved directly while approximative techniques (namely "sub-grid scale" or SGS) are used to capture small scale turbulent features. This trade-off leads to more efficient simulations compared to DNS while providing a reasonable accuracy that can improve the understanding of the turbulent structures. However, the computational cost of these two techniques is still high, making their use unsuitable in the design stage.

As a result, Reynolds-Average Navier-Stokes (RANS) models have been pursued which can significantly reduce the computational cost of turbulence modeling, making them suitable candidates for simulations in the design stage. In this work, the RANS equations are solved in the framework of the UNPAC solver and the details of the governing equations, turbulence, and transition models are described in this section.

2.3.1 Reynolds Averaging and Eddy-Viscosity Hypothesis

The concept of Reynolds averaging, first introduced by Reynolds [174], decomposes the flow quantities into a mean and a fluctuating part. As an example, the density can be described as

$$\rho = \overline{\rho} + \rho' \tag{2.45}$$

where $\overline{\rho}$ is the mean part approximated by an averaging process and ρ' is the fluctuating part. As a common practice (for flows at Mach numbers below 5), Morkovin's hypothesis [160] is used which assumes that $\rho' \ll \overline{\rho}$, thus the density fluctuations can be ignored. However, the substitution of the mean and fluctuating parts of the conservation variables (similar to the one shown in Eq. [2.45]) into the governing equations results in two additional terms given as

$$\overline{\overline{\tau}}^R_{ij} = -\overline{\rho}\ \overline{v'_i\ v'_j} \tag{2.46}$$

$$\vec{F}^T_D = -\overline{\rho}\ \overline{h'\ \vec{v}'} \tag{2.47}$$

The first term is called the Reynolds-stress tensor [174] which includes the mean density, $\overline{\rho}$, and the mean value for the product of the two fluctuating velocity components, v'_i and v'_j. The second term is called the turbulent heat-flux vector [98] which includes the mean density and the mean value for the product of the enthalpy, h', and velocity vector fluctuations, $\vec{v}'$. Due to the appearance of these two additional terms, the RANS equations are no longer closed and complementary assumptions and equations are required to address this closure problem which will be discussed next.

Eddy-Viscosity Hypothesis

The eddy-viscosity hypothesis, also known as Boussinesq hypothesis, was first introduced by French physicist Joseph Boussinesq and assumes a linear correlation between the turbulent shear stress and mean rate of strain [21]. Therefore, the Reynolds-stress tensor can be now written as

$$\overline{\overline{\tau}}^R_{ij} = -\overline{\rho}\ \overline{v'_i\ v'_j} = 2\mu_T \tilde{S}_{ij} - \frac{2}{3}\overline{\rho}\tilde{K}\delta_{ij} - \left(\frac{2\mu_T}{3}\right)\frac{\partial \tilde{v}_k}{\partial x_k}\delta_{ij} \tag{2.48}$$

where $\tilde{S}_{ij}$ is the averaged strain rate, $\tilde{K}$ is the averaged turbulent kinetic energy, and μ_T is the eddy viscosity [227]. The main result of the Boussinesq hypothesis is the introduction of the eddy viscosity which is directly related to the local flow properties. In fact, the main purpose of the turbulence models is to determine this quantity based on the flow conditions and thus closing the system of equations.

The turbulent heat-flux vector or the Reynolds enthalpy flux [93], is approximated using the gradient-diffusion hypothesis [226, 227] originally proposed by Reynolds [175] so that

$$\vec{F}^T_D = -\overline{\rho}\ \overline{h'\ \vec{v}'_i} = -k_T \frac{\partial \tilde{T}}{\partial x_i} \tag{2.49}$$

where k_T is called the turbulent thermal conductivity coefficient [227, 21] given by

$$k_T = c_p \frac{\mu_T}{\mathrm{Pr}_T} \tag{2.50}$$

Here, Pr_T is the turbulent Prandtl number which is assumed to be constant throughout the fluid domain and is taken to be $\mathrm{Pr}_T = 0.9$ for air. Once again, it can be seen that the eddy viscosity, μ_T, is required to approximate the turbulent heat-flux vector. It must be noted that since the RANS equations are written and solved for averaged or mean components of the flow variables, the tilde sign can be omitted for clarity. The eddy-viscosity hypothesis has certain flaws and a discussion about its weaknesses and the conditions at which this hypothesis fails is presented by Wilcox [227].

As discussed previously in section 2.2, in the framework of the RANS equations, the dynamic viscosity μ is replaced by an "*effective*" viscosity which is taken to be the sum of the laminar and eddy viscosities, i.e.,

$$\mu_{\mathrm{eff}} = \mu_L + \mu_T \tag{2.51}$$

where the laminar viscosity is calculated using the Sutherland's law (see Eq. [2.23]) and the eddy viscosity is computed via the turbulence model. Similarly, the thermal conductivity coefficient can be written as a sum of laminar and turbulent components via

$$k = k_L + k_T = c_p \left(\frac{\mu_L}{\mathrm{Pr}_L} + \frac{\mu_T}{\mathrm{Pr}_T} \right) \tag{2.52}$$

where Pr_L is given in Table 2.1 for air as the working fluid. Next, the turbulence model used in this work to complete the RANS governing equations will be presented.

2.3.2 Spalart-Allmaras Turbulence Model

As discussed earlier in this chapter, in order to close the RANS equations, the eddy viscosity must be approximated. For this reason, closure models are often used which depending on the number of equations that need to be solved, are classified as (1) zero-equation or algebraic, (2) one-equation, and (3) two-equation turbulence models. The zero-equation or

algebraic turbulence models use empirical formulations based on the local flow conditions to calculate the eddy viscosity. Among these, the Baldwin and Lomax [10] algebraic model is the most widely used for aerodynamic applications. However, the accuracy of the eddy viscosity evaluation is directly related to the flow history which is neglected in the algebraic models. This is addressed by solving a transport equation for the convective and diffusive components of the turbulent features, which is the case for one-equation and two-equation turbulence models.

In this work, the modified one-equation Spalart-Allmaras turbulence model [194] is used which is enhanced with a "rotation correction" mechanism that reduces the eddy viscosity in the vicinity of regions where the vorticity surpasses the strain rate [46, 45]. The Spalart-Allmaras (SA) turbulence model is capable of accurately predicting adverse pressure gradients [227] while maintaining a local (grid-transparent) formulation that makes it suitable for implementation in the framework of structured, unstructured, and mixed-grid solvers.

By taking advantage of the eddy-viscosity hypothesis, the SA model can be written in a conservation form similar to the rest of the governing equations. Therefore, in this work, the RANS equations are augmented by the turbulence model and solved in a fully-coupled fashion. The SA model can be written in the integral form for a finite control volume (see Figure 2.1) as

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \rho\tilde{\nu}\ d\mathcal{V} + \oint_{\partial\mathcal{V}} \left(F_c^{SA} - F_v^{SA}\right)\ dS = \int_{\mathcal{V}} Q^{SA}\ d\mathcal{V} \tag{2.53}$$

where $\tilde{\nu}$ is the eddy viscosity variable or the working variable of the SA turbulence model. Here, the convective and viscous fluxes of the SA model are given by

$$F_c^{SA} = \rho\tilde{\nu}V \tag{2.54}$$

$$F_v^{SA} = \vec{\tau}_{x_i x_i}^{SA} \cdot \vec{n} \tag{2.55}$$

where V is once again the contravariant flow velocity and $\vec{\tau}^{SA}_{x_i x_i}$ is the vector of diagonal or normal viscous stresses for the SA model. The components of the normal stresses for the SA turbulence model are defined as [227]

$$\begin{aligned}
\tau^{SA}_{xx} &= \frac{\rho}{\sigma}\left(\nu_L + \tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial x} \\
\tau^{SA}_{yy} &= \frac{\rho}{\sigma}\left(\nu_L + \tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial y} \\
\tau^{SA}_{zz} &= \frac{\rho}{\sigma}\left(\nu_L + \tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial z}
\end{aligned} \tag{2.56}$$

where the laminar kinematic viscosity, ν_L, is defined by

$$\nu_L = \frac{\mu_L}{\rho} \tag{2.57}$$

The most important part of the SA turbulence model is its volumetric source term which is defined as [21]

$$Q^{SA} = \mathcal{P} - \mathcal{D} + c_{b2}(\nabla\tilde{\nu})^2 \tag{2.58}$$

where the production, $\mathcal{P}$, and wall destruction, $\mathcal{D}$, terms read

$$\mathcal{P} = c_{b1}\tilde{S}\tilde{\nu}, \quad \mathcal{D} = (c_{w1}f_w)\left[\frac{\tilde{\nu}}{d}\right]^2 \tag{2.59}$$

with $\tilde{S}$ denoting the modified vorticity, i.e.,

$$\tilde{S} = |\vec{\Omega}| + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.60}$$

where $|\vec{\Omega}|$ is the magnitude of the vorticity, and d is the distance to the closest wall. Finally, the turbulent (eddy) viscosity, μ_T, as defined by the standard Spalart-Allmaras turbulence model [194] is given by

$$\mu_T = \rho\tilde{\nu} f_{v1}(\chi) \tag{2.61}$$

where

$$f_{v1}(\chi) = \frac{\chi^3}{\chi^3 + c_{v1}^3}\ , \quad \chi = \frac{\mu_t}{\mu_l} \tag{2.62}$$

The function f_w is:

$$f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6}, \quad g = r + c_{w2}(r^6 - r), \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2},\ r_{\lim}\right) \tag{2.63}$$

The constants used in the above definitions are taken from [5] and, for completeness, are given in Table 2.2.

Table 2.2: Constants and closure coefficients for the Spalart-Allmaras turbulence model.

constant	value	constant	value
c_{b1}	0.1355	κ	0.41
c_{b2}	0.622	c_{w1}	$c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma$
c_{v1}	7.1	c_{w2}	0.3
c_{v2}	5.0	c_{w3}	2.0
σ	2/3	$r_{\lim}$	10.0

It must be noted that the SA turbulence model presented in Eq. (2.53) is a single partial differential equation written in the integral form similar to the RANS equations provided earlier in this section. In this work, the Navier-Stokes equations (2.9) and the SA turbulence model (2.53) are coupled together to give the final set of RANS-SA governing equations. These equations are presented in the integral form as below

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \vec{U}\ d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[\vec{F}_c - \vec{F}_v\right]\ dS = \int_{\mathcal{V}} \vec{Q}\ d\mathcal{V} \tag{2.64}$$

where $\vec{U}$ is the vector of conservation variables defined as

$$\vec{U} = \begin{bmatrix} \rho \\ \rho\vec{v} \\ \rho E \\ \rho\tilde{\nu} \end{bmatrix} \tag{2.65}$$

and the convective and viscous fluxes are given by

$$\vec{F_c} = \begin{bmatrix} \rho V \\ \rho u V + p\ n_x \\ \rho v V + p\ n_y \\ \rho w V + p\ n_z \\ \rho H V \\ \rho \tilde{\nu} V \end{bmatrix}, \quad \vec{F_v} = \begin{bmatrix} 0 \\ \vec{\tau}_{x_1} \cdot \vec{n} \\ \vec{\tau}_{x_2} \cdot \vec{n} \\ \vec{\tau}_{x_3} \cdot \vec{n} \\ \vec{\Theta} \cdot \vec{n} \\ \vec{\tau}^{SA}_{x_i x_i} \cdot \vec{n} \end{bmatrix} \tag{2.66}$$

where V is the contravariant flow velocity and the viscous stress tensor, work of the viscous stresses, and the normal stresses of the SA turbulence model are calculated according to Eqs. (2.14), (2.15), and (2.56), respectively. Additionally, the augmented vector of source terms now reads

$$\vec{Q} = \begin{bmatrix} 0 \\ \rho f_{e,x} \\ \rho f_{e,y} \\ \rho f_{e,z} \\ \rho \vec{f_e} \cdot \vec{v} \\ Q^{SA} \end{bmatrix} \tag{2.67}$$

where the source term for the SA turbulence model is given by Eq. (2.58) and body forces for the momentum and energy equations can be described by the Coriolis and the centrifugal forces in the case of the rotating frame of reference. In summary, 5 or 6 equations are solved for two-dimensional or three-dimensional problems, respectively.

2.3.3 B-C Transition Model

In external flows such as those arising in wind turbine simulation and rotorcraft problems, the boundary layer is initially laminar before transitioning to turbulent further downstream. However, the Spalart-Allmaras turbulence model works with an assumption of fully turbulent flow throughout the fluid domain. In fact, the SA model is capable of simulating a transition between laminar and turbulent flows at a pre-specified location. However, this requires an

a priori knowledge of the tripping point which can be very challenging to determine in off-design conditions and for the unconventional designs. Therefore, transition models can be utilized to predict the laminar-turbulent transition phenomenon. In fact, Howison and Ekici [104] have shown the necessity of a transition model in accurately predicting the static and dynamic stall for wind turbine blades in the framework of a RANS-SA solver.

Similar to the turbulence models, the transition models can also be classified based on the number of additional equations that are required to be solved for the prediction of the flow separation and reattachment. A widely used approach is the two-equation $\gamma - \overline{\mathrm{Re}_{\theta\ t}}$ transition model of Langtry and Menter [128, 129]. Although, it was initially proposed for the two-equation $k - \omega$ SST turbulence model [155, 156], it was later adapted for the one-equation Spalart-Allmaras turbulence model by Medida and Baeder [153, 152] and also used in the framework of time-spectral methods by Howison and Ekici [104].

More recently, Baş and Çakmakçıoğlu [14, 34] developed a zero-equation (algebraic) correlation-based transition model that is readily applicable to the SA turbulence model. This transition model (known as B-C model) uses an intermittency, γ_{trans}, function rather than solving an intermittency transport equation, as is the case in the $\gamma - \overline{\mathrm{Re}_{\theta\ t}}$ transition model. It is shown [34] that the intermittency function can effectively and accurately predict the laminar-turbulent transition without having to solve any additional equations. Based on the local flow conditions, the intermittency scalar is calculated which will then scale the production term of the SA turbulence model. Therefore, unless the turbulence onset requirements are met, the production of eddy viscosity by the turbulence model is damped.

In this work, the B-C transition model [33] is utilized which will be incorporated into our RANS-SA solver with minor modifications. Therefore, the source term of the SA turbulence model (2.58) is modified to include the intermittency scalar, γ_{trans}, which varies between 0 and 1, where the lower and upper bounds correspond to laminar and fully turbulent flow conditions, respectively. Thus, the modified SA source term now reads

$$Q^{SA} = \gamma_{\text{trans}}\, \mathcal{P} - \mathcal{D} + c_{b2}(\nabla\tilde{\nu})^2 \tag{2.68}$$

The intermittency scalar is a function of vorticity Reynolds number, Re_ν, momentum thickness Reynolds number, Re_θ, as well as a critical eddy viscosity, $\tilde{\nu}_{\mathrm{cr}}$. The intermittency function is defined as [14]

$$\gamma_{\mathrm{trans}} = 1 - e^{-(T_{\mathrm{Re}}+T_{\tilde{\nu}})} \tag{2.69}$$

where the Reynolds-based terms and the eddy-viscosity-based terms are given by [34]

$$T_{\mathrm{Re}} = \sqrt{\frac{\max(\frac{\mathrm{Re}_\nu}{c_{t_1}} - \mathrm{Re}_{\theta c}, 0)}{c_{t_2}\mathrm{Re}_{\theta c}}} \tag{2.70}$$

$$T_{\tilde{\nu}} = \sqrt{\frac{\max(\tilde{\nu}_{\mathrm{cr}} - c_{t_3}, 0)}{c_{t_3}}} \tag{2.71}$$

The vorticity Reynolds number and the critical eddy viscosity are defined based on the local flow conditions (including the density, velocity, laminar dynamic viscosity, and the wall distance) via

$$\mathrm{Re}_\nu = \frac{\rho \Omega d^2}{\mu} \tag{2.72}$$

$$\tilde{\nu}_{\mathrm{cr}} = \frac{\tilde{\nu}}{|\vec{v}|\ d} \tag{2.73}$$

where Ω is the vorticity magnitude, $|\vec{v}|$ is the velocity magnitude, $\tilde{\nu}$ is the turbulent eddy viscosity (working variable) of the SA turbulence model, and d is the closest wall distance.

Now, the only remaining part is to determine the critical momentum thickness, $\mathrm{Re}_{\theta c}$. In order to calculate this parameter, a correlation based on the experimental results is used. This correlation is based on a zero-pressure gradient assumption [154] and is only a function of the free-stream turbulence intensity, Tu_∞. In the framework of the $k-\omega$ SST turbulence model, the local turbulence intensity is calculated based on the solution of the turbulent kinetic energy, k. Since the SA turbulence model does not solve for this quantity, a user-specified free-stream turbulence intensity is used. According to Menter et al. [154], the experimental correlation for the critical momentum thickness is defined as

$$\mathrm{Re}_{\theta c} = c_{\theta_1} \left(\mathrm{Tu}_\infty + c_{\theta_2}\right)^{-c_{\theta_3}} \tag{2.74}$$

The coefficients of the B-C transition model are given in Table 2.3. It must be noted that the first coefficient, c_{t_1}, is a proportionality constant that relates the momentum thickness and the vorticity Reynolds number [154].

Table 2.3: Coefficients of the B-C transition model [34].

constant	value	constant	value
c_{t_1}	2.193	c_{θ_1}	803.73
c_{t_2}	0.002	c_{θ_2}	0.6067
c_{t_3}	5.0	c_{θ_3}	1.027

It must be noted that the B-C transition model has been previously applied to zero-pressure gradient flow over a flat-plate, transonic flow around DLR-F5 wing, as well as the low-speed NREL wind turbine and good agreements between the numerical and experimental results are reported [33]. Moreover, the fact that no additional equation is solved to simulate the laminar to turbulent transition, makes this model even more attractive. Therefore, the B-C transition model described in this section is incorporated in the framework of the UNPAC solver.

2.4 Harmonic Balance Equations

As explained earlier, since many flows of interest considered in this work are time-periodic, the conservative variables for which the governing equations are solved, can be written in terms of a truncated Fourier series up to a predefined number of harmonics by:

$$\vec{U}^*(t_i) = \mathbf{A}_0 + \sum_{n=1}^{N_H} \left[\mathbf{A}_n \cos(\Omega n t_i) + \mathbf{B}_n \sin(\Omega n t_i)\right] \;\; ; \;\; i = 1 \; : \; 2N_H + 1 \tag{2.75}$$

where Ω is the fundamental frequency of excitation and $\mathbf{A}_0$, $\mathbf{A}_n$ and $\mathbf{B}_n$ are the Fourier series coefficients defined in terms of the conservative variables. The Fourier series is truncated in a way that the flow variables are computed and stored at $2N_H + 1$ equally-spaced sub-time levels over a single period. Based on Eq. (2.75) and using a discrete Fourier transform,

the Fourier coefficients can be determined from the solutions stored at each sub-time level through

$$\hat{\vec{U}} = \mathbf{E}\vec{U}^* \tag{2.76}$$

where $\mathbf{E}$ is the discrete Fourier transformation matrix defined as

$$\mathbf{E} = \frac{2}{2N_H+1}\begin{bmatrix} 1/2 & 1/2 & 1/2 & \cdots & 1/2 \\ \cos(\Omega t_1) & \cos(\Omega t_2) & \cos(\Omega t_3) & \cdots & \cos(\Omega t_{2N_H+1}) \\ \vdots & \vdots & \vdots & & \vdots \\ \cos(N_H\Omega t_1) & \cos(N_H\Omega t_2) & \cos(N_H\Omega t_3) & \cdots & \cos(N_H\Omega t_{2N_H+1}) \\ \sin(\Omega t_1) & \sin(\Omega t_2) & \sin(\Omega t_3) & \cdots & \sin(\Omega t_{2N_H+1}) \\ \vdots & \vdots & \vdots & & \vdots \\ \sin(N_H\Omega t_1) & \sin(N_H\Omega t_2) & \sin(N_H\Omega t_3) & \cdots & \sin(N_H\Omega t_{2N_H+1}) \end{bmatrix} \tag{2.77}$$

In a similar fashion, the conservative variables at the sub-time levels can be determined using the inverse discrete Fourier transform:

$$\vec{U}^* = \mathbf{E}^{-1}\hat{\vec{U}} \tag{2.78}$$

where $\mathbf{E}^{-1}$ is the inverse of the discrete Fourier transformation matrix given by

$$\mathbf{E}^{-1} = \begin{bmatrix} 1 & \cos(\Omega t_1) & \cdots & \cos(N_H\Omega t_1) & \sin(\Omega t_1) & \cdots & \sin(N_H\Omega t_1) \\ 1 & \cos(\Omega t_2) & \cdots & \cos(N_H\Omega t_2) & \sin(\Omega t_2) & \cdots & \sin(N_H\Omega t_2) \\ 1 & \cos(\Omega t_3) & \cdots & \cos(N_H\Omega t_3) & \sin(\Omega t_3) & \cdots & \sin(N_H\Omega t_3) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & \cos(\Omega t_{2N_H+1}) & \cdots & \cos(N_H\Omega t_{2N_H+1}) & \sin(\Omega t_{2N_H+1}) & \cdots & \sin(N_H\Omega t_{2N_H+1}) \end{bmatrix} \tag{2.79}$$

It is worth mentioning that the discrete Fourier transformation matrix, $\mathbf{E}$, and its inverse, $\mathbf{E}^{-1}$ are both square matrices with a size corresponding to $N_T \times N_T$ where $N_T = 2N_H + 1$ is the total number of sub-time levels.

In order to incorporate the harmonic balance method into the governing equations, let us first assume a volume-averaged representation of the conservation variables defined at the center of the control volume via

$$\vec{\overline{U}} = \frac{1}{\mathcal{V}} \int_{\mathcal{V}} \vec{U} \; d\mathcal{V} \tag{2.80}$$

where $\mathcal{V}$ is the control volume. Inserting Eq. (2.80) into Eq. (2.64) and dropping the bar notation for clarity gives

$$\frac{\partial(\mathcal{V}\vec{U})}{\partial t} + \oint_{\partial\mathcal{V}} \left[\vec{F}_c - \vec{F}_v\right] \; dS = \int_{\mathcal{V}} \vec{Q} \; d\mathcal{V} \tag{2.81}$$

The details on finite-volume discretization approach used in this work will be presented in Chapter 3. However, for the moment, let us assume that the convective and viscous fluxes as well as the volumetric source term are discretized and cast into a residual vector, $\vec{R}(\vec{U})$, as a function of the flow variables at all grid nodes. Therefore, the semi-discrete form of the governing equations shown in Eq. (2.81) can now be written as an ODE in terms of the conservation variables via

$$\frac{d(\mathcal{V}\vec{U})}{dt} + \vec{R}(\vec{U}) = 0 \tag{2.82}$$

As described earlier, in the HB method, the flow variables are computed at different sub-time levels that are equally-spaced over a single period. Therefore, Eq. (2.82) can be written for each conservation variable at each grid point in terms of the sub-time level solutions via

$$\frac{d(\mathcal{V}^*\vec{U}^*)}{dt} + \vec{R}(\vec{U}^*) = 0 \tag{2.83}$$

where $\mathcal{V}^*$ are the sub-time level control volumes and $\vec{R}(\vec{U}^*)$ includes the residuals consisting of convective, viscous as well as source terms at each sub-time level. Next, the time-derivative term in Eq. (2.83) can be approximated with a pseudo-spectral operator $\mathbf{D}$ given by

$$\frac{d}{dt} \approx \mathbf{D} = \frac{d\mathbf{E}^{-1}}{dt}\ \mathbf{E} \tag{2.84}$$

so that Eq. (2.83) is rewritten as

$$\mathbf{D}(\mathcal{V}^*\vec{U}^*) + \vec{R}(\vec{U}^*) = 0 \tag{2.85}$$

where the above equation is solved for all sub-time levels. It can be seen that the above equation is mathematically steady which is the most important feature of the HB method. In general, in order to march the governing equations to steady-state at each sub-time level, a "pseudo-time" derivative term is added to the time-spectral HB equation [Eq. (2.85)]. Therefore, almost all convergence acceleration techniques that are commonly used for steady cases can be applied to these equations to accelerate the convergence to time-periodic solutions. With the inclusion of a pseudo-time term, the harmonic balance equations become

$$\frac{d(\mathcal{V}^*\vec{U}^*)}{d\tau} + \mathbf{D}\left(\mathcal{V}^*\vec{U}^*\right) + \vec{R}(\vec{U}^*) = 0 \tag{2.86}$$

where τ is the pseudo-time. The first term in the equation above must vanish when marching the HB solution to convergence, thus recovering Eq. (2.85). As the sub-time level control volumes, $\mathcal{V}^*$, do not change in pseudo-time, they can be taken out of the pseudo-time derivative to get the final form of the HB-RANS-SA equations that read

$$\frac{d\vec{U}^*}{d\tau} + \frac{1}{\mathcal{V}^*}\left[\mathbf{D}\left(\mathcal{V}^*\vec{U}^*\right) + \vec{R}(\vec{U}^*)\right] = 0 \tag{2.87}$$

The details of the discretization and the numerical procedure used in UNPAC solver are described in the next chapter.

Chapter 3

UNstructured PArallel Compressible (UNPAC) Solver

In this work, a hybrid-grid solver is developed that solves the governing equations described in Chapter 2 using the Finite Volume Method (FVM). These equations are implemented in both two-dimensional and three-dimensional forms using proper vectorization of the primitive and grid variables such that based on the dimensionality of the input grid, the corresponding form of the governing equations is utilized. The UNPAC solver is written in Fortran programming language (standard Fortran 2003) and is also parallelized using the message passing interface (MPI) tools with a non-overlapping domain decomposition [236]. Additionally, ParMETIS software package [123] is used for parallel partitioning of the computational domain. In this chapter the numerical procedure and the details of the UNPAC solver are presented.

3.1 Non-Dimensionalization

A common practice in almost all numerical solvers is to non-dimensionalize the governing equations. By having all flow variables defined in a dimensionless form, consistency can be maintained for the entire set of governing equations independent of the units used for each individual variable. This allows one to use the angle of attack, the Mach number and the Reynolds number as the only input parameters for two-dimensional compressible flow solutions. Therefore, similarity parameters can be properly identified which reduce

the number of input parameters for each test case in the modeling and design process. Ultimately, by using a proper normalization, flow variables become of order of magnitude unity thus allowing for easier comparison with benchmark datasets [168, 105].

The non-dimensionalization process uses reference values for certain primary variables. This allows one to define secondary (or the rest of the) variables based on the reference quantities. In this work, reference length, reference pressure, and reference temperature values are specified. Additionally, the gas constant, R_{gas}, as well as the specific heat ratio, γ, for the working fluid are input. The primary (independent) and secondary (derived) variables are listed in Tables 3.1 and 3.2. In general, the reference pressure is taken to be the atmospheric pressure at sea level, i.e., 1 atm = 101325 N/m^2, and the reference temperature is set to 273 K. Also, for air used as the working fluid, the values of gas constant and the specific heat ratio are defined.

Table 3.1: Primary reference variables (user-specified).

variable name	UNPAC variable	value	units (SI)
Length	L_{ref} (input)	problem-specific	m
Pressure	p_{ref} (input)	101325.0	N/m^2
Temperature	T_{ref} (input)	273.0	K
Gas Constant	R_{gas} (input)	287.0	J/kg-K
Specific Heat Ratio	γ (input)	1.4	-

Table 3.2: Secondary (derived) reference variables.

variable name	UNPAC variable	units (SI)
Density	$\rho_{\text{ref}} = p_{\text{ref}}/(R_{\text{gas}} T_{\text{ref}})$	kg/m^3
Velocity	$V_{\text{ref}} = \sqrt{R_{\text{gas}}\ T_{\text{ref}}}$	m^2/s
Dynamic Viscosity	$\mu_{\text{ref}} = \rho_{\text{ref}}\ V_{\text{ref}}\ L_{\text{ref}}$	kg/m-s
Frequency	$\omega_{\text{ref}} = V_{\text{ref}}/L_{\text{ref}}$	1/s

Using the reference values defined in Tables 3.1 and 3.2, the dimensionless variables (denoted by overbars) are defined [105] as

$$\overline{x} = \frac{x}{L_{\text{ref}}}, \quad \overline{y} = \frac{y}{L_{\text{ref}}}, \quad \overline{z} = \frac{z}{L_{\text{ref}}}, \quad \overline{t} = \frac{t}{L_{\text{ref}}/V_{\text{ref}}} \tag{3.1}$$

$$\overline{\rho} = \frac{\rho}{\rho_{\text{ref}}}, \quad \overline{p} = \frac{p}{p_{\text{ref}}}, \quad \overline{\vec{v}} = \frac{\vec{v}}{V_{\text{ref}}} \tag{3.2}$$

$$\overline{\omega} = \frac{\omega}{\omega_{\text{ref}}}, \quad \overline{T} = \frac{T}{T_{\text{ref}}}, \quad \overline{\mu} = \frac{\mu}{\mu_{\text{ref}}} \tag{3.3}$$

It must be noted that substituting the above dimensionless variables into the equation of state (Eq. [2.19]) leads to a modified form of this equation that reads

$$\overline{p} = \overline{\rho}\,\overline{T} \tag{3.4}$$

which no longer includes the gas constant.

As described earlier, the non-dimensionalization is followed by a normalization process so that the primitive free-stream flow variables (density, velocity, pressure, and temperature) are of order of magnitude unity. In the UNPAC solver, the following non-dimensional free-stream flow variables are set to unity:

$$\overline{\rho}_\infty = 1.0, \quad \overline{V}_\infty = 1.0 \tag{3.5}$$

Therefore, based on the equation of state given in Eq. (3.4), the free-stream pressure, temperature, and the speed of sound are

$$\overline{p}_\infty = \frac{1}{\gamma\, M_\infty^2}, \quad \overline{T}_\infty = \frac{\overline{p}_\infty}{\overline{\rho}_\infty} = \overline{p}_\infty, \quad \overline{a}_\infty = \sqrt{\frac{\gamma \overline{p}_\infty}{\overline{\rho}_\infty}} = \frac{1}{M_\infty} \tag{3.6}$$

which means that for air at free-stream Mach numbers in the range of $0.3 < M_\infty < 0.9$ (as an example), the free-stream pressure and temperature are varied between $0.88 < \overline{p}_\infty$, $\overline{T}_\infty < 7.94$ and the free-stream speed of sound is in range $1.11 < \overline{a}_\infty < 3.33$ which are all in the order of $\mathcal{O}(1)$.

In the UNPAC solver, the free-stream Mach number is the main input variable and for viscous (laminar and turbulent) cases, the user also specifies the Reynolds number which is defined in terms of the reference variables as

$$\mathrm{Re} = \frac{\rho_{\mathrm{ref}} V_{\mathrm{ref}} L_{\mathrm{ref}}}{\mu_{\mathrm{ref}}} \tag{3.7}$$

Thus, in order to match the user-specified Reynolds number, the reference length, L_{ref}, is tuned automatically. Another important preliminary step for setting up the flow solver is the definition of the free-stream velocity vector which is used for flow initialization as well as in the far-field boundary. For two-dimensional cases, the two components of the velocity vector are defined based on the free-stream velocity magnitude, V_∞, and the angle of attack, α, such that

$$\vec{v}_\infty = [V_\infty \cos(\alpha), V_\infty \sin(\alpha)]^T \tag{3.8}$$

On the other hand, for three-dimensional cases, the free-stream velocity vector is provided as part of the case settings, i.e., $\vec{v}_\infty = [u_\infty, v_\infty, w_\infty]$ is input.

3.2 Data Structure and Grid Transparency

In the UNPAC solver, the integral form of the governing equations introduced in Chapter 2 are discretized and solved numerically. Using the *method of lines*, the temporal and spatial discretizations are performed separately which enables us to use different numerical approximations with various levels of accuracy in space and time independently.

3.2.1 Primal and Dual Grids

As discussed earlier, the discretization process starts by taking an average of the conservation variables over each control volume and writing according to the Taylor series expansion

$$\overline{U} = \frac{1}{\mathcal{V}_i} \left[\int_{\mathcal{V}_i} \vec{U} \; d\mathcal{V} - (\vec{\nabla}\overline{U}) \int_{\mathcal{V}_i} (\vec{x} - \vec{x}_i) \; d\mathcal{V} + \mathrm{H.O.T.} \right] \tag{3.9}$$

where $\mathcal{V}_i$ is the arbitrary control volume whose centroid is located at $\vec{x}_i$.

Therefore, this averaging process requires us to determine a location where the averaged quantities are defined at, which will shortly be discussed. Since grid generation is not in the

scope of this work, the unstructured grids are provided as an input to the UNPAC solver. The unstructured grid subdivides the physical domain into a number of grid cells or elements which will be called the "primal grid". In general, there are six types of elements that are recognized by the UNPAC solver which are all shown in Figure 3.1 for two-dimensional (2D) and three-dimensional (3D) cases.

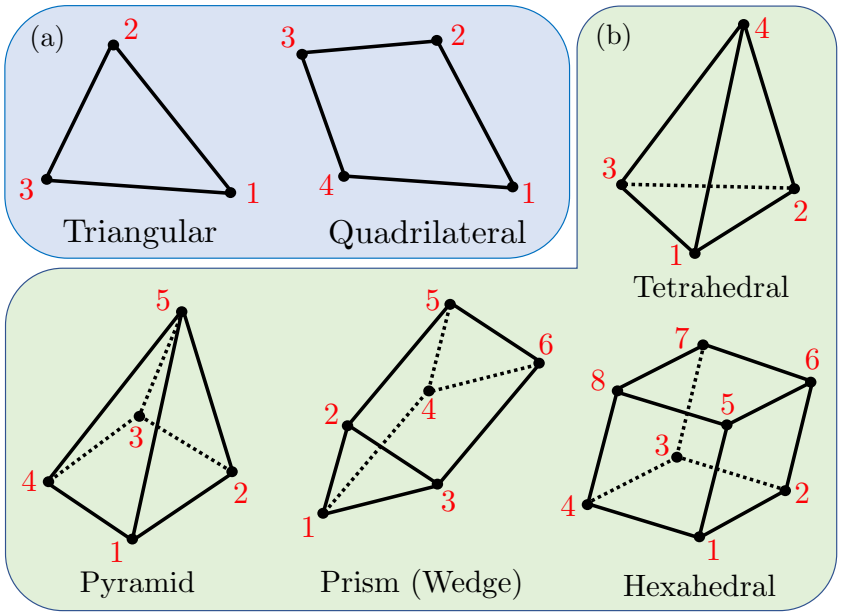


Figure 3.1: Different types of cells in the primal grid for the (a) 2D and (b) 3D cases

As can be seen in Figure 3.1, each cell type has different number of nodes (vertices), edges, and faces. Each edge is associated with only two vertices and for 2D cells, the faces of the grid cell are the same as the grid edges.

To motivate the finite volume discretization of the governing equations (in the integral form), the control volumes and the location at which the averaged state variables are stored need to be defined. The definition of the control volumes results in what is called the "dual grid". In the framework of an unstructured solver, there are two available choices:

1. **Cell-centered approach**: In this approach, the control volumes are the same as the grid cells such that the "primal grid" and the "dual grid" are identical. Also the conservation variables (volume averages in Eq. [3.9]) will be located at the centroid of the grid cells shown in Figure 3.1.

2. **Cell-vertex approach**: Here, the conservation variables are stored at the grid vertices and the control volumes are defined by connecting the edge medians and the cell centroids of all edges and cells connected to each grid vertex. This type of control volume is called "*median-dual*" which results in a dual grid that obviously does not match the primal grid as shown in Figure 3.2.

It must be noted that for structured grids, there is another type of cell-vertex approach that results in "overlapping" control volumes but since this is not pursued in unstructured grid topologies, it will not be addressed here [21].

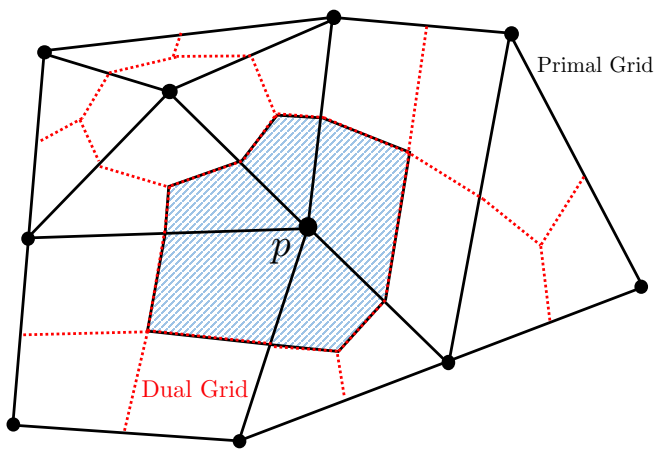


Figure 3.2: Primal (solid black) and dual grids (dashed red). The median-dual control volume associated with grid vertex p is shaded.

There is a lot of debate in the CFD community over the choice of cell-centered or cell-vertex approaches. Each approach has its own advantages and disadvantages compared to the other and the interested reader is referred to Ref. [21] for an in-depth discussion. The goal of this work is to have a grid-transparent discretization scheme and for the reasons that become obvious in the next few sections, the median-dual control volume approach is used in the UNPAC solver. However, it must be noted that in the cell-vertex approach used herein, since the flow variables are stored at grid vertices, which is not necessarily at the centroid of the control volume, extra care must be given to the volume averaging given by Eq. (3.9).

According to Eq. (3.9), due to the mismatch between the centroid of the control volume, $\vec{x}_i$, and the grid vertex, $\vec{x}$, at which the flow variables are stored, second-order errors are

introduced [133]. However, according to Leonard [133], this is only an issue for high-order (third-order or higher) schemes. On the other hand, this mismatch also necessitates the coupling of the flow variables at any control volume to those of its neighboring cells through a mass matrix. However, for steady flows (or mathematically steady as in the case of HB method), this mass matrix can be replaced by an identity matrix (lumped) without sacrificing accuracy [93, 220]. It must be added that the median-dual definition used in this work arises naturally in Galerkin-type FEM approach when applied to the triangular elements with linear shape function [93]. Hence, the present solver gives accuracy comparable to those of Galerkin-type FEM solvers on triangular and tetrahedral elements.

3.2.2 Data Structure and Classes

For structured grids, the cell, face, edge, and the node connectivity data is pretty much straightforward and neighborhoods can be simply identified by increasing and decreasing indices. For unstructured grids, on the other hand, the connectivity data is non-trivial and an elaborate data structure is required. This data structure must match the discretization method while containing necessary information for pre- and post-processing stages.

For the reasons that become clear in the next section, an edge-based data structure is used in this work. Therefore, the goal of the data structure is to use cellular, facial, and nodal data and associate them with the edge information. This is due to the fact that in the median-dual control volume approach, the edges of the primal and dual grids match. In the FVM method used in this work, it is required to calculate fluxes across the faces (control surfaces) of the control volume. Thus, by associating the edges of the dual grid (faces of the median-dual control volume) to the edges of the primal grid, all fluxes can be calculated by simply looping over the edges of the primal grid.

The data structure of the UNPAC solver uses the object-oriented programming capabilities of **Fortran 2003** standard, which in turn allows the use of special classes with their corresponding components (attributes) and modules (methods). Three main classes are defined in the UNPAC data structure that handle cells, edges, and nodes. While a separate class can be used for the faces, due to the fact that in 2D cases edges are treated the same

as the faces of the control volume, extra components and methods are added to the cell class for handling face data only used for 3D cases.

First, the **node** class is described which contains the nodal locations in the Cartesian coordinate system. Since control volumes are associated with the grid nodes, the **node** class also stores the volume of the median-dual cell. Finally, the node-node and node-cell connectivity data as well as the node-edge associations are stored in this class.

Next, a **cell** class is defined in the data structure which contains all the cellular information (as well as the facial information for the 3D cases). This class is parameterized based on the dimensionality of the provided grid such that for 2D and 3D cases the components of the cell class are pre-allocated to their maximum allowable sizes. For example, in 2D cases, each cell has a maximum of 4 nodes and 4 edges/faces that happens in the case of a quadrilateral element (see Figure 3.1). Also, each face of the element contains 2 nodes (same for an edge) and each node inside the cell has only 2 node neighbors that belong to that same element. On the other hand, for 3D cases, each cell has a maximum of 8 nodes, 6 faces, and 12 edges that all happen in a hexahedral cell. Moreover, each face has a maximum of 4 nodes (quadrilateral faces in pyramid, prism, and hexahedral elements), each edge has 2 nodes (true for all edges), and each node has a maximum of 4 node neighbors connected to it by an edge (happens only for the top node in the pyramid cell).

Another important feature of the **cell** class is the local node, edge, and face indices. In general, nodes and edges have global indices that go from 1 to the total number of nodes (*nNodes*) and edges (*nEdges*) (same for primal and dual grids), respectively. However, inside any given cell, local indices are used to ease up the grid pre-processing. These local indices are shown as an example for triangular and tetrahedral elements of the primal grid in Figure 3.3. A similar approach is used for all element types (6 in total) for 2D and 3D cases.

An important issue in the case of the local indexing is the cell orientation. According to VTK [8] and CGNS [132] standards, nodes are arranged according to the indices shown in Figure 3.1. Therefore, in the definition of the facial node lists, a right-hand-rule is utilized to make sure that the nodes are in the counter-clockwise order so that the face normals are all pointing outward from the element perspective.

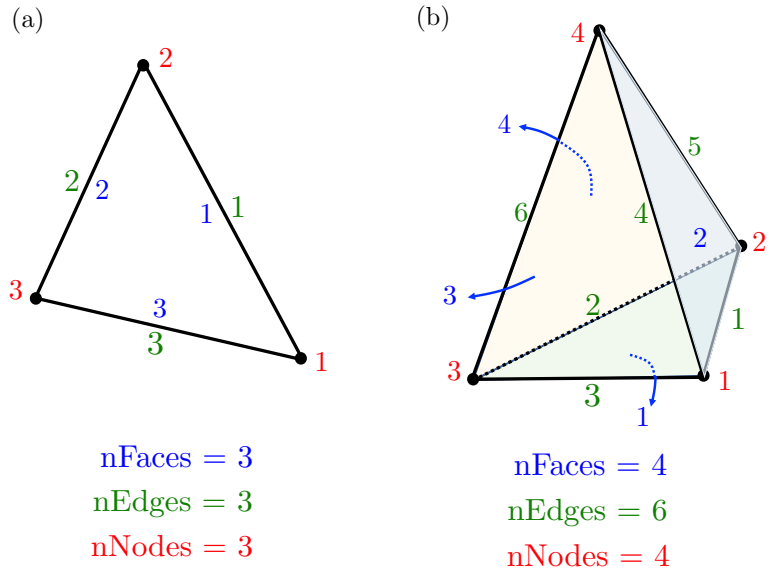


Figure 3.3: Local indexing for the (a) triangular and (b) tetrahedral elements.

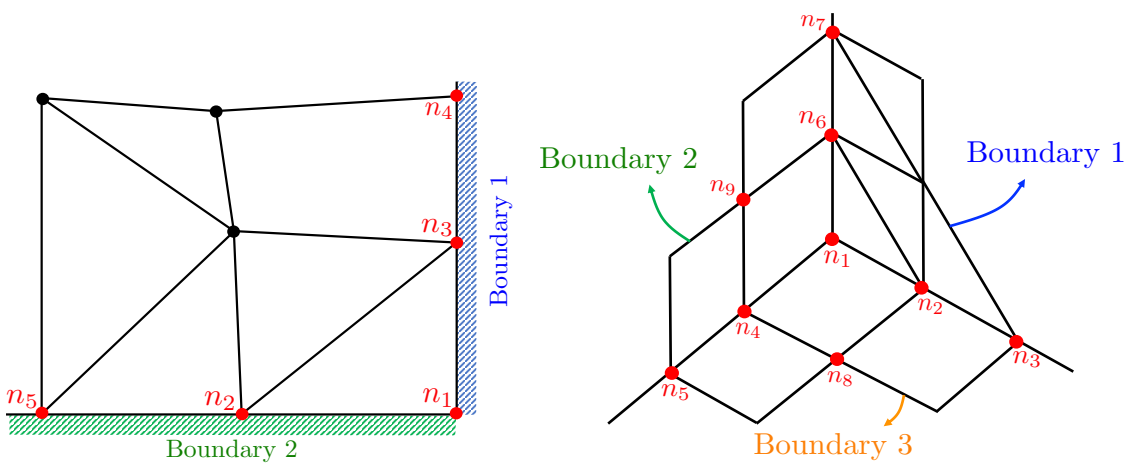
While the VTK/CGNS nodal conventions are followed in all grid generation solver packages, sometimes cells are exported in a reversed order due to the complex meshing and block-merging processes. Therefore, a cell reorientation procedure is implemented in UNPAC that first checks the orientation of the grid cells according to the vector calculus and, if required, reorients the cells to their standard form.

Another issue is the mesh entanglement or cell inversion that can happen during the grid adaptation (AMR) as well as during mesh deformation. For a cell that undergoes large deformations, sometimes one of the nodes passes through an opposing face that leads to an inverted cell. To identify these cells, the cell orientation checks are performed after each AMR or mesh motion cycle and since the orientation of the elements have been checked in the pre-processing step, any cell that requires reorientation is obviously inverted. This method is useful for identifying invalid meshes which will be discussed later.

Boundary Considerations

As mentioned earlier, an edge-based approach is used throughout the solution process. Therefore, extra attention must be given to the boundary conditions and their treatment in the numerical solver. Although the discretized boundary conditions are presented in Section

3.9 in detail, data structure considerations for the boundaries need to be addressed here. For 2D problems, boundaries consist of edges, while for 3D problems, they consist of faces. As shown in Figure 3.4, in some cases two or more boundaries can intersect with each other. For 2D cases, a node-based boundary treatment can create issues since a node can be shared by two boundary edges. As an alternative, an edge-based approach needs to be followed which requires the addition of the boundary edge normal information to the data structure. This issue becomes more apparent for 3D problems, where not only the nodes, but also the edges can be shared by more than one boundary. Therefore, a face-based boundary treatment would be necessary for 3D cases, which requires having the boundary face normal information evaluated at the pre-processing stage.



(a) 2D intersection **(b)** 3D intersections

Figure 3.4: Boundary intersections. For the 2D case (a) node n_1 is shared between boundaries 1 and 2. For the 3D case (b) edge n_1n_2 is shared between boundaries 1 and 3.

As discussed previously for the cell orientation process, a similar approach will be used for boundary faces (or boundary edges in 2D cases) to fix the local ordering of the nodes for each face in a way that face normals are always pointing outward (exiting the computational domain).

3.2.3 Grid Transparency

As discussed earlier, the goal of this work is to develop a robust solver that can handle mixed unstructured grids where different cell types can be used. The goal here is to have a solution method that is independent of the various cell types involved in the primal grid. Obviously, different cell types should be treated differently and separately during the pre-processing as well as post-processing stages of the numerical solver. However, during the discretization and solution processes, one wants to avoid the necessity of having different or separate procedures according to the various cell types. Otherwise, auxiliary arrays must be included in the data structure and the solution process will involve many conditional statements that can negatively affect the performance of the solver and the readability of the computer code.

Ideally, it is desired to have a solver with a concise data structure that can handle the solution process independent of the cell type, hence the term "*grid transparent*" [93]. In order to achieve this goal, attention must be shifted away from cellular data and focused on the edge and vertex information. This is the main reason for using an edge-based data structure in this work.

Another important feature of the UNPAC solver is its dimensional flexibility such that it can handle both 2D and 3D grids without having many special instruction or subroutines to distinguish between the two. This makes the "grid transparent" approach used in this work even more attractive. In fact, edges are always defined by two vertices independent of the problem dimensionality and therefore, an edge-based data structure can lead us to an ultimately grid transparent solver. In the framework of the UNPAC solver, different cell types or even mixed grids are handled identically during the solution process.

It must be noted that polyhedral elements and grids of arbitrary topology which have been used primarily in the Finite-Element (FE) solvers, are becoming more popular in the FVM-based CFD solvers [162]. Therefore, by having a grid transparent solver, addition of the arbitrary grid topologies only requires modifications to the pre- and post-processing stages without altering the computational core of the CFD solver.

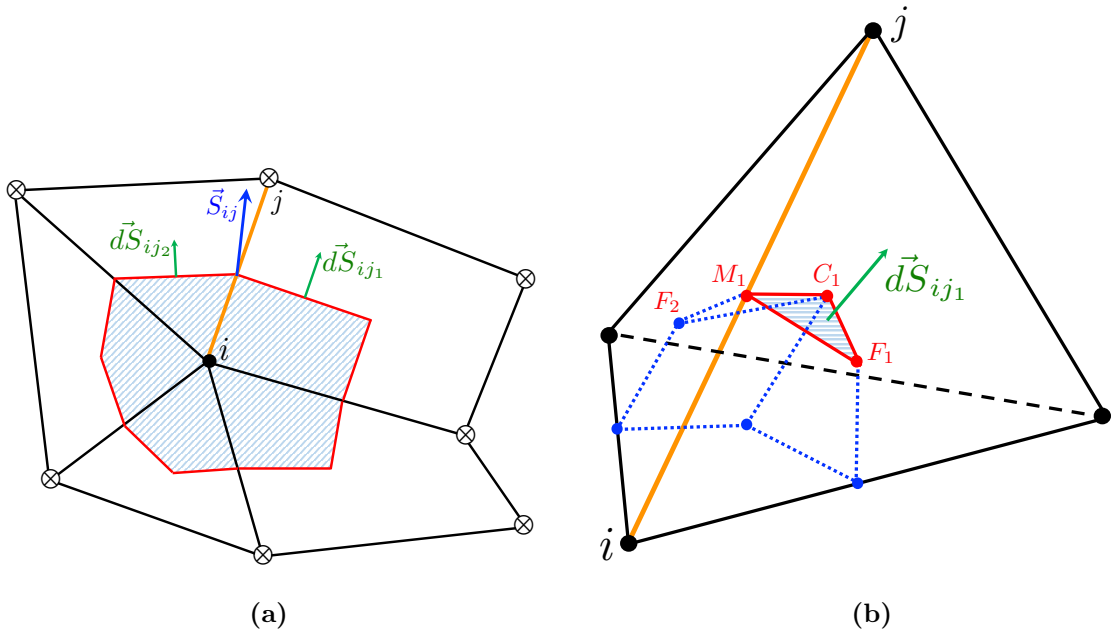


Figure 3.5: Fractional face vectors used in the calculation of the total edge vector, $\vec{S}_{ij}$, for (a) the median-dual control volume of node i in a 2D mixed grid, and (b) a fraction of the median-dual control volume of node i in a 3D tetrahedral element sharing node i. (Face vectors are not drawn to scale)

Considering the cell-vertex median-dual control volume approach used in this work, fluxes need to be integrated over the control surfaces for each cell. As discussed earlier, these control surfaces are associated with the edges of the primal grid. The median-dual control volume is defined by connecting the cell and face centroids and the edge medians of all the cells, faces, and edges sharing each vertex. This is shown in Figure 3.5a for a mixed grid in 2D. As can be seen, an arbitrary edge is associated with two fractional face vectors, $d\vec{S}_{ij_1}$ and $d\vec{S}_{ij_2}$, that share the same edge. Therefore, by collecting these two fractional face vectors, the total face vector of this edge, $\vec{S}_{ij}$, can be determined via

$$\vec{S}_{ij} = d\vec{S}_{ij_1} + d\vec{S}_{ij_2} \tag{3.10}$$

For a second order scheme, the conservation variables are assumed to be constant for each face and the fluxes (both convective and viscous) are evaluated at the edge median. Therefore, the edge vector, $\vec{S}_{ij}$, that includes both the area and the normal direction of the control surface is used for integration. For the sake of improving performance and to avoid

extra computations, edges are defined such that they go from node i to node j and the $\vec{S}_{ij}$ vector is aligned with the same direction. Therefore, the edge vector $\vec{S}_{ij}$ is used for the integration at node i and $-\vec{S}_{ij}$ is used for the integration at node j.

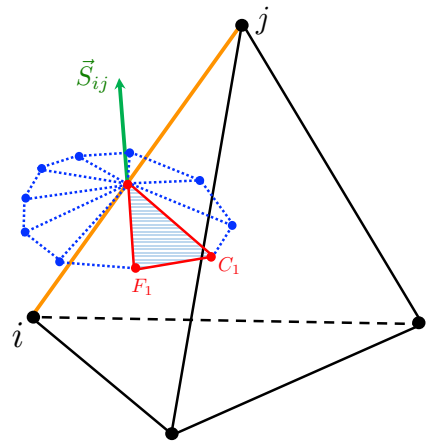


Figure 3.6: Total face vector, $\vec{S}_{ij}$, for an arbitrary edge ij in 3D case using the median-dual approach.

Obviously, for 2D cases, each edge is only shared by two cells on each side although at the boundary edges, only one cell (and hence one fractional face vector) is contributing to the edge vector, $\vec{S}_{ij}$. However, in 3D cases, this process is more complicated as each edge can be shared by many cells. Therefore, in these instances, fractional face vectors should be calculated one at a time for each edge-face-cell combination. This is shown in Figure 3.5b for a tetrahedral element where local node indices are used for clarity. Here, the fractional face vector, $d\vec{S}_{ij_1}$, is considered which is constructed by connecting the edge median, M_1, the face centroid, F_1, and the cell centroid, C_1. This process is repeated for all edge-face-cell combinations sharing the same edge and the edge vector, $\vec{S}_{ij}$, will be the sum of all these fractional face vectors as shown in Figure 3.6.

3.3 Convective Fluxes

As described in Chapter 2, fluxes in the Navier-Stokes equations consist of convective and viscous parts. In the absence of viscous effects (inviscid flows), the convective flux is the only flux term in the governing equations (now called the "Euler" equations). Convective

fluxes are also the main source of non-linearity in the equations governing the fluid flow. Therefore, extra attention has been given to their discretization and various schemes have been introduced to improve accuracy of their evaluation.

Traditionally, central schemes based on simple arithmetic averaging of the conservation variables have been used in the framework of Euler and Navier-Stokes solvers. To eliminate the odd-even decoupling phenomenon, as well as stability and shock handling issues associated with the central schemes, artificial dissipation terms have been added with the JST scheme, named after Jameson-Schmith-Turkel [117].

While computationally cheap, central schemes fail to offer high resolution shock and boundary layer capturing. As an alternative, upwind schemes have been developed which, compared to the arithmetic averaging in central schemes, use a biased averaging of the flow variables. Upwind schemes are generally categorized into two classes of (1) flux-vector splitting and (2) flux-difference splitting schemes. In the former approach, only the direction of the wave propagation is accounted for with the Van Leer's scheme [214] and the Jameson's convective upwind split pressure (CUSP) scheme [113] being among the most widely used techniques.

In contrast, the flux-difference splitting schemes consider both the direction and the magnitudes of the propagating waves. Initially proposed by Godunov [79], the idea of the flux-difference splitting scheme is to solve the Riemann shock tube problem by considering the left and right states of the solution. More than a decade later, Philip Roe [177] and Stanely Osher [166] introduced approximate Riemann solver techniques which provide comparable accuracy at a fraction of the cost of the exact Riemann solvers.

Due to its popularity, the Roe upwind scheme [177] is used in this work. While details of the numerical scheme can be found in references [21, 98], for completeness, the discretization process is presented next.

3.3.1 Flux-Difference Splitting (Roe Scheme)

Let us consider the 1D Euler equations written in strong conservation form as a non-linear hyperbolic PDE via

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}(\vec{U})}{\partial x} = 0 \tag{3.11}$$

where $\vec{U}$ is the vector of conservation variables and $\vec{F}$ is the convective flux vector. The above equation can be written in the quasi-linear form using the chain rule of differentiation such that

$$\frac{\partial \vec{U}}{\partial t} + \mathbf{A}(\vec{U})\frac{\partial \vec{U}}{\partial x} = 0 \tag{3.12}$$

where $\mathbf{A}(\vec{U})$ is the Jacobian of the flux vector, i.e., $\mathbf{A}(\vec{U}) = \frac{\partial \vec{F}(\vec{U})}{\partial \vec{U}}$. Applying the FVM to Eq. (3.12) requires having the Jacobian defined at the face of the control volume between the two states, $\vec{U}_i$ and $\vec{U}_j$. The main idea of the Roe scheme is to find matrix $\tilde{\mathbf{A}}_{\text{Roe}}(\vec{U}_L, \vec{U}_R)$, also known as *Roe matrix*, by decomposing the flux vectors into left and right states such that matrix $\tilde{\mathbf{A}}_{\text{Roe}}$ would be constant between the two states. Therefore, Eq. (3.12) can be solved as a truly linear hyperbolic PDE for which the exact solution can be easily found. This idea can be easily extended to 3D cases with the exact solutions (also known as Roe averages) given as [177, 21]

$$\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R} \\
\tilde{u} &= \frac{1}{\sqrt{\rho_L} + \sqrt{\rho_R}} \left(u_L\sqrt{\rho_L} + u_R\sqrt{\rho_R}\right) \\
\tilde{v} &= \frac{1}{\sqrt{\rho_L} + \sqrt{\rho_R}} \left(v_L\sqrt{\rho_L} + v_R\sqrt{\rho_R}\right) \\
\tilde{w} &= \frac{1}{\sqrt{\rho_L} + \sqrt{\rho_R}} \left(w_L\sqrt{\rho_L} + w_R\sqrt{\rho_R}\right) \\
\tilde{H} &= \frac{1}{\sqrt{\rho_L} + \sqrt{\rho_R}} \left(H_L\sqrt{\rho_L} + H_R\sqrt{\rho_R}\right)
\end{aligned} \tag{3.13}$$

and

$$\begin{aligned}
\tilde{c} &= \sqrt{(\gamma - 1)(\tilde{H} - \tilde{q}^2/2)} \\
\tilde{V} &= \vec{\tilde{v}} \cdot \vec{n} \\
\tilde{q} &= \tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2
\end{aligned} \tag{3.14}$$

where left (L) and right (R) states will be defined later. Thus, the convective fluxes at the face of the control volume located at the midpoint of the edge ij can be written as

$$(\vec{F}_c)_{ij} = \frac{1}{2}\left[\vec{F}_c(\vec{U}_R) + \vec{F}_c(\vec{U}_L) - |\tilde{\mathbf{A}}_{\mathrm{Roe}}|_{ij}(\vec{U}_R - \vec{U}_L)\right] \tag{3.15}$$

As can be seen in Eq. (3.15), the convective flux for edge ij consists of a flux-averaged term and a Roe-difference (upwind dissipation due to the Roe flux splitting) term where the latter can be also decomposed to three terms based on the left traveling acoustic wave ($_{la}$), convective waves ($_{conv}$), and the right traveling acoustic wave ($_{ra}$) as

$$|\tilde{\mathbf{A}}_{\mathrm{Roe}}|(\vec{U}_R - \vec{U}_L) = |\Delta\vec{F}_{\mathrm{la}}| + |\Delta\vec{F}_{\mathrm{conv}}| + |\Delta\vec{F}_{\mathrm{ra}}| \tag{3.16}$$

where

$$|\Delta\vec{F}_{\mathrm{la}}| = |\tilde{V} - \tilde{c}|\left(\frac{\Delta p - \tilde{\rho}\tilde{c}\Delta V}{2\tilde{c}^2}\right)\begin{bmatrix} 1 \\ \tilde{u} - \tilde{c}n_x \\ \tilde{v} - \tilde{c}n_y \\ \tilde{w} - \tilde{c}n_z \\ \tilde{H} - \tilde{c}\tilde{V} \end{bmatrix} \tag{3.17}$$

$$\begin{aligned} |\Delta\vec{F}_{\mathrm{conv}}| = |\tilde{V}|\Bigg\{ & \left(\Delta p - \frac{\Delta\rho}{\tilde{c}^2}\right)\begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \tilde{q}^2/2 \end{bmatrix} \\ & +\tilde{\rho}\begin{bmatrix} 0 \\ \Delta u - \Delta V n_x \\ \Delta v - \Delta V n_y \\ \Delta w - \Delta V n_z \\ \tilde{u}\Delta u + \tilde{v}\Delta v + \tilde{w}\Delta w - \tilde{V}\Delta V \end{bmatrix}\Bigg\} \end{aligned} \tag{3.18}$$

$$|\Delta\vec{F}_{\mathrm{ra}}| = |\tilde{V} + \tilde{c}| \left(\frac{\Delta p + \tilde{\rho}\tilde{c}\Delta V}{2\tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{u} + \tilde{c}n_x \\ \tilde{v} + \tilde{c}n_y \\ \tilde{w} + \tilde{c}n_z \\ \tilde{H} + \tilde{c}\tilde{V} \end{bmatrix} \tag{3.19}$$

in which the difference $\Delta(\blacksquare) = (\blacksquare)_R - (\blacksquare)_L$ defines the jump condition and averaged quantities are based on Eq. (3.13) and Eq. (3.14).

As can be seen, Eqs. (3.17), (3.18), and (3.19) are scaled by the modulus of the eigenvalues associated with the left and right traveling acoustic waves, $|\tilde{V} \pm \tilde{c}|$, and that of the convective wave, $|\tilde{V}|$, where $\tilde{V}$ is the contravariant velocity defined in terms of the Roe averages (see Eq. [3.14]).

A common issue with the original Roe scheme is associated with the stationary expansion ($\tilde{c} \to 0$) which can be caused in the cases where the left and right flux vectors are identical, i.e., $\vec{F}_c(\vec{U}_L) = \vec{F}_c(\vec{U}_R)$, but the left and right flow variables are not, i.e., $\vec{U}_L \neq \vec{U}_R$. To remedy these issues, Harten, Lax and Van Leer [90] have proposed a modification to the acoustic wave modules, known as the Harten's entropy correction, such that

$$|\Lambda_c| = |\tilde{V} \pm \tilde{c}| = \begin{cases} |\Lambda_c| & |\Lambda_c| > \delta \\ \frac{{\Lambda_c}^2 + \delta^2}{2\delta} & |\Lambda_c| \leq \delta \end{cases} \tag{3.20}$$

where δ is a small value taken as 0.05 in this work [21]. A similar problem can happen at stagnation points where the convective wave modulus, $|\tilde{V}|$, goes to zero which can be avoided in a similar fashion.

Now the only remaining part is to determine the left and right states of the flow variables, i.e., $\vec{U}_L$ and $\vec{U}_R$. In the edge-based approach used in this work, the flux integration involves looping over edges of the primal grid. For an arbitrary edge ij, the flow solutions are defined and stored at the two end nodes, i.e., $\vec{U}_i$ and $\vec{U}_j$. The process of calculating the left and right states, $\vec{U}_L$ and $\vec{U}_R$, based on the solutions at the end nodes of the edge, $\vec{U}_i$ and $\vec{U}_j$, is called "solution reconstruction" and two approaches are implemented in UNPAC solver which will be discussed next.

3.3.2 First-Order Reconstruction

The solution reconstruction process helps us to determine the left and right states at the midpoint of the edge as shown in Figure 3.7. In the first approach, the solution is assumed to remain constant along the half-edge such that the left and right states can be simply defined by the end node solutions, i.e.,

$$\vec{U}_L = \vec{U}_i$$

$$\vec{U}_R = \vec{U}_j$$

This approximation, leads to a first-order spatial discretization which can lead to excessive diffusion in viscous problems as well as smeared shocks in general. Thus, a higher order approximation scheme is required which will be pursued next.

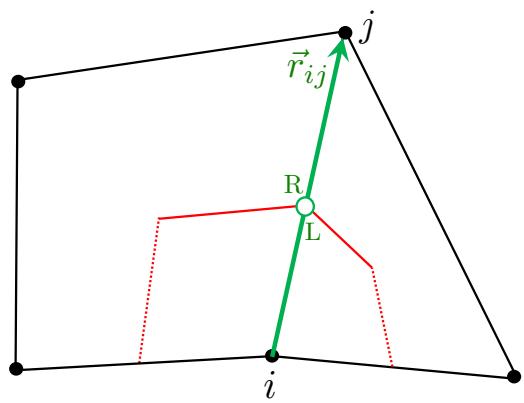


Figure 3.7: Solution reconstruction at the midpoint of the edge ij (shown with a hollow circle). $\vec{r}_{ij}$ is the vector connecting node i to node j

3.3.3 Second-Order Reconstruction

In order to motivate the second order Roe scheme, a piecewise linear reconstruction is used which assumes that solution varies linearly between node i (or node j) and the edge midpoint (flux location). This approach was initially proposed by Barth and Jespersen [13] and follows the exact same procedure involved in the Galerkin-type FEM discretization on the linear

elements. Using a piecewise linear reconstruction of the solution along edge ij (as shown in Figure 3.7), left and right states can be defined based on the solutions and their gradients (slopes) at the two end nodes, i.e.,

$$\vec{U}_L = \vec{U}_i + \frac{1}{2}\phi_i(\vec{\nabla}\vec{U}_i \cdot \vec{r}_{ij})$$

$$\vec{U}_R = \vec{U}_j - \frac{1}{2}\phi_j(\vec{\nabla}\vec{U}_j \cdot \vec{r}_{ij})$$

where $\vec{\nabla}\vec{U}_i$ and $\vec{\nabla}\vec{U}_j$ are the vectors of solution gradients at nodes i and j, respectively, and ϕ_i and ϕ_j are the limiter functions defined at these points. Gradient calculation and the definition of the limiter functions will be discussed later in this chapter. The vector $\vec{r}_{ij}$ connects node i to node j and since the reconstruction is required at the midpoint of the edge, a one-half factor is included. Also, due to the direction of the $\vec{r}_{ij}$ vector (pointing from i toward j), the right state reconstruction requires a negative sign. According to the Taylor series expansion, the piecewise linear reconstruction leads to a second order spatial discretization [2]. Barth and Frederickson [12] and Barth [11] have shown that the same idea can be extended to n-order methods using a reconstruction based on polynomials of degree n. As an example, a piecewise quadratic reconstruction would require solution Hessians (second derivatives of the solution) to give a third-order Roe scheme. However, higher order schemes are not in the scope of this work.

It must be noted that the second-order Row scheme comes at a price of increased computational demand specially due to the cost of gradient calculations at each solution stage. Furthermore, the second-order scheme can lead to a slower convergence rate for the numerical solver. An interesting remedy would be to use the first order scheme to achieve a certain level of accuracy before switching to the second order scheme and using the previous (low resolution) solution as the initial condition to speedup the solution process.

3.3.4 Limiter Function

As discussed earlier, artificial dissipation terms are necessary in the central schemes to provide stability around shocks and discontinuities (regions with large flow gradients) while

offering higher resolution in the smooth regions. In the framework of upwind schemes, limiter functions offer a similar mechanism to guarantee solution stability and accuracy.

As shown in the previous section, for the second-order Roe scheme, the limiter functions limit the solution reconstruction. This is necessary for achieving a "monotonicity preserving" scheme around the shocks where large gradients can deteriorate the solution accuracy, create non-physical oscillations, and jeopardize the stability of the method [21]. In smooth regions, the idea is to have unlimited reconstruction to provide higher resolution and better accuracy. In this work, the limiter function of Venkatakrishnan [218, 219] is used which provides enough dissipation around the shock and regions of discontinuity to preserve resolution and stability while offering a very good convergence rate for the numerical solver. The Venkatakrishnan's limiter function at node i is defined by [21]

$$\phi_i = \min_j \begin{cases} \frac{1}{\Delta_2}\left[\frac{(\Delta_{1,\max}^2+\epsilon^2)\Delta_2+2\Delta_2^2\Delta_{1,\max}}{\Delta_{1,\max}^2+2\Delta_2^2+\Delta_{1,\max}\Delta_2+\epsilon^2}\right] & \Delta_2 > 0 \\ \frac{1}{\Delta_2}\left[\frac{(\Delta_{1,\min}^2+\epsilon^2)\Delta_2+2\Delta_2^2\Delta_{1,\min}}{\Delta_{1,\min}^2+2\Delta_2^2+\Delta_{1,\in}\Delta_2+\epsilon^2}\right] & \Delta_2 < 0 \\ 1 & \Delta_2 = 0 \end{cases} \tag{3.21}$$

where

$$\begin{aligned} \Delta_2 &= \frac{1}{2}\left(\vec{\nabla}\vec{U}\cdot\vec{r}_{ij}\right) \\ \vec{U}_{\max} &= \max(\vec{U}_i, \max_j \vec{U}_j) \\ \vec{U}_{\min} &= \min(\vec{U}_i, \min_j \vec{U}_j) \\ \Delta_{1,\max} &= \vec{U}_{\max} - \vec{U}_i \\ \Delta_{1,\min} &= \vec{U}_{\min} - \vec{U}_i \end{aligned} \tag{3.22}$$

A similar function can be defined at node j. The calculated limiter function will range between 0 for the fully limited case (no reconstruction hence first order) and values close to 1 for the unlimited case (linearly reconstructed hence second order). As can be seen in Eq. (3.21) and Eq. (3.22), this limiter function requires the calculation of minimum and maximum values of the conservation variables at all distance-one neighbors for each node.

This is done via an initialization step before calculating the final limiter functions. The goal of evaluating the min and max values is to enforce monotonicity which requires that:

1. local maxima does not increase,
2. local minima does not decrease,
3. no new local extrema is introduced in the flow field [21].

It is apparent that the limiter function is controlled by the parameter, ϵ^2, defined as

$$\epsilon^2 = (KL_{\text{local}})^3$$

where L_{local} is the local length scale and K is the coefficient that controls the limiter with $K = 0$ corresponding to the fully limited (first order scheme) and $K >>$ corresponding to the unlimited (full reconstruction) settings. It must be noted that the computational and the memory cost of the Venkatakrishnan's limiter is relatively high due to the recalculation and storage of the local minimum and maximum values.

3.4 Gradient Calculation

As demonstrated in the previous section, the second order Roe scheme relies on the availability of the gradient information for all conservation variables to perform the solution reconstruction. Additionally, the velocity and temperature gradients are required in the calculation of the viscous fluxes.

In this work, the Green-Gauss method is used for the calculation of the gradients. According to the Green-Gauss theorem, the volume integral of the gradient of a scalar U is equal to the surface integral of that same scalar function such that

$$\int_{\mathcal{V}} \vec{\nabla} U \; d\mathcal{V} = \oint_{\partial\mathcal{V}} U \cdot \vec{n} \; d\mathcal{V} \tag{3.23}$$

Assuming that the gradient is constant over the control volume, Eq. (3.23) can be approximated as

$$\vec{\nabla} U = \frac{1}{\mathcal{V}} \oint_{\partial \mathcal{V}} U \cdot \vec{n} \, d\mathcal{V} \tag{3.24}$$

As can be seen, the Green-Gauss gradient is, in essence, very similar to the flux term of the conservation laws discussed in the previous chapter. This similarity makes it possible to use the same edge-based approach and rewrite Eq. (3.24) as a summation of the surface fluxes over all edges connected to node i such that for the gradient of the scalar function U_i at the control volume $\mathcal{V}_i$ we have

$$\vec{\nabla} U_i \approx \frac{1}{\mathcal{V}} \sum_{j=1}^{\mathrm{Ngb}_i} U_{ij} \cdot \vec{S}_{ij} \tag{3.25}$$

where $\vec{S}_{ij}$ is the edge vector for edge ij and Ngb_i is the number of node neighbors at distance-one (direct neighbors) of the node i (same as the number of edges connected to node i). The face value U_{ij} can be simply defined as the arithmetic average of the scalar function at the two end points, i.e.,

$$U_{ij} = \overline{U}_{ij} = \frac{1}{2} \left(U_i + U_j \right) \tag{3.26}$$

In the UNPAC solver, the gradients of the primitive variables (ρ, $\vec{v}$, p, h, $\tilde{\nu}$) are calculated according to Eq. (3.25) and stored at the same location as the flow variables, i.e., the primal/dual grid nodes. The Green-Gauss gradient approximation approach discussed herein is reported to run into some accuracy issues for the mixed grid [92]. However, validation and verification tests performed in this work have led to very good agreements with the literature in the case of mixed grids. Nonetheless, there are two approaches that can improve the accuracy of the gradient approximation which will be discussed next.

In the first approach, Haselbacher and Blazek [92] and Blazek [21] have suggested the use of mixed set of distance-one and distance-two neighbors of node i to calculate the summation term in Eq. (3.25) (these nodes are marked with a $\bigotimes$ sign in Figure 3.5a). However, this approach requires an additional data structure to include faces of the cells sharing the node i. As a result, the approach would no longer be "grid transparent" since cell information would be required [21].

As an alternative, the least-squares approach as described by Haselbacher and Blazek [92] and Blazek [21] can be used. However, this method also leads to the introduction of virtual edges at the boundaries in the case of prism and/or hexahedral elements and therefore, is not pursued here.

3.5 Viscous Fluxes

In this work, a central scheme is used for the calculation of the viscous fluxes. These flux vectors require the solution and gradient information to be approximated at the face centers. Due to the elliptic nature of the viscous fluxes, the simple arithmetic averaging would suffice for a second-order scheme. Therefore, the velocity vector and the effective viscosity ($\mu_{\text{eff}} = \mu_L + \mu_T$ for RANS solver or simply μ_L for the laminar cases) are averaged at the midpoint of each edge. Additionally, the gradient information needs to be averaged so that the viscous stresses can be evaluated. Ultimately, the viscous flux on edge ij is defined by:

$$\vec{F}_{v_{ij}} = \vec{F}_v(\vec{U}_{ij}, \vec{\nabla}\vec{U}_{ij}); \quad \vec{U}_{ij} = \overline{\vec{U}}_{ij} = \frac{1}{2}\left(\vec{U}_i + \vec{U}_j\right) \tag{3.27}$$

where $\vec{U}_{ij}$ and $\vec{\nabla}\vec{U}_{ij}$ are the flow variables and their gradients at the face center (midpoint of edge ij). Now, the gradients at the face centers need to be averaged which will be discussed next.

3.5.1 Gradient Averaging

Clearly, the first choice for evaluating the gradients at the midpoint of edge ij would be to use a simple arithmetic average such that

$$\vec{\nabla}\vec{U}_{ij} = \overline{\vec{\nabla}\vec{U}_{ij}} = \frac{1}{2}\left[\vec{\nabla}\vec{U}_i + \vec{\nabla}\vec{U}_j\right] \tag{3.28}$$

However, Mavriplis [146] has shown that this simple averaging can create unbalanced weighting and solution decoupling on quadrilateral elements (in 2D) as well as on prism and

hexahedral elements in 3D cases. To fix these issues, Mavriplis [146] and Haselbacher [93] have proposed the use of a corrected averaging based on the directional derivatives so that

$$\vec{\nabla}\vec{U}_{ij} = \overline{\vec{\nabla}\vec{U}_{ij}} - \left[\overline{\vec{\nabla}\vec{U}_{ij}} \cdot \hat{\vec{r}}_{ij} - \left(\frac{\partial U}{\partial r}\right)_{ij}\right] \hat{\vec{r}}_{ij} \tag{3.29}$$

where $\overline{\vec{\nabla}\vec{U}_{ij}}$ is the arithmetic averaging of Eq. (3.28) and $\hat{\vec{r}}_{ij}$ is the norm of the vector $\vec{r}_{ij}$ connecting the two end nodes of edge ij. Here, the directional derivative is simply defined based on a second-order central difference approximation in the direction of edge ij, i.e.,

$$\left(\frac{\partial U}{\partial r}\right)_{ij} \approx \frac{U_j - U_i}{|\vec{r}_{ij}|} \quad ; \quad \mathcal{O}(|\vec{r}_{ij}|^2) \tag{3.30}$$

where $|\vec{r}_{ij}|$ is the length of the edge ij.

3.6 Spatial Discretization

In the previous section, the evaluation of the convective and viscous fluxes were discussed. Let us go back to the RANS-SA governing equations described in Chapter 2 which in the integral form read

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \vec{U} d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[\vec{F}_c - \vec{F}_v\right] dS = \int_{\mathcal{V}} \vec{Q} d\mathcal{V}$$

As explained earlier, the volume-averaged values at the centroid of the control volume are used here. For the reasons discussed in Section 3.2.1, due to the median-dual CV, the volume averaging leads to the introduction of a mass matrix that couples the solution at node i to its neighboring control volumes. However, for the steady and HB (mathematically-steady) solutions, the volumes are lumped and the mass matrix is replaced by an identity matrix to get the following form of the governing equations for node i

$$\frac{\partial \mathcal{V}_i \vec{U}_i}{\partial t} + \oint_{\partial\mathcal{V}_i} \left[\vec{F}_{c_i} - \vec{F}_{v_i}\right] dS_i = \int_{\mathcal{V}_i} \vec{Q}_i \; d\mathcal{V}_i$$

Using the method of lines, the spatial discretization for the convective and viscous fluxes is performed. Also, the source terms are volume-averaged at the centroid of the control

volume. Thus, the semi-discrete form of the RANS-SA governing equation can be obtained as

$$\frac{\partial \mathcal{V}_i \vec{U}_i}{\partial t} + \sum_{j=1}^{\mathrm{Ngb}_i} \left[\vec{F}_{c_{ij}} - \vec{F}_{v_{ij}} \right] \vec{S}_{ij} = \vec{Q}_i \mathcal{V}_i$$

Here, the discretized fluxes and the source terms are grouped into a vector of residuals $\vec{R}_i$ that reads

$$\vec{R}_i = \sum_{j=1}^{\mathrm{Ngb}_i} [\vec{F}_{c_{ij}} - \vec{F}_{v_{ij}}] \vec{S}_{ij} - \vec{Q}_i \mathcal{V}_i \tag{3.31}$$

In fact, with the decomposition of the convective flux due to flux-difference splitting Roe-scheme (Eq. [3.16]) to fully-convective ($\vec{F}_{\underline{c}}$) and dissipative ($\vec{F}_d$) parts, the total residual can be rewritten as

$$\vec{R}_i = \sum_{j=1}^{\mathrm{Ngb}_i} \vec{F}_{\underline{c}_{ij}} \cdot \vec{S}_{ij} - \sum_{j=1}^{\mathrm{Ngb}_i} \vec{F}_{d_{ij}} \cdot \vec{S}_{ij} - \sum_{j=1}^{\mathrm{Ngb}_i} \vec{F}_{v_{ij}} \cdot \vec{S}_{ij} - \vec{Q}_i \mathcal{V}_i \tag{3.32}$$

where $\vec{F}_{\underline{c}}$ and $\vec{F}_d$ are the first and second terms in the Roe scheme flux-difference Eq. (3.16). This is equivalent to

$$\vec{R}_i = \vec{R}_{\underline{c}} - \vec{R}_d - \vec{R}_v - \vec{R}_s \tag{3.33}$$

where $\vec{R}_{\underline{c}}$, $\vec{R}_d$, $\vec{R}_v$, and $\vec{R}_s$ are the truly-convective, dissipative, viscous, and source term residuals. Finally, by having the total residual, the semi-discrete governing equations can be written as

$$\frac{\partial \mathcal{V}_i \vec{U}_i}{\partial t} + \vec{R}_i = 0 \tag{3.34}$$

For steady cases, the physical time-step will be replaced by a "pseudo-time" step and the temporal discretization is performed which will be discussed next.

3.7 Temporal Discretization

In the previous section, it was shown how, with the help of the "method of lines", the semi-discrete form of the governing equations was obtained. Therefore, we are now left with an ODE in time domain, i.e., Eq. (3.34), which reads

$$\frac{d(\mathcal{V}_i\vec{U}_i)}{dt} + \vec{R}_i = 0 \tag{3.35}$$

at each control volume $\mathcal{V}_i$ defined around node i. It must be noted that for the time-accurate solutions of the unsteady flow problems, the dual-time stepping technique [112] can be used. However, the goal here is to obtain the steady-state solution and therefore, the physical time step can be replaced by a "pseudo time", τ.

For the temporal discretization, one has the choice of the explicit and implicit approaches. In the former approach, the residuals will be evaluated at time $n+1$, so that using a backward finite-difference approximation (backward Euler) of the time derivative term, we will have

$$\frac{(\mathcal{V}_i\vec{U}_i)^{n+1} - (\mathcal{V}_i\vec{U}_i)^{n}}{\Delta\tau} + \vec{R}_i(\vec{U}_i^{n+1}) = 0 \tag{3.36}$$

The implicit time discretization technique is not in the scope of the present study. Therefore, as an alternative, the forward Euler method can be used to rewrite Eq. (3.35) using the explicit approach which reads

$$\frac{\mathcal{V}_i\Delta\vec{U}_i^{n}}{\Delta\tau} + \vec{R}_i(\vec{U}_i^{n}) = 0 \tag{3.37}$$

where

$$\nabla\vec{U}_i^{n} = \vec{U}_i^{n+1} - \vec{U}_i^{n} \tag{3.38}$$

is the change in the flow solution from time-step n to $n+1$ which will be used eventually to update the solution. Also, the control volumes are assumed to remain constant within each time-step, which holds for steady flow problems. The problem with the explicit (forward difference, FD) method is that it is "unconditionally unstable" for the present hyperbolic governing equation. Many temporal discretization techniques for explicit methods are

available in the literature which provide improved stability criteria among which the hybrid multistage Runge-Kutta (RK) scheme of Martinelli [145] and Mavriplis and Jameson [148] is used in this work due to its favorable stability and improved computational efficiency compared to the original multistage RK scheme [215].

3.7.1 Hybrid Multistage Runge-Kutta Scheme

While multistage RK scheme with optimized stage coefficients provide very good stability conditions, their computational cost can be overwhelming due to the recalculation of the entire residual vector at each stage. As discussed in Section 3.6, the upwind convective flux of the Roe scheme can be decomposed to truly-convective and dissipative parts to have the total residual written in the form of Eq. (3.33).

Martinelli [145], and Mavriplis and Jameson [148] suggested combining the four terms in the total residual (Eq. [3.33]) to two un-blended and blended terms such that:

$$\vec{R}_i = \vec{R}_{\mathrm{ub}_i} - \vec{R}_{\mathrm{b}_i} \tag{3.39}$$

where the un-blended part includes the truly convective and the source term portions, i.e.,

$$\vec{R}_{\mathrm{ub}_i} = \vec{R}_{\underline{c}_i} - \vec{R}_{s_i} \tag{3.40}$$

and the blended part includes the residuals due to the viscous and dissipative fluxes, i.e.,

$$\vec{R}_{\mathrm{b}_i} = \vec{R}_{v_i} + \vec{R}_{d_i} \tag{3.41}$$

Therefore, the five-stage hybrid RK scheme applied to Eq. (3.37) can be written as

$$
\begin{aligned}
\vec{U}_i^{(1)} &= \vec{U}_i^{(n)} - \alpha_1 \frac{\Delta\tau}{\mathcal{V}_i}\left[\vec{R}_{\mathrm{ub}_i}^{(n)} - \vec{R}_{\mathrm{b}_i}^{(n)}\right] \\
\vec{U}_i^{(2)} &= \vec{U}_i^{(n)} - \alpha_2 \frac{\Delta\tau}{\mathcal{V}_i}\left[\vec{R}_{\mathrm{ub}_i}^{(1)} - \vec{R}_{\mathrm{b}_i}^{(n)}\right] \\
\vec{U}_i^{(3)} &= \vec{U}_i^{(n)} - \alpha_3 \frac{\Delta\tau}{\mathcal{V}_i}\left[\vec{R}_{\mathrm{ub}_i}^{(2)} - \vec{R}_{\mathrm{b}_i}^{(2,n)}\right] \\
\vec{U}_i^{(4)} &= \vec{U}_i^{(n)} - \alpha_4 \frac{\Delta\tau}{\mathcal{V}_i}\left[\vec{R}_{\mathrm{ub}_i}^{(3)} - \vec{R}_{\mathrm{b}_i}^{(2,n)}\right] \\
\vec{U}_i^{(n+1)} = \vec{U}_i^{(5)} &= \vec{U}_i^{(n)} - \alpha_5 \frac{\Delta\tau}{\mathcal{V}_i}\left[\vec{R}_{\mathrm{ub}_i}^{(4)} - \vec{R}_{\mathrm{b}_i}^{(4,2)}\right]
\end{aligned}
\tag{3.42}
$$

where the stage coefficients, α_k, are carefully optimized and are given in Table 3.3. The last terms in Eq. (3.42) at each stage apply a blending mechanism to reuse $\vec{R}_{\mathrm{b}_i}$ residuals from previous stages such that the viscous and dissipative fluxes are only evaluated at odd stages. The blending process at the third and fifth stages are given by

$$
\begin{aligned}
\vec{R}_{\mathrm{b}_i}^{(2,n)} &= \beta_3 \vec{R}_{\mathrm{b}_i}^{(2)} + (1-\beta_3)\vec{R}_{\mathrm{b}_i}^{(n)} \\
\vec{R}_{\mathrm{b}_i}^{(4,2)} &= \beta_5 \vec{R}_{\mathrm{b}_i}^{(4)} + (1-\beta_5)\vec{R}_{\mathrm{b}_i}^{(2,n)}
\end{aligned}
\tag{3.43}
$$

where the blending coefficients, β_k, are also given in Table 3.3.

Table 3.3: Stage and blending coefficients for (5,3)-scheme (hybrid multistage RK).

Coefficients	stage 1	stage 2	stage 3	stage 4	stage 5
α	0.2742	0.2067	0.5020	0.5142	1.0000
β	1.0000	0.0000	0.5600	0.0000	0.4400

Due to the fact that calculation of the viscous and dissipative fluxes at stages (1, 3, 5) and their blending at stages 3 and 5, this hybrid multistage scheme is also known as the (5, 3)-scheme [21]. The stability conditions for the hybrid RK schemes will be discussed next.

3.7.2 Stability and Time-step Determination

As explained earlier, explicit schemes are conditionally stable, meaning that the time-step Δt used in Eq. (3.42) cannot exceed a certain value. This introduces a stability criteria known as the Courant-Friedrichs-Lewy (CFL) condition which is defined based on the approximate stability analyses for multi-dimensional and non-linear problems. Vijayan and Kallinderis [221] proposed a technique for calculating the maximum time-step at each node

based on the spectral radii of the convective and viscous fluxes. It must be noted that the spectral radius of a flux vector is the largest eigenvalue of its Jacobian. The method of Vijayan and Kallinderis [221] is tailored for the unstructured mixed grid and therefore, it is used in this work. This method gives the maximum allowable time-step at each node i according to

$$\Delta t_i = \mathrm{CFL}\frac{\mathcal{V}_i}{\Lambda_m} \tag{3.44}$$

where Λ_m is the sum of the convective and viscous spectral radii defined as

$$\Lambda_m = \overline{\Lambda}_c + \gamma\overline{\Lambda}_v \tag{3.45}$$

where γ is a scaling factor for the viscous spectral radii which according to Swanson and Turkel is taken to be $\gamma = 1$ for second-order Roe scheme and $\gamma = 2$ for first-order Roe scheme [202].

The convective and viscous spectral radii defined in Eq. (3.45) are given as a sum of radii in all Cartesian coordinates (x and y for 2D and x, y, and z for 3D). As an example, in x-direction, we have

$$\begin{aligned}
\overline{\Lambda}_{c_i}{}^{x} &= (|u| + c)_i \overline{S}_i{}^{x} \\
\overline{\Lambda}_{v_i}{}^{x} &= \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right)_i \left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right)_i \frac{\overline{S^x}_i^2}{\mathcal{V}_i}
\end{aligned}$$

where $\overline{S}_i$ is the projection of the summation of edge vectors connected to node i defined as

$$\overline{\vec{S}}_i = \frac{1}{2}\sum_{j=1}^{\mathrm{Ngb}_i} |\vec{S}_{ij}|$$

Finally, according to Martinelli [145] and Mavriplis and Jameson [148], for the (5, 3)-scheme used in this work, the CFL condition is taken to be CFL = 2.0 and CFL = 1.0 for the first and second order Roe schemes, respectively. The reason for halving the CFL condition for the second-order Roe scheme is that the characteristic waves originated from opposing

sides of the control volume should not be allowed to interact with each other within the cell in one time-step.

Although not pursued here, a global time-step can be defined as the minimum of all nodal time-steps to ensure time accuracy in the case of unsteady flow problems. However, in this work, the maximum permissible time-step at each node (Eq. [3.44]) is used rather than a global time. This method is usually referred to as the "local time-stepping" (LTS) which can result in significant convergence acceleration especially for viscous flows. Since HB method relies on steady-state solutions at coupled sub-time levels, the loss of time accuracy for unsteady flows due to the use of LTS method is not an issue.

3.7.3 Convergence Acceleration using CIRS

In order to accelerate the convergence to steady state, the local time-stepping (Section 3.7.2) and the residual smoothing methods are used in this work and details of the latter approach are discussed here. Additionally, a novel convergence acceleration technique based on the reduced-order-modeling (ROM) is developed herein which will be presented independently in Section 3.10.

The concept of residual smoothing was proposed by Jameson and Baker [115] to mimic the characteristics of an implicit scheme in the framework of an explicit scheme. That is why this method is also known as the implicit residual smoothing or IRS. Jameson and Baker [115] showed that by replacing the total residual at each node by a weighted average of the residuals in the adjacent nodes (direct neighbors), the stability condition (CFL criterion) of the explicit scheme can be substantially improved. They also showed that such averaging process leads to the attenuation of high frequency errors in the smoothed residual.

While different variations of the implicit residual smoothing are available in the literature [21], in this work, the central IRS (CIRS) method of Jameson et al. [116] is used which is tailored for unstructured grids. The CIRS method, applies a Laplacian operator to the residual vector such that

$$\mathcal{L}(\vec{R}(\vec{U}_i)) = \sum_{j=1}^{\mathrm{Ngb}_i} \theta_{ij} \left[\vec{R}(\vec{U}_j) - \vec{R}(\vec{U}_i) \right] \tag{3.46}$$

where θ_{ij} are the weights for each node neighbor j connected to node i. Thus, the smoothed residual at node i can be evaluated by solving the following system of linear equations

$$\vec{R}^*(\vec{U}_i) + \sum_{j=1}^{\mathrm{Ngb}_i} \epsilon \left[\vec{R}^*(\vec{U}_i) - \vec{R}(\vec{U}_j)\right] = \vec{R}(\vec{U}_i) \tag{3.47}$$

where $\vec{R}^*$ and the RHS term are the smoothed and unsmoothed (original) residuals, respectively. The above system can be easily solved using two to three steps of Jacobi iteration since the matrix is diagonally dominant. Moreover, the smoothing coefficient ϵ can take any value between zero and one. While $\epsilon = 0$ means no smoothing, $\epsilon \to 1$, can lead too excessive smoothing that can result in the loss of essential information. Therefore, Jameson et al. [116] suggest a smoothing coefficient close to 1/2 (or $0.3 \leq \epsilon \leq 0.8$ according to [21]).

It must be noted that the CIRS technique can be performed at each stage of the explicit RK scheme. However, a more relaxed approach can involve the application of CIRS at odd stages of the (5,3)-scheme to improve the efficiency of the numerical solver.

3.8 HB Method for Periodic Flows

For fluid flows that are temporally periodic, the harmonic balance (HB) method is used to write the conservation flow variables at each node i in terms of a truncated Fourier series including a predefined number of harmonics, N, such that

$$\vec{U}_i(t) = \mathbf{A}_0 + \sum_{n=1}^{N} \left[\mathbf{A}_n cos(\omega n t) + \mathbf{B}_n sin(\omega n t)\right] \tag{3.48}$$

where $i = 1 : 2N + 1$, ω is the fundamental frequency of excitation, and $\mathbf{A}_0, \mathbf{A}_n$, and $\mathbf{B}_n$ are the Fourier series coefficients. In the framework of the HB method (also referred to as the high-dimensional harmonic balance or HDHB), the flow variables are computed and stored at $2N+1$ equally-spaced sub-time levels over a single period. Thus, the time-derivative term in Eq. (3.34) can be replaced by a pseudo-spectral operator, $\mathbf{D}$, that gives

$$\mathbf{D}(\mathcal{V}_i \vec{U}_i^*) + \vec{R}(\vec{U}_i^*) = 0 \tag{3.49}$$

where both the residual and the time-derivative terms are defined in terms of the sub-time level flow solutions at node i, i.e., $\vec{U}_i^*$. As explained in Chapter 2, the pseudo-spectral operator is defined as

$$\frac{d}{dt} \approx \mathbf{D} = \frac{d\mathbf{E}^{-1}}{dt}\mathbf{E}$$

where $\mathbf{E}$ and $\mathbf{E}^{-1}$ are the discrete Fourier transform and its inverse. In order to obtain the steady-state solution at each sub-time level, a "pseudo-time" derivative term is added to the HB Equation of 3.49. This equation now reads

$$\frac{\partial}{\partial \tau}(\mathcal{V}_i \vec{U}_i^*) + \mathbf{D}(\mathcal{V}_i \vec{U}_i^*) + \vec{R}(\vec{U}_i^*) = 0$$

The HB term, $\mathbf{D}(\mathcal{V}_i \vec{U}_i^*)$ can be treated as a volumetric source term and combining it with the residual term $\vec{R}(\vec{U}_i^*)$ leads to

$$\frac{\partial(\mathcal{V}_i \vec{U}_i^*)}{\partial \tau} + \vec{R}'(\vec{U}_i^*) = 0 \tag{3.50}$$

which is exactly similar to Eq. (3.35) with physical time, t, being replaced by a pseudo-time, τ. The total residual term $\vec{R}'(\vec{U}_i^*)$ now includes the discretized convective flux, viscous flux, flow source terms, and the HB source term. Finally, the system of semi-discretized initial-value ODEs displayed in Eq. (3.50) is marched temporally toward a steady-state solution using an explicit five-stage hybrid Runge-Kutta scheme described in Section 3.7.1.

3.8.1 Mesh Motion

In the nominal HB solver without the r-refinement AMR (as will be discussed later in this chapter), a mesh deformation technique is required to obtain the sub-time level grids. Also, during the design optimization process, the geometry deformation requires the computational mesh to be moved in order to conform to the surface displacements. The mesh deformation in UNPAC solver is performed using radial basis function (RBF) approach that was initially introduced by de Boer et al. [48]. In this approach, the nodal displacements at the interior nodes are calculated based on radial basis functions of their distance to the control points

(boundary nodes). The RBF approach used in this work follows the method of Rendall and Allen [173] where the location of the grid nodes in the interior domain is defined by

$$\vec{x}' = \sum_{i=1}^{N_s} \alpha_i \phi(||\vec{x} - \vec{x}_i||) \tag{3.51}$$

Here, $\vec{x}_i$ is the location of the discrete base points on the moving/deforming boundary at which the displacements are known *a priori* according to the excitation motion (in the HB method) or the deformation due to the shape optimization process. In the RBF equation (3.51) described above, ϕ is the basis function based on the normalized distances of the volume nodes (nodes in the interior domain whose movement is being evaluated) to the surface nodes (based points). There are many different choices for the basis function and the interested readers are referred to [173] for a review of various functions. However, in this work, Wendland's C2 basis function [224] is adopted which is defined as

$$\phi(\xi) = (1 - \xi)^4 (4\xi + 1) \tag{3.52}$$

where ξ is the distance between the two points, $\vec{x}$ and $\vec{x}_i$, scaled by a support radius, R, i.e., $\xi = ||\vec{x} - \vec{x}_i||/R$. The Wendland's C2 basis function not only provides high quality transformed meshes but also leads to a well-conditioned linear system due to its compact support based on the pre-tuned support radius [173]. In practice, the support radius, R, is taken to be around 5-10 times the chord length and must be less than the far-field boundary radius so that the node movements are damped close to the far-field boundary. The RBF mesh deformation method described herein, determines the updated locations at the interior nodes using a weighted sum approach with coefficients α_i. These coefficients are determined according to the relative motion of the base points and their evaluation requires solving a system of linear equations in each Cartesian coordinate [48, 173].

Obviously, in the HB method, sub-time level grids need to be updated individually according to the corresponding motion of the body in that sub-time level. With the sub-time level grids obtained using the RBF mesh deformation technique, the grid velocity components can be approximated at grid node i using the pseudo-spectral operator (2.84) such that

$$\dot{\vec{x}}_i \approx \mathbf{D}\vec{x}_i^* \tag{3.53}$$

where $\vec{x}_i^*$ is the position vector at node i for $2N+1$ sub-time levels. It must be noted that for highly non-linear body motions, retaining a relatively small number of harmonics in the HB solver can lead to large approximation errors in grid velocities that can negatively affect the accuracy of the ALE calculations. Tardif and Nadarajah [205] has introduced a radial basis function method for velocities (RBFV) that follows a similar approach to the original RBF technique for mesh deformation considering the fact that the velocities at the control points (boundary nodes) can be determined analytically based on the prescribed body motion. These velocities can be then interpolated at the interior nodes using the RBFV approach [205] to obtain the accurate grid velocities to be used in the ALE formulation.

3.8.2 GCL Error Source Term

As discussed in Chapter 2, in order to avoid numerical errors produced by the deformation of the median-dual control volumes, an additional conservation law must be solved simultaneously with the rest of the governing equations. This additional equation that was first demonstrated by Thomas and Lombard [209] is referred to as the Geometric Conservation Law (GCL) and its violation can lead to degradation of accuracy especially in large amplitude aeroelastic applications [135]. In general, GCL states that the rate of the total change of the control volume must be equal to the rate of incremental volume change due to the movement of the boundaries of the CV. It was shown that this can be represented by a semi-discrete differential equation given as

$$\frac{\partial \mathcal{V}_i}{\partial t} - \sum_{n=1}^{N_{f_i}} \left(\vec{v}_{\text{grid}} \cdot \vec{S} \right)_n = 0 \tag{3.54}$$

where N_{f_i} is the number of faces of the control volume i and $\vec{v}_{\text{grid}}$ is the velocity averaged at the midpoint of each face. The geometric conservation equation given in Eq. (3.54) is usually solved in the framework of unsteady time-accurate solvers using the same numerical scheme that was used to discretize and solve the rest of the governing equations. This is necessary

for obtaining a consistent solution method [21] and was also applied to non-linear frequency domain (NLFD) techniques [205]. Alternatively, Ma et al. [142] have shown that a source term approach can also be used to preserve the GCL condition eliminating the need to solve an additional conservation equation. In their approach, a volumetric error (or imbalance) is added as a GCL source term to the governing equations. This error at node i can be defined as

$$\left(\vec{E}_{\mathrm{GCL}}\right)_i = \frac{\partial \mathcal{V}_i}{\partial t} - \sum_{n=1}^{N_{f_i}} \left(\vec{v}_{\mathrm{grid}} \cdot \vec{S}\right)_n \tag{3.55}$$

In this work, the source term approach of Ma et al. [142] is adopted due to its simplicity. To decrease the error induced by the deformation of the control volumes, the semi-discretized governing equations (Eq. [3.50]) are modified to include the GCL volumetric source term so that [142]

$$\frac{\partial}{\partial \tau}\left(\mathcal{V}_i \vec{U}_i^*\right) + \vec{R}'(\vec{U}_i^*) = \left(\vec{E}_{\mathrm{GCL}}\right)_i \cdot \vec{U}_i^* \tag{3.56}$$

It must be noted that HB results have shown that the GCL errors tend to be very small in magnitude and have minimal effects on the convergence rate and accuracy of the numerical solver. Nevertheless, the GCL error source terms are included in the UNPAC solver to preserve the geometric conservation law.

3.9 Boundary Condition Treatment

With the spatial and temporal discretizations taken care of, the flow conditions at the boundaries will be discussed next. These boundaries can be classified as (1) physical (due to the physical boundaries such as walls) and (2) numerical (due to numerical boundaries of the computational domain such as farfield, symmetry, etc.). For the median-dual control volumes neighboring the boundaries (as can be seen in Figure 3.8), boundary fluxes must be evaluated. Inaccurate or faulty calculation of these fluxes can spoil the accuracy of the simulations as well as the convergence rate of the numerical solver. In this section, boundary condition (BC) treatments at different boundary types are presented.

3.9.1 Solid Wall

An important part of the computational domain is the natural boundaries occurring at the surface of the physical obstacles. These solid walls can be stationary or moving depending on the definitions of the flow problem being studied. Here, the details of the boundary condition treatment for different types of wall boundaries are discussed.

Free-Slip BC

For inviscid flows, due to the absence of friction forces at the wall, the fluid is allowed to slip over the surface. Therefore, normal components of the fluid velocity would be zero such that

$$\vec{v} \cdot \vec{n} = V = 0 \tag{3.57}$$

holds on the wall surface. Since there are no viscous effects involved, only the convective flux must be considered which according to Eq. (2.12) reads

$$\vec{F}_{c_{\text{wall}}} = \begin{bmatrix} \rho V \\ \rho u V + p\ n_x \\ \rho v V + p\ n_y \\ \rho w V + p\ n_z \\ \rho H V \end{bmatrix}_{\text{wall}} \tag{3.58}$$

at the wall boundary. In the case of a stationary wall, due to the fully tangential velocity, the contravariant velocity, V, must be zero (Eq. [3.57]) which leads to

$$\vec{F}_{c_{\text{wall}}} = \begin{bmatrix} 0 \\ p\ n_x \\ p\ n_y \\ p\ n_z \\ 0 \end{bmatrix}_{\text{wall}} \tag{3.59}$$

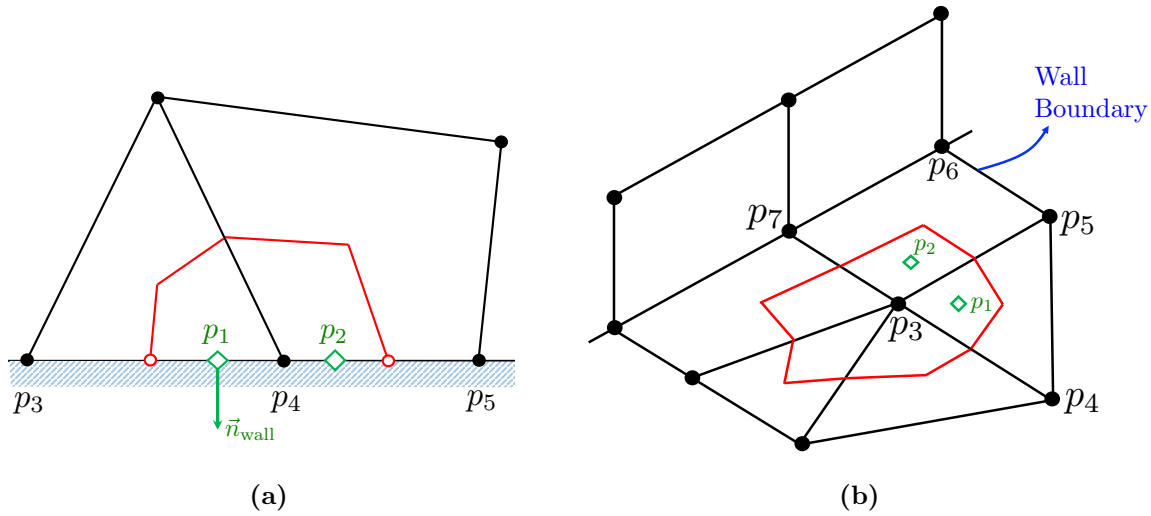


Figure 3.8: Locations at which the boundary flux is evaluated for (a) 2D and (b) 3D cases (marked by diamonds).

Therefore, only the static pressure at the wall needs to be determined to evaluate the convective flux. The location at which this flux is evaluated falls at the centroid of the boundary face as shown in Figure 3.8 for both 2D and 3D cases. Luo et al. [141] have shown that in the framework of a median-dual finite-volume method, FEM-based averaging must be used to approximate pressure at the wall boundary.

In the 2D cases, the boundary face is always an edge that belongs to either a triangular or a quadrilateral element. As shown in [141, 21], the type of the boundary element should be considered when the wall pressure is approximated. In this work, the wall pressure at nodes p_1 and p_2 shown in Figure 3.8a are evaluated by

$$p_1 = \frac{1}{6}(5p_4 + p_3) \tag{3.60}$$

$$p_2 = \frac{1}{4}(3p_4 + p_5) \tag{3.61}$$

For 3D cases, however, the boundary face can be either a triangular or a quadrilateral face of the boundary element and the convective flux requires the information regarding the static pressure at the centroid of that face. Once again, using a finite-element approximation [141],

the wall pressure at node p_1 of a triangular boundary face and at node p_2 of a quadrilateral boundary face (as shown in Figure 3.8b) are determined according to

$$p_1 = \frac{1}{8}(6p_3 + p_4 + p_5) \tag{3.62}$$

$$p_2 = \frac{1}{16}(9p_3 + 3p_5 + 3p_7 + p_6) \tag{3.63}$$

No-Slip BC

For viscous flows, a no-slip boundary condition in enforced at the wall which requires the relative flow velocities to vanish. This means that for a stationary wall, the fluid velocity vector must be zero, i.e., $\vec{v} = 0$. Therefore, the contravariant flow velocity at the wall would also be zero and the convective flux of Eq. (3.58) must be considered again. Once again, the same process will be used to determine the static pressure at the wall as described for the case of the free-slip wall.

Now, let us consider the viscous flux which is defined according to Eq. (2.13) on the wall boundary as

$$\vec{F}_{v_{\text{wall}}} = \begin{bmatrix} 0 \\ \vec{\tau}_{x_1} \cdot \vec{n} \\ \vec{\tau}_{x_2} \cdot \vec{n} \\ \vec{\tau}_{x_3} \cdot \vec{n} \\ \vec{\Theta} \cdot \vec{n} \end{bmatrix}_{\text{wall}} \tag{3.64}$$

In the case of the stationary wall, the viscous flux at the boundary only includes the heat flux such that

$$\vec{F}_{v_{\text{wall}}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vec{\Theta} \cdot \vec{n} \end{bmatrix}_{\text{wall}} \tag{3.65}$$

where the elements of the $\vec{\Theta}$ vector are now defined as

$$\Theta_{\text{wall}_i} = k \left(\frac{\partial T}{\partial x_i} \right)_{\text{wall}} ; \quad i = 1, 2, 3 \tag{3.66}$$

Therefore, according to the heat transfer boundary condition, the wall boundary can be classified as (1) adiabatic or (2) isothermal (with prescribed temperature). In the case of an adiabatic wall (idealized adiabatic condition), since

$$\vec{\nabla} T_{\text{wall}} \cdot \vec{n} = 0 \tag{3.67}$$

the viscous stress work, $\vec{\Theta}$, will vanish and the viscous flux wil be entirely zero. However, in the case of an isothermal wall, the heat flux should be approximated numerically based on the prescribed temperature at the wall and the interior solution. For this reason, a search algorithm is performed on all direct neighbors of the boundary node to find the neighbor that is located at the most perpendicular location to the boundary. Here, the angle between the boundary node normal vector and the vector connecting the boundary node to its neighbors are considered. Therefore, the neighboring node corresponding to the minimum angle (most perpendicular) is selected. Assuming that this neighbor is at node j, the temperature gradient at the wall boundary is approximated using a one-sided finite-difference formula via

$$\left(\frac{\partial T}{\partial x_i} \right)_{\text{wall}} \approx \frac{T_{\text{wall}} - T_j}{||\vec{r}_{\text{wall},j}||} \tag{3.68}$$

where $\vec{r}_{\text{wall},j}$ is the vector connecting the wall boundary node to its neighbor.

Moving Wall BC

For flows involving a moving boundary, special care must be given to the evaluation of the convective and viscous fluxes at the moving walls. As discussed previously in Chapter 2, the Arbitrary Lagrangian-Eulerian (ALE) form of the governing equations results in a modified form of the convective flux at the wall boundary which now reads

$$\vec{F}^{\text{ALE}}_{c_{\text{wall}}} = \begin{bmatrix} \rho V_r \\ \rho u V_r + p\ n_x \\ \rho v V_r + p\ n_y \\ \rho w V_r + p\ n_z \\ \rho H V_r + p\ V_{\text{grid}} \end{bmatrix}_{\text{wall}} \tag{3.69}$$

where V_r and V_{grid} are the contravariant velocity relative to the grid motion and the contravariant grid velocity, respectively. As explained earlier, the contravariant flow velocity is zero at the boundary for both the free-slip and the no-slip walls. Since the contravariant relative velocity at the wall zero, the ALE form of the convective flux can be rewritten as

$$\vec{F}^{\text{ALE}}_{c_{\text{wall}}} = \begin{bmatrix} 0 \\ p\ n_x \\ p\ n_y \\ p\ n_z \\ p\ V_{\text{grid}} \end{bmatrix}_{\text{wall}} \tag{3.70}$$

This means that in addition to the pressure contribution to the momentum equations, a convective flux for the energy equation can now be written which is based on the grid velocity at the wall boundary (wall velocity). Viscous fluxes are similar to those shown previously in Eq. (3.64). Assuming that the gradients of the grid velocity are negligible at the wall boundary, the viscous stresses for the momentum equations can be ignored. However, the elements of the $\vec{\Theta}$ vector are now defined as

$$\Theta_{\text{wall}_i} = \vec{v}_{\text{wall}} \cdot \vec{\tau}_{x_i} + k\frac{\partial T}{\partial x_i}; \quad i = 1, 2, 3 \tag{3.71}$$

which means that the discretized form of the viscous flux for the energy equation is now augmented by a $(V_{\text{wall}} \times \overline{\overline{\tau}})$ term, where V_{wall} is the contravariant grid velocity at the wall boundary and $\overline{\overline{\tau}}$ is the viscous stress tensor. The second term in Eq. (3.71) is evaluated using the same procedure that was described for the stationary no-slip walls depending on the choice of isothermal or adiabatic boundary condition.

Before moving on to the treatment of the far-field boundary conditions, it must be noted that for the Spalart-Allmaras turbulence model used in this work, the eddy viscosity at the solid wall is set to zero, i.e., $\tilde{\nu}_{\text{wall}} = 0$. Thus, there are no contributions to the convective or viscous fluxes of the turbulence model at the wall boundary.

3.9.2 Far-Field

While physical domains are unbounded in external flows around wings or airfoils that are sufficiently far from the ground, the computational domain imposes an artificial boundary that limits the size of the grid. Ideally, two conditions must be satisfied at the far-field boundary:

1. Outgoing waves should not be reflected back into the computational domain,
2. Extent of the computational domain must not affect the near-field solution (boundary independent).

To achieve these goals, different methods have been introduced in the framework of compressible RANS solvers. In this work, the method of characteristics [225] is used which offers non-reflecting boundary conditions. It must be noted that the characteristic-based boundary condition treatment is based upon the assumption of zero circulation. Unless the far-field boundary is placed far enough from the lifting body, the no-circulation assumption does not hold. Therefore, in the UNPAC solver, the vortex correction method of Usab and Murman [213] is used to account for the circulation that is proportional to the lift force generated by the lifting body.

The concept of characteristic-based boundary treatment is based on the solution of the uni-directional (or 1D) Euler equations in the direction normal to the far-field boundary. In this approach the sign of the eigenvalues for the convective flux Jacobian is considered to determine which characteristic waves are leaving or entering the computational domain. It must be noted that the characteristic variables consist of pressure, density, and the velocity vector components (a total of 4 variables for 2D and 5 variables for 3D cases).

Details of the characteristic-based non-reflecting boundary condition method can be found in [21, 225] and here only the final relations are provided for subsonic/supersonic

inflow and outflow conditions. In order to determine the direction of the flow (inflow or outflow) for all the above cases, the sign of the contravariant velocity at the boundary node is used. Therefore, based on the default orientation of the boundary normal (always pointing outward), positive and negative contravariant velocities would determine an outflow and an inflow condition, respectively.

Subsonic Inflow

For the total of five characteristic variables in 3D subsonic inflow case, four characteristic variables are determined based on the free-stream conditions while the last one is extrapolated from within the computational domain. Therefore, boundary values of the characteristic variables are defined as

$$p_b = \frac{1}{2}\left[p_\infty + p_b' - \rho_b' c_b' \left(\vec{v}_d \cdot \vec{n}\right)\right] \tag{3.72}$$

$$\rho_b = \rho_\infty + \frac{p_b' - p_\infty}{c_b'^2} \tag{3.73}$$

$$\vec{v}_b = \vec{v}_\infty - \left[\frac{p_\infty - p_b'}{\rho_b' c_b'}\right]\vec{n} \tag{3.74}$$

where the primed variables are based on the old solutions at the boundary node (before getting corrected) and $\vec{v}_d$ is the relative velocity at the boundary based on the old solution given by $\vec{v}_d = \vec{v}_\infty - \vec{v}_b'$. It must be noted that the boundary normal vector, $\vec{n}$, is assumed to be pointing outward.

Supersonic Inflow

In the case of the supersonic inflow, things are much simpler since all characteristic waves are incoming and the boundary variables can be simply set equal to the free-stream conditions, i.e.,

$$p_b = p_\infty \tag{3.75}$$

$$\rho_b = \rho_\infty \tag{3.76}$$

$$\vec{v}_b = \vec{v}_\infty \tag{3.77}$$

Subsonic Outflow

For the case of the subsonic outflow, situation is reversed compared to the subsonic inflow conditions. Therefore, in 3D case, four characteristic variables are extrapolated and only the last one is determined based on the free-stream conditions such that

$$p_b = p_\infty \tag{3.78}$$

$$\rho_b = \rho'_b + \frac{p_b - p'_b}{c'^2} \tag{3.79}$$

$$\vec{v}_b = \vec{v}_\infty - \left[\frac{p'_b - p_b}{\rho'_b c'_b}\right]\vec{n} \tag{3.80}$$

where, once again, the primed variables are the old solutions at the boundary (before they are corrected).

Supersonic Outflow

Finally, for the supersonic outflow, all characteristic variables are extrapolated from the interior domain which means that no correction is necessary. Therefore, the following boundary values can be defined

$$p_b = p'_b \tag{3.81}$$

$$\rho_b = \rho'_b \tag{3.82}$$

$$\vec{v}_b = \vec{v}'_b \tag{3.83}$$

As can be seen, the treatment of the inflow and outflow conditions at supersonic regimes is much simpler due to the fact that all characteristic waves are propagating in the same direction. Additionally, the turbulent eddy viscosity at the far-field boundary is usually taken to be $\tilde{\nu} = 3\nu_\infty$ for the Spalart-Allmaras turbulence model.

3.9.3 Symmetry

Implementation of a symmetry boundary condition is very similar to that of the free-slip wall described earlier for the inviscid flows. Due to the fact that there is no flux across the boundary face, convective and viscous fluxes can be ignored all together. However, best practice would be to correct velocity vectors as well as the gradients at the symmetry boundary such that only tangent components are preserved.

3.9.4 Periodic

In fluid flow problems involving spatial periodicity, shrinking the computational domain around the repeating region can be highly useful for computational efficiency. As an example, when simulating flow around wind turbine blades or helicopter rotors, one can simply limit the computational domain to a single blade and incorporate periodic boundaries. This problem is often referred to as "rotational periodicity" and involves the rotation of the coordinate system to correct the vector information.

Let us assume a case where there is rotation about the x-axis. As shown in Figure 3.9, only a ψ-degree cut of the computational domain is considered and periodic BC is applied to boundaries A and B. The periodic boundary condition treatment implemented in the UNPAC solver is then described as follows:

1. First, the median-dual control volumes from the two periodic boundaries should be matched with each other. This is done using a search algorithm performed during the pre-processing stage. In this approach, master nodes from boundary A (e.g. node a in Figure 3.9) are rotated by an angle ψ and then matched to slave nodes from boundary B that are at a certain user-defined threshold distance which is taken to be 10^{-12} in this work (e.g. node b in Figure 3.9).

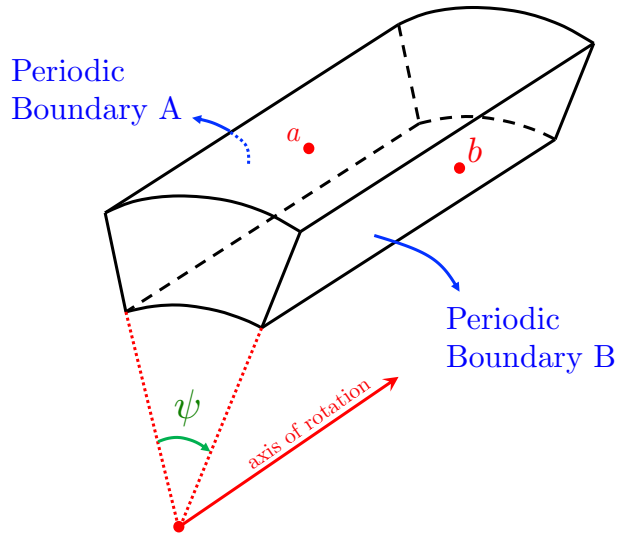


Figure 3.9: Definition of the periodic boundary nodes a and b for a case of rotational periodicity with angle ψ applied to boundaries A and B.

2. During the solution process, the total residuals at all nodes (including the periodic boundary nodes) are evaluated.

3. Next, the residuals at the matched nodes on periodic boundaries are summed up according to

$$\vec{R}_{a_{\text{total}}} = \vec{R}_a + \vec{R}'_b \tag{3.84}$$

$$\vec{R}_{b_{\text{total}}} = \vec{R}_b + \vec{R}'_a \tag{3.85}$$

   where $\vec{R}'_a$ and $\vec{R}'_b$ are the corrected residuals due to coordinate rotation which will be described in the next step.

4. Since the scalar variables (pressure, density, and eddy viscosity in our case) are invariant with respect to coordinate rotation [21], only the velocity vectors and gradients must be corrected. This correction process involves rotating the vector quantities from one periodic boundary to the other using a rotation matrix such that

$$\vec{R}'_a = \mathcal{P}\vec{R}_a \tag{3.86}$$

$$\vec{R}'_b = \mathcal{P}'\vec{R}_b \tag{3.87}$$

where the matrices $\mathcal{P}$ and $\mathcal{P}'$ are defined for the ψ-degree rotation about x-axis according to

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_r) & -\sin(\psi_r) \\ 0 & \sin(\psi_r) & \cos(\psi_r) \end{bmatrix} \tag{3.88}$$

$$\mathcal{P}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_r) & \sin(\psi_r) \\ 0 & -\sin(\psi_r) & \cos(\psi_r) \end{bmatrix} \tag{3.89}$$

where ψ_r is the cut (rotation) angle in radian, i.e., $\psi_r = \frac{\pi}{180}\psi$.

Note 1: Similar rotation matrices can be defined for special cases with rotation about y-axis and z-axis.

Note 2: Rotation about an arbitrary axis involves a more elaborate process and details can be found in [161].

3.10 ROM-Based Convergence Acceleration[1]

One of the main contributions of this work to the state-of-the-art is the development of a novel convergence acceleration technique based on the reduced-order-modeling (ROM) technique. The idea of model reduction of dynamical systems is to transform the original

[1]This section, in part, is a reprint of the material as it appears in AIAA Journal 55 (9), 3059-3071 titled "*Convergence Acceleration of Fluid Dynamics Solvers Using a Reduced–Order Model*" (2017). Authors: **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici.

high-order system of governing equations to a system of much lower order, whereby only the most important or dominant parts of the system dynamics are preserved. In the present approach proposed in this work, the well-chosen snapshots of a slowly converging solution are used to obtain the reduced order model. The simplicity and ease of implementation of the method makes it an elegant approach in accelerating the convergence of iterative explicit CFD solvers, which have been pursued by the author and his colleagues in the framework of a 2D structured compressible RANS solver [54]. Kaminsky et al. [121] have also applied this technique to approximate the converged sensitivities for a discrete adjoint solver using unconverged (early) sensitivity values. It was shown that the computational cost of the adjoint solver (and the accompanying optimization algorithm) can be greatly reduced using this technique. Also, more recently, the same approach has been used in the framework of a continuous adjoint solver for accelerating the convergence of the iterative solver for obtaining the adjoint solutions [120].

The entire computational cost of the proposed technique at each cycle (including the I/O and projection computation) is in general equivalent to having up to 10 additional iterations of the nominal solver (usually about 3-4 iterations) which makes the technique a very robust acceleration technique that can efficiently reduce the required number of iterations of the explicit solver for reaching the steady state solution. It must be noted that, the proposed technique can be particularly useful for unstructured solvers, as pursued herein, for which the agglomeration of nodes (or cells) in a multigrid scheme is not straightforward [147]. In this section, details of this method and its implementation in the UNPAC solver is discussed.

3.10.1 Model Reduction

As explained previously, the model order reduction is utilized to transform the original high-order system of governing equations to a system of much lower order, whereby only the most important or dominant parts of the system dynamics are preserved. In general, a projection-based model reduction method "compresses" the system's state information by projecting the state behavior onto a lower dimensional subspace and rewrites the governing equations in a compressed representation. In mathematical terms, the projection-based class

of reduced-order model (ROM) methods can be described by assuming the following set of governing equations defined in the semi-discrete form as

$$\frac{d\vec{U}}{dt} + \mathcal{N}(\vec{U}) = 0 \tag{3.90}$$

where $\vec{U}$ is the N-dimensional vector of state and $\mathcal{N}$ is the non-linear discretization operator. All projection-based model reduction methods share the same feature which is defined as follows. The projection-based ROM tries to find M-dimensional ($M \ll N$) subspaces S_1 and S_2 of the state space that will yield a reduced system as a result of projection of the state onto S_1 and the residual onto S_2. If subspaces S_1 and S_2 are equal, the projection is *orthogonal*, otherwise it is *oblique*. For the case of an orthogonal projection, a matrix of basis vectors $\mathbf{\Phi}$ (which will be defined in the following subsection) and its Hermitian transpose can be used, resulting in the reduced-order model given by

$$\Phi^T \frac{d}{dt}\left(\Phi\vec{\xi}\right) + \Phi^T \mathcal{N}\left(\Phi\vec{\xi}\right) = 0 \tag{3.91}$$

where the values of the vector $\vec{\xi}$ are picked so as to minimize the error in the state-solution approximation via

$$\vec{U} \cong \mathbf{\Phi}\vec{\xi} \tag{3.92}$$

and $\mathbf{\Phi}$ is the matrix of basis vectors.

3.10.2 Solution Projection

To motivate the proposed convergence acceleration technique, Eq. (3.37) can be first rewritten in the form of an update formula for the iterative process as below

$$\vec{U}^{n+1} = \vec{U}^{n} + \vec{R}(\vec{U}) \tag{3.93}$$

Note that the residual vector, $\vec{R}$, used here must include a negative sign when compared to the one shown in Eq. 3.37. Therefore, the fully converged solution, $\vec{\tilde{U}}$, is reached when

$$\vec{\tilde{U}} = \vec{\tilde{U}} + \vec{R}(\vec{\tilde{U}}) \tag{3.94}$$

which means that the residual of the fully converged solution would be very small (machine accuracy zero). The origin of the idea for the proposed convergence acceleration technique comes from the assumption that in later stages of the iterative procedure, the iteration becomes linearly convergent. Given the fact that this assumption holds, the residual vector can be written as a linear function of the flow variables, i.e.,

$$\vec{R}(\vec{U}) \approx \mathbf{A}\vec{U} - \vec{b} \tag{3.95}$$

with the assumption that both $\mathbf{A}$ and $\mathbf{b}$ become independent of the iteration number, n, as the solver is converging. The goal here is to find a projected solution as a linear combination of the dominant states that would drive the residual of the flow solver to machine accuracy. For this reason, M state-vector solutions $\{\vec{U}_i\}_{i=1}^{M}$, called "snapshots", are first stored over a portion of the iterative process. It is assumed that there are N state variables in the flow solver where N is the total number of computational nodes times the number of state variables for each node (5 for three-dimensional Euler and laminar Navier-Stokes equations and 6 for three-dimensional Reynolds-Averaged Navier-Stokes (RANS) equations with one-equation Spalart-Allmaras turbulence model [194]) as discussed previously in Chapter 2.

Correlation-Based Acceleration

In the first version of the present ROM-based convergence acceleration, a set of solution snapshots is taken which forms an $N \times M$ system of the following form

$$\mathbf{\Phi} = \begin{bmatrix} | & | & & | \\ \vec{U}_1 & \vec{U}_2 & \dots & \vec{U}_M \\ | & | & & | \end{bmatrix}_{N \times M} \tag{3.96}$$

While the basic idea behind the proper-orthogonal-decomposition is to use orthogonal basis vectors as a mean for the development of the reduced-order-model, here, the snapshots are taken to be the basis vectors, thus eliminating the need for the calculation of an

orthogonal subspace. Ekici and Hall [59] state that this is a valid alternative to the POD-based orthogonal basis vector calculation although it requires the snapshots of the solution to be taken such that they would capture the dominant behaviors of the flow field. This means that one may have to use more basis vectors compared to a POD approach. With the well-chosen set of snapshots, it can be assumed that the span of our snapshots

$$\text{span}(\mathbf{\Phi}) = \left\{ \sum_{i=1}^{M} \vec{U}_i \xi_i \quad | \quad \xi_i \in R \text{ and } \vec{U}_i \in \mathbf{\Phi} \right\} \tag{3.97}$$

covers the fully converged (or a better projected) solution. In other words, the projected solution, which hypothetically approximates the fully converged solution, belongs to the span of the solution snapshots, i.e., span($\mathbf{\Phi}$). Thus, the projected solution can be written as

$$\vec{U}_{\text{projected}} = \sum_{i=1}^{M} \vec{U}_i \xi_i = \mathbf{\Phi}\, \boldsymbol{\xi} \tag{3.98}$$

The goal is to find a projected solution that can approximate the fully converged solution as closely as possible. Therefore, the expansion coefficients, ξ_i, need to be found such that

$$\vec{R}(\vec{U}_{\text{projected}}) \approx 0 \tag{3.99}$$

According to Eq. (3.95), the residual can be written as

$$\vec{R}(\mathbf{\Phi}\, \vec{\xi}) = \mathbf{A}\mathbf{\Phi}\, \vec{\xi} - \vec{b} = 0 \tag{3.100}$$

which means the following system of linear equations must be solved to obtain the expansion coefficients:

$$\mathbf{A}\mathbf{\Phi}\, \vec{\xi} = \vec{b} \tag{3.101}$$

Once again, the coefficients, ξ, are chosen so that the residual of the flow solver that is projected onto the space spanned by the snapshots is zero. The size of the above system is $N \times N$, which makes it computationally demanding to solve. As a remedy, the system can

be pre-multiplied by the transpose of the matrix $\mathbf{\Phi}$, thus, reducing the order of the equation down to $M \times M$ ($M \ll N$) which is computationally cheap to solve. Therefore, we have

$$\mathbf{\Phi}^T \mathbf{A} \mathbf{\Phi}\, \vec{\xi} = \mathbf{\Phi}^T \vec{b} \tag{3.102}$$

In the above equation, it is first required to calculate the right hand side vector, $\vec{b}$. In the framework of the compressible flow solver utilized in this work, the solution cannot admit values of zero since the density and pressure terms must have strictly positive values. If the solver can be initiated from an all zero initial condition (e.g. for the incompressible streamfunction-vorticity form of Navier-Stokes), then we have

$$\vec{R}(\vec{U} = \vec{0}) = -\vec{b} \quad \rightarrow \quad \vec{b} = -\vec{R}(\vec{U} = \vec{0}) \tag{3.103}$$

In density-based flow cases where it is not possible to initiate the CFD solver from an all zero initial solution vector, a matrix-free Jacobian-vector product approximation method [125] is used. Thus, the following second-order centered-difference formula can be used to approximate the Jacobian-vector-product such that

$$\frac{\partial \vec{R}}{\partial \vec{U}} \vec{U} = \mathbf{A}\vec{U} \approx \frac{\vec{R}(\vec{U} + \epsilon \vec{U}) - \vec{R}(\vec{U} - \epsilon \vec{U})}{2\epsilon} \quad ; \quad \mathcal{O}(\epsilon^2) \tag{3.104}$$

Here, ϵ is the perturbation parameter, which is generally taken to be the square-root of machine zero ($2^{-53} \approx 10^{-16}$ in double precision). Therefore, the perturbation parameter is set 10^{-8} for all Jacobian approximations used in this work. The reader is referred to the seminal work of Knoll and Keyes [125] for details regarding the choice of perturbation parameter. Having an approximation for $\mathbf{A}\vec{U}$, using Eq. (3.95), we have

$$\vec{b} = \mathbf{A}\vec{U} - \vec{R}(\vec{U}) \tag{3.105}$$

where the approximation of the right hand side vector, $\vec{b}$, can be performed at any stage given the state variables $\vec{U}$ and the corresponding residual of the flow solver, $\vec{R}(\vec{U})$.

An important feature of the present convergence acceleration technique is that the matrix $\mathbf{A}$ is never computed explicitly. Instead, having the $\vec{b}$ vector, one can simply calculate the

product of $\mathbf{A\Phi}$ using Eq. (3.95) and the residual of the flow solver at each snapshot location. Thus, as an example for the first snapshot of the solution, $\phi_1 = \vec{U}_1$:

$$\mathbf{A}\phi_1 = \mathbf{A}\vec{U}_1 = \vec{R}(\vec{U}_1) + \vec{b} \tag{3.106}$$

Equation (3.102) can now be solved for the expansion coefficients that projects the solution using Eq. (3.98).

Covariance-Based Acceleration

In the framework of the POD-based convergence acceleration or stabilization, Markovinović and Jansen [144] and Ekici et al. [62] propose an alternative approach in which the covariance matrix of

$$\tilde{\mathbf{\Phi}} = \begin{bmatrix} | & | & & | \\ \vec{U}_1 - \overline{\vec{U}} & \vec{U}_2 - \overline{\vec{U}} & \dots & \vec{U}_M - \overline{\vec{U}} \\ | & | & & | \end{bmatrix}_{\mathrm{N\times M}} \tag{3.107}$$

is used instead of the correlation matrix where

$$\overline{\vec{U}} = \frac{1}{M}\sum_{i=1}^{M} \vec{U}_i \tag{3.108}$$

represents the mean of the snapshots. According to Markovinović and Jansen [144], a potential benefit of the subtraction of the mean is an increased level of detail in the reduced-order description in the case of near-parallel snapshot vectors.

In the case of the covariance-based acceleration, the projected solution will be defined as a linear combination of the perturbations plus the mean of the snapshots such that

$$\vec{U}_{\text{projected}} = \tilde{\mathbf{\Phi}}\,\vec{\xi} + \overline{\vec{U}} \tag{3.109}$$

Therefore, similar to Eq. (3.99), the coefficients ξ are chosen such that the residual vanishes:

$$\vec{R}(\tilde{\mathbf{\Phi}}\vec{\xi} + \overline{\vec{U}}) = 0 \tag{3.110}$$

which means

$$\mathbf{A}\tilde{\mathbf{\Phi}}\vec{\xi} = \vec{b} - \mathbf{A}\overline{\vec{U}} \tag{3.111}$$

Knowing that $\vec{R}(\overline{\vec{U}}) = \mathbf{A}\overline{\vec{U}} - \vec{b}$ and pre-multiplying by the transpose of the matrix of covariances, the reduced-order system of equations to solve for the expansion coefficients can be written as

$$\tilde{\mathbf{\Phi}}^T\mathbf{A}\tilde{\mathbf{\Phi}}\vec{\xi} = -\tilde{\mathbf{\Phi}}^T\vec{R}(\overline{\vec{U}}) \tag{3.112}$$

Once again the matrix $\mathbf{A}$ is not required to be computed explicitly. The procedure here starts by first setting $\vec{U} = \overline{\vec{U}}$ and running the CFD solver for one iteration to obtain $\vec{R}(\overline{\vec{U}})$. Thus, with the residuals of the flow solver at each snapshot stage, the following matrix-vector product

$$\mathbf{A}\tilde{\phi}_1 = \mathbf{A}(\vec{U}_1 - \overline{\vec{U}}) = \mathbf{A}(\vec{U}_1) - \mathbf{A}(\overline{\vec{U}}) = \vec{R}(\vec{U}_1) + \vec{b} - (\vec{R}(\overline{\vec{U}}) + \vec{b}) = \vec{R}(\vec{U}_1) - \vec{R}(\overline{\vec{U}}) \tag{3.113}$$

can be written for the first entry of the covariance-based matrix.

Other entries of the covariance matrix can be calculated in a similar fashion. An important advantage of using the covariance matrix instead of the correlation matrix is that it is no longer required to approximate the $\vec{b}$ vector. Also, as will be shown in the results section, the covariance-based acceleration is significantly more robust when the solver is in the linearly convergent stage with the global residuals getting close to the machine zero.

The present ROM-based acceleration method can be summarized as follows:

1- A set of M solutions and their corresponding residual vectors are stored with a user-specified sampling frequency over a specified length of acceleration cycle.

2- Correlation or covariance matrices are set up.

3- In the case of correlation-based acceleration, the $\vec{b}$ vector is approximated using the matrix-free perturbation method. In the case of the covariance-based acceleration, the

solver is run for one iteration using the mean solution as the initial condition and the residual is stored.

4- Depending on whether it is desired to have correlation or covariance-based acceleration, either Eq. (3.102) or Eq. (3.112) is solved for the expansion coefficients which are then used for the calculation of the projected solution.

3.10.3 Snapshot Selection

As explained earlier, the idea of using the solution snapshots as the basis vectors without an orthogonalization requires a careful selection process. In this section, some guidelines are presented that can be useful in selecting the set of best snapshots for the projection.

To begin, Djeddi et al. [54] have shown that taking the first snapshot after two orders of magnitude of drop in the maximum residual results in good acceleration. This is to make sure that the flow solver has reached the linearly convergent phase. The next issue would be to determine the span of the acceleration cycle over which the snapshots are taken and at the end of this cycle the projection process is performed. It has been shown that once the flow solver is linearly converging and depending on the convergence rate, 1 or 2 orders of magnitude drop in the global residual can be taken as the span of the acceleration cycle [54]. UNPAC solver automatically monitors the convergence rate and can easily determine the span of the acceleration cycle over which the nominal solver reaches 1 or 2 orders of magnitude drop in its global residual.

It has also been found [54] that, for the inviscid and subsonic cases, 10-15 snapshots is usually enough to capture the necessary flow features for a successful projection. On the other hand, transonic cases and *high*-Re flow cases that usually exhibit a more oscillatory or very slow convergence would require 20 or more snapshots for robust projections. The snapshot interval is then determined from the span of the acceleration cycle and the number of snapshots used. It must be noted here that in general, the performance of the acceleration technique is enhanced as the number of snapshots is increased.

3.11 Grid Adaptation[2]

In this work, an r-refinement adaptive mesh redistribution (AMR) technique is used that can relocate nodes to have clustering in the regions with high flow gradients and curvatures [52]. The ball-vertex approach is utilized which is directly related to the spring analogy where all edges connected to each node are represented by a linear spring that can control the stiffness of that node. Also, virtual edges are defined to provide virtually-added stiffness that can resist mesh entanglement and cell inversion. Based on Hooke's law, the resistance force can be expressed in terms of a spring stiffness and displacement according to

$$\vec{F} = K\vec{\Delta x} \tag{3.114}$$

where K is the total stiffness at each node, $\vec{F}$ is the force vector, and $\vec{\Delta x}$ is the vector of node displacement given by

$$\vec{\Delta x} = \delta x \cdot \hat{i} + \delta y \cdot \hat{j} + \delta z \cdot \hat{k} \tag{3.115}$$

in three-dimensions. In the ball-vertex approach applied to node i, all edges that connect this node to each of its node neighbors, j, are replaced by a stiffness, k_{ij}. This stiffness will be an inverse function of the edge length, L_{ij}, according to

$$k_{ij} = \frac{1}{(L_{ij})^n} \tag{3.116}$$

where n is normally taken to be 2, and is found to work well in CFD applications [50, 235]. Generally speaking, when a node is moved such that it passes through its opposite face (or diagonal in the case of a quadrilateral cell), a concave element is produced that is the source of mesh entanglement. Therefore, in order to further prevent the cell inversion, a virtual stiffness is added to each node that is defined based on a virtual linear spring. The length

[2]This section, in part, is a reprint of the material as it appears in AIAA Paper 2018-3245 titled "*An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers*" (2018). Authors: **Reza Djeddi** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi** and Kivanc Ekici.

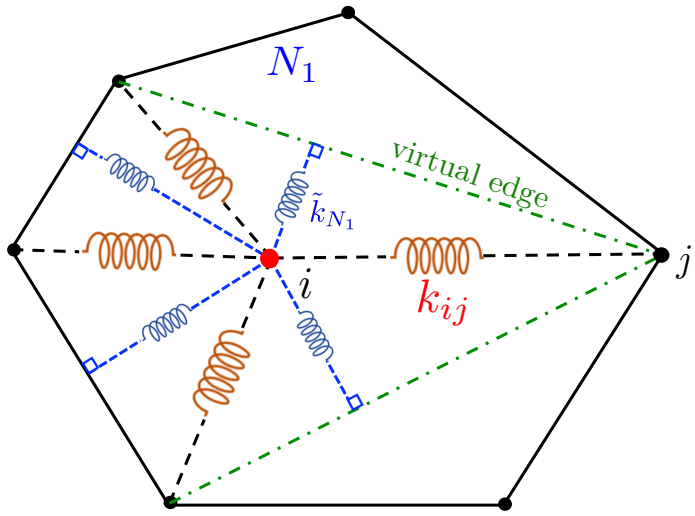


Figure 3.10: Physical and virtual edges with the corresponding springs for the grid node i and its neighboring nodes j and neighboring cells N_i.

of this spring would be the distance between the node and its opposite face (in triangular cells) or its opposite diagonal (in quadrilateral cells).

This stiffness is calculated for each cell neighbor of the grid node. As shown in Figure 3.10, for node i, the virtual stiffness is defined by $\tilde{k}_{N_1}$ where N_1 refers to the first cell neighbor of node i. It is worth noting that the stiffness of the virtual edge with length L_{N_1} is calculated based on the same formula given by Eq. (3.116). Finally, the total stiffness at each grid node, i, can be defined as a sum of physical and virtual stiffness values:

$$K_i = \sum_{j=1}^{\mathrm{Ngb}_i} k_{ij} + \sum_{j=1}^{\mathrm{CNgb}_i} \tilde{k}_{N_j} \tag{3.117}$$

where Ngb_i and CNgb_i are the number of node and cell neighbors of node i, respectively.

As discussed earlier, UNPAC uses primarily an edge-based data structure. Therefore, using a single loop over all edges ij, the linear stiffness values, k_{ij}, can be calculated. Next, for each node, another loop is performed over all cell neighbors to calculate the distances, L_{N_c}, where c is the number of cell neighbors. This leads to the virtual stiffness $\tilde{k}_{N_c}$ according to Eq. (3.116). Preliminary results have shown that even with the addition of the virtual edges in the framework of the ball-vertex approach, mesh entanglement can occur in cases

with extreme node relocation. Therefore, a local relaxation technique based on a collapse length [119] is used in this work that requires the minimum edge lengths (physical and virtual) to be defined at each node. In this regard, during the process of calculating the physical and virtual stiffnesses, the minimum edge lengths at each grid node are also calculated and stored in the data structure to be used later. The local relaxation technique will be discussed in detail later in this section.

In the r-adaptive AMR, a driving force is needed to control the node relocation process. As explained before, the goal of the AMR approach is to cluster nodes around regions where the gradient and curvature of a certain flow variable is high. Thus, a natural choice for the AMR driving force would be to use the gradients and curvatures of flow variable along each edge of the numerical grid. Assuming that the flow variable of interest, ϕ, and its gradient, $\vec{\nabla}\phi$, are defined at each node, the gradient and curvature forces can be calculated on edge ij according to

$$\begin{aligned} \vec{F}_{ij,\,\text{gradient}} &= \frac{|\phi_i - \phi_j|}{|\vec{x}_i - \vec{x}_j|}\vec{r}_{ij} \\ \vec{F}_{ij,\,\text{curvature}} &= \frac{\left\|\left(\vec{\nabla}\phi_i - \vec{\nabla}\phi_j\right) \circ \vec{r}\right\|}{|\vec{x}_i - \vec{x}_j|}\vec{r}_{ij} \end{aligned} \tag{3.118}$$

where $\vec{r}_{ij}$ is the vector connecting node i to j and $\circ$ is the entry-wise or Hadamard product operator. Jones [119] has used a clamped cubic spline approach for curvature reconstruction in order to improve the accuracy of the curvature forces. However, our numerical tests have shown that a simple approximation based on the gradient information at grid nodes (3.118) can lead to identical results. These gradient and curvature forces are applied to node i while a similar force with the same magnitude but in opposite direction will be applied to the other end of each edge such that

$$\vec{F}_{ji,\,\text{gradient}} = -\vec{F}_{ij,\,\text{gradient}} \quad \text{and} \quad \vec{F}_{ji,\,\text{curvature}} = -\vec{F}_{ij,\,\text{curvature}} \tag{3.119}$$

Having the gradient and curvature forces calculated at each edge, the forces can be distributed to each grid node to obtain the total force vector. In doing so, the gradient and curvature forces can be scaled by their corresponding factors so that

$$\vec{F}_{\text{total}} = C_{f,\,\text{gradient}}\,\vec{F}_{ij,\,\text{gradient}} + C_{f,\,\text{curvature}}\,\vec{F}_{ij,\,\text{curvature}} \tag{3.120}$$

where $C_{f,\,\text{gradient}}$ and $C_{f,\,\text{curvature}}$ are some user-defined force coefficients that can control the effects of gradient and curvature forces. It must be noted that before each AMR cycle, the selected flow variable and its gradients are smoothed using a pseudo-Laplacian smoother [21]. Our numerical tests have shown that a smooth force field can result in better adapted-grid qualities and a more efficient AMR process. Finally, having the force vectors given by Eq. (3.118) and the total stiffness described in Eq. (3.117) calculated at all nodes, the displacements at each node can be calculated according to Eq. (3.114). Thus, with the calculated displacements, the nodal locations can be updated according to

$$\vec{x}_i^{\,\text{new}} = \vec{x}_i^{\,\text{old}} + \vec{\Delta x}_i \tag{3.121}$$

As discussed before, even with the addition of the virtual edges in the ball-vertex approach, it cannot be guaranteed that the r-adaptive AMR technique would not lead to mesh entanglement and cell inversion. In fact, for cases with sharp gradients and those with strong discontinuities due to shocks, severe node relocation can still introduce invalid elements in the adapted grid. Therefore, in this work, a local relaxation approach similar to that presented by Jones [119] is used to further control the node movements. Assuming that the minimum edge lengths (physical and virtual) at node i are given by $L_{i,\,\text{min}}$, and the magnitude of the calculated displacement at this node is greater than $L_{i,\,\text{min}}$, the displacement must be limited by using a local relaxation factor in the update formula given in Eq. (3.121). Here, a safety factor λ is used to further shrink the safe displacement zone. Finally, the calculated displacement can be modified except for cases where the original displacement does not violate the safety zone:

$$\vec{\Delta x}_i^{\,\text{corrected}} = \frac{\lambda\, L_{i,\,\text{min}}}{\|\vec{\Delta x}_i\|}\,\vec{\Delta x}_i \quad \text{only if} \quad \|\vec{\Delta x}_i\| > \lambda L_{i,\,\text{min}} \tag{3.122}$$

The above correction is incorporated into the update formula as a local relaxation parameter ω_i such that

$$\vec{x}_i^{\,\text{new}} = \vec{x}_i^{\,\text{old}} + \omega_i \; \vec{\Delta x}_i \tag{3.123}$$

where

$$\omega_i = \begin{cases} \frac{\lambda \; L_{i,\,\min}}{\|\vec{\Delta x}_i\|} & \text{if} \quad \|\vec{\Delta x}_i\| > \lambda L_{i,\,\min} \\ 1 & \text{otherwise} \end{cases} \tag{3.124}$$

The update process described in Eq. (3.123) needs to be executed iteratively until nodes have reached a certain level of equilibrium which can be determined by calculating the norm of the effective displacements.

In the framework of a CFD solver, an important feature of the r-adaptive AMR would be to preserve the original aerodynamic geometry while effectively moving and clustering nodes along the boundaries. In order to force the boundary nodes to move only tangential to the boundaries, the normal projections of the calculated displacements at each boundary node must be killed. This correction can be done for boundary node b via

$$\vec{x}_b^{\,\text{bndy correction}} = \vec{x}_b - \left(\vec{x}_b \cdot \hat{\vec{S}}_b\right) \hat{\vec{S}}_b \tag{3.125}$$

where $\hat{\vec{S}}_b$ is the unit normal vector (outward) at each boundary node b. Since the update process is repeated iteratively, the boundary normal vectors must be recalculated at each iteration to make sure correct normal vectors are being used to fix the boundary node displacements.

Although the process described here can be effective in guaranteeing that the boundary nodes move along the boundaries and not normal to them, it would not be essentially shape preserving. Therefore, in order to preserve the original geometry that was described by the unadapted grid, a parameterization process is used in this work. Thus, to maintain the original geometry, boundary nodes are parameterized as

$$\vec{x} = \vec{x}(s)$$
$$\vec{y} = \vec{y}(s) \tag{3.126}$$

where the parameterization variable s is taken to be the normalized arc-length of the sorted boundary nodes with $0 \leq s \leq 1$. Finally, a piecewise cubic spline is used to fit the parameterization data in each Cartesian coordinate. Therefore, at the end of the node relocation process, the final nodal locations at the boundary nodes are corrected according to the parameterized functional forms of Eq. (3.126). Additionally, the nodes at the trailing edge or any other point that defines a sharp corner are fixed to avoid having these regions rounded during the parameterization process that can alter the geometrical features of the aerodynamic body.

Following the r-adaptive AMR process, it is possible to have highly skewed cells even to the point where the flow solver might face convergence issues or even diverge. This problem can be remedied using a grid smoothing process that can follow the AMR process. In this work, a Laplacian grid smoother [21] is used to increase the quality of the adapted grid. It must be noted that during the Laplacian smoothing process, boundary nodes are allowed to move along the boundary similar to what was described earlier for the AMR process [see Eq. (3.125)]. Moreover, a similar geometry preserving process using the parameterization data is ultimately applied to guarantee that the smoothing process does not alter the original shape of the aerodynamic body.

In the framework of the HB solver, the AMR process is applied to each sub-time level individually using the solution and the gradients at each of those time instances. Therefore, the r-adaptive AMR can effectively cluster nodes at different sub-time levels according to the instantaneous flow solution, location of the discontinuities and shocks or the separation zones. Finally, the adapted grid is once again pre-processed to calculate the new metrics and cell volumes. This is a great feature of the r-adaptive approach where there would be no change in the nodal connectivities, number of nodes/edges and the data structure. During the pre-processing of the adapted grid, the grid velocities are also updated according to

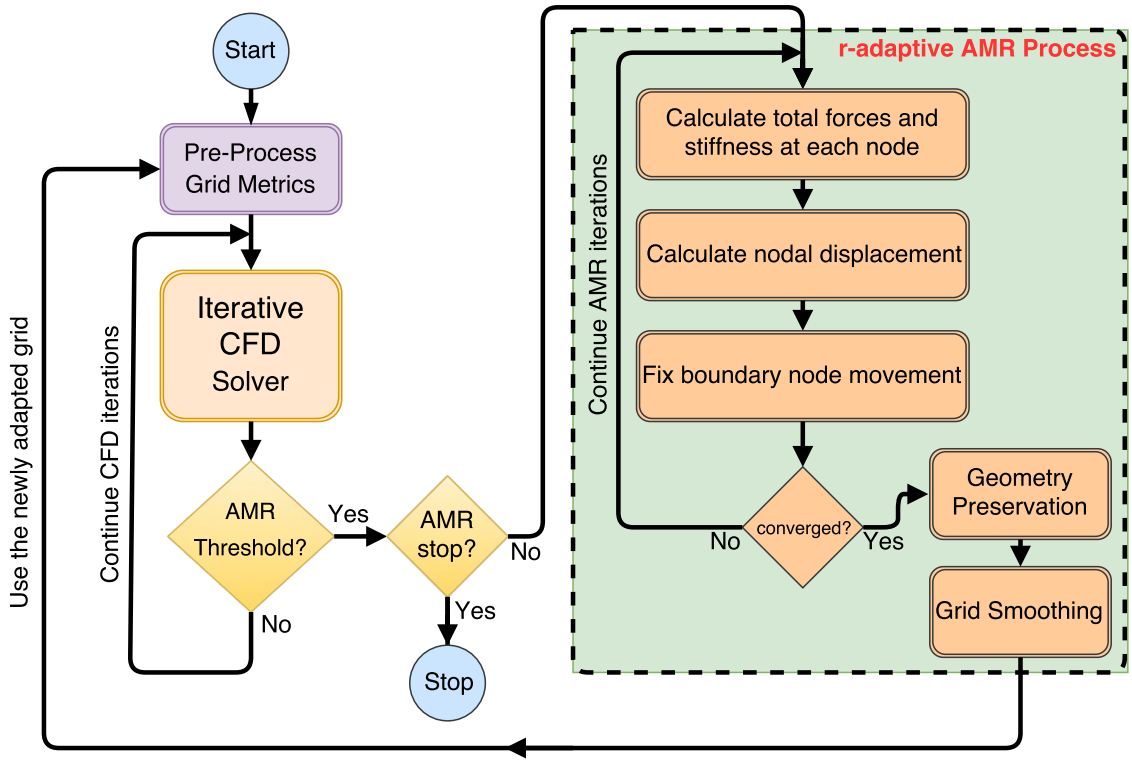


Figure 3.11: Flowchart of the r-adaptive AMR approach coupled with the nominal CFD solver.

Eq. (3.53) based on the new nodal locations. Having the updated grid velocities, the GCL errors given in Eq. (3.55) are also recalculated.

After the flow solver is converged to a certain level, i.e., AMR threshold, the AMR process is initiated. The overall r-adaptive AMR process is provided below:

1. If requested, a pre-smoothing process is applied to the selected flow variable and its gradient.

2. Using the selected flow variable, gradient and curvature forces are calculated at each edge and are distributed to the two end nodes.

3. Physical and virtual stiffness values are calculated for each node by one loop over all edges and another over all cell neighbors. The total stiffness is accumulated at all grid nodes.

4. Using the minimum edge length at each node and the user-defined safety factor, the local relaxation coefficient is calculated according to Eq. (3.124).

5. Displacements at the boundary nodes are corrected so that nodes would only move along the boundaries and not normal to them.

6. The nodal locations are updated using Eq. (3.123).

7. Iterations are stopped if the convergence criterion is met or the maximum number of iterations is reached, otherwise the process is repeated by going back to step 2.

8. The geometry preserving process is applied based on the pre-calculated parameterization data according to the unadapted mesh.

9. If requested, a user-defined number of Laplacian grid smoothing iterations are performed with a user-defined smoothing coefficient.

10. Grid metrics, median-dual control volumes, grid velocities, and GCL errors (for HB solver) are recalculated before restarting the CFD solver with the newly adapted grid.

The process described here follows the flowchart shown in Figure 3.11.

3.12 Parallelization

As engineering simulations keep growing in size and the extent of flow features that need to be captured are increasing, it is necessary to focus our attention to developing fast and efficient CFD solvers. From the perspective of numerical schemes involved in an iterative solver, computational efficiency can be achieved using convergence acceleration techniques which were discussed in Section 3.7.

As far as the design of the computer code is concerned, parallel processing would be another option to improve the computational efficiency. With the advancements in computer science and the fact that high performance computing resources are becoming more and more readily available, parallel solvers are replacing sequential solvers that have been conventionally developed. The UNPAC solver developed in this work is parallelized using an advanced distributed-memory model and the details of the parallelization process are presented next.

3.12.1 Domain Decomposition

The basic of idea of parallel computing for a numerical solver is to break down the computational domain into multiple sub-domains over which the computational task is handled by an individual processing unit (or core). This partitioning process is usually done using a method called domain decomposition (DD) which, depending on how the inter-facial boundary of the two adjacent sub-domains is handled, can be categorized as: (1) overlapped-DD and (2) non-overlapped-DD. In the former approach, neighboring sub-domains overlap each other by sharing a certain number of control volumes. While this can be beneficial to the numerical accuracy of the parallel solver due to the strong coupling of solutions between sub-domains, it can spoil the performance of the parallel solver by introducing redundant computations and communications [236]. On the other hand, in the non-overlapped-DD approach, the two adjacent sub-domains only share faces of the control volumes at the inter-facial boundaries which reduces the amount of communications in between sub-domains. However, this process involves the introduction of "*ghost*" nodes that are required for the

calculation of fluxes across the inter-facial boundaries. In this work, a non-overlapped-DD approach is utilized due to its computational efficiency.

When decomposing a computational domain, one must make sure that all processors handle an equal (or almost-equal) share of computational load. This means that the number of cells (in a cell-based approach) or the number of edges (in an edge-based approach) are in the same order for all sub-domains. The parallel performance can be negatively affected (in some cases very severely) due to unbalanced partitioning [203]. While domain decomposition in structured grids can be straightforward, the partitioning of an unstructured grid is a much more elaborate task. Fortunately, there are many non-commercial partitioning tools available that can be used for this purpose. UNPAC solver adopts the METIS [123] partitioning package due to its performance and ease of use.

In the next section, the parallel process used in the UNPAC solver is described and the details of the non-overlapped-DD approach, definition of the ghost nodes, and the communications between adjacent sub-domains are discussed.

3.12.2 Parallelization of UNPAC

The parallelization process in the UNPAC solver follows a "*single program multiple data*" (SPMD), also known as single instruction multiple data (SIMD), approach. In this approach, a set of processors or computing cores perform the exact same instructions on different sets of data [193] which can clearly ease up the parallelization process as the solution algorithm would be identical for all processing units.

As discussed earlier, the non-overlapped-DD approach introduces "*ghost*" nodes at the inter-facial boundaries. This is shown in Figure 3.12 for a hybrid unstructured grid using the median-dual control volume approach. The partitioning process performed by METIS, generates the nodal partitions with the most balanced number of nodes allocated to each sub-domain. As can be seen in Figure 3.12, the two median-dual control volumes share a face at the inter-facial boundary. When decomposed, each control volume is handled by a different processor. In the sub-domain to the left, the finite-volume method requires the calculation of the total residuals for the control volume $\mathcal{V}_{p_5}$. Here, node p_5 is called the "*core*" node of the left sub-domain as it physically belongs to it. Calculation of the residuals for the

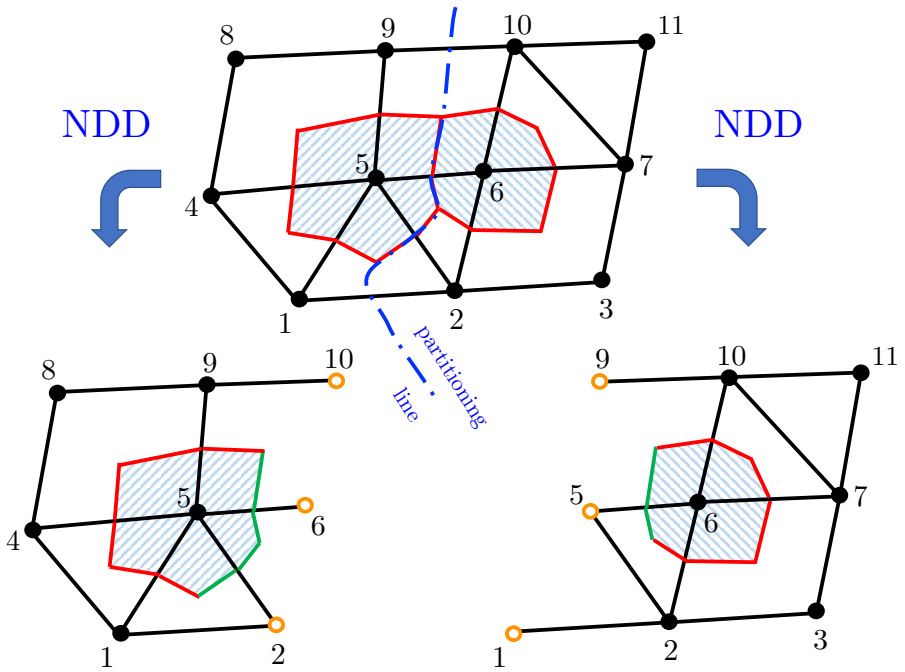


Figure 3.12: Non-overlapped domain decomposition (NDD) used in the parallelization of the UNPAC solver. Shown here are the decomposition of the computational domain into two adjacent sub-domains, definition of the "ghost" nodes (hollow circles), and the median-dual control volumes on each sub-domain.

core node p_5 involves evaluating the flux at edge p_5p_6 although p_6 does not physically belong to the left partition. Therefore, node p_6 is added as a *ghost* node to the left sub-domain. Due to the edge-based structure of the UNPAC solver, the addition of the *ghost* node p_6 automatically generates the edge p_5p_6 in the left sub-domain. Similarly, node p_5 is also added to the right sub-domain as one of its *ghost* nodes. In fact, the ghost nodes of any sub-domain are actually the duplicate copies of the core nodes of its adjacent partitions.This process does not disturb the balanced partitions since *ghost*-typed node-edge pairs are distributed evenly between adjacent sub-domains.

During the iterative process, different types of information are required at *ghost* nodes. These include conservation variables, limiter functions (for second order Roe scheme), and flow gradients (for second order Roe scheme and/or viscous flow cases). This information is communicated using the message passing interface (MPI) protocol. Before starting the iterative process, all the connectivity data and *core-ghost* associations are determined on all

computing cores. Additionally, the entire nodal data that needs to be transferred from one sub-domain to the other is packed in a point-to-point message to reduce the communication cost. This requires a complex data structure that is developed as part of the parallel pre-processing.

It must be noted that solutions at the *ghost* nodes can be communicated in a "non-lagged" or a "lagged" manner. In the former approach, information is communicated during each RK stage while in the latter, the communication is postponed until after each iteration. Obviously, the "non-lagged" approach will result in a much higher parallel accuracy although it significantly increases the number of Send/Receive calls to MPI functions. On the other hand, the "lagged" approach can lead to slight inaccuracies due to parallelization especially in cases where the inter-facial boundaries are crossing regions of large gradients (e.g. shocks or boundary layer). Also, the "lagged" communication slightly affects the convergence rate of the parallel solver due to delayed update of the solution at the *ghost* nodes. Overall, the non-overlapped-DD approach implemented in the UNPAC solver using both non-lagged or lagged techniques preserves essentially the same accuracy as the sequential solver. It must be added that, irrespective of the communication method used, the solution at the *ghost* nodes is not updated (is frozen) and the data flow direction is always from *core* nodes to *ghost* nodes and not the other way around.

Chapter 4

Sensitivity Analysis using FDOT Toolbox

As discussed in Chapter 1, the goal of this work is to employ the gradient-based optimization technique for aerodynamic shape optimization of wind turbine blades. This process relies on the efficient and accurate calculation of the gradients or sensitivities of the objective function (goal) to the design variables. In this dissertation, a new paradigm for the computation of the discrete adjoint sensitivities is introduced. This novel approach directly addresses the inherently large memory footprint required by existing OO/AD tools. The details about the method and its underlying algorithm are discussed in this chapter.

4.1 Adjoint Sensitivity Analysis in CFD

In a CFD solver, the discretized form of the governing equations can be written in terms of a residual operator $\vec{R}(\vec{x}, \vec{U}(\vec{x}))$ with $\vec{x}$ and $\vec{U}(\vec{x})$ being the vector of design variables and flow solutions, respectively. This residual vector is driven to zero using a time-marching scheme to reach steady-state solution. At the fully converged state, the total derivative of the residual with respect to the design variables must be zero, i.e.,

$$\frac{d\vec{R}}{d\vec{x}} = \frac{\partial \vec{R}}{\partial \vec{x}} + \frac{\partial \vec{R}}{\partial \vec{U}} \frac{d\vec{U}}{d\vec{x}} = 0 \tag{4.1}$$

In the framework of the gradient-based optimization approach, it is required to have the total derivative of the objective function that needs to be minimized with respect to the design variables. Thus, for an objective function, $I(\vec{x}, \vec{U}(\vec{x}))$, the gradient information can be calculated using the chain rule so that

$$\frac{dI}{d\vec{x}} = \frac{\partial I}{\partial \vec{x}} + \frac{\partial I}{\partial \vec{U}} \frac{d\vec{U}}{d\vec{x}} \tag{4.2}$$

The cost of evaluating $\frac{d\vec{U}}{d\vec{x}}$ increases linearly with the number of design variables which makes it impractical to use a direct approach in sensitivity analysis. However, by rearranging Eq. (4.1) such that

$$\frac{d\vec{U}}{d\vec{x}} = -\left[\frac{\partial \vec{R}}{\partial \vec{U}}\right]^{-1} \frac{\partial \vec{R}}{\partial \vec{x}} \tag{4.3}$$

which upon substituting into Eq. (4.2) yields

$$\frac{dI}{d\vec{x}} = \frac{\partial I}{\partial \vec{x}} - \frac{\partial I}{\partial \vec{U}} \left[\frac{\partial \vec{R}}{\partial \vec{U}}\right]^{-1} \frac{\partial \vec{R}}{\partial \vec{x}} \tag{4.4}$$

one can lay the groundwork for the adjoint approach. Therefore, the following adjoint equation is solved for the adjoint vector, ψ

$$\psi^T = -\frac{\partial I}{\partial \vec{U}} \left[\frac{\partial \vec{R}}{\partial \vec{U}}\right]^{-1} \tag{4.5}$$

This is the essence of the continuous adjoint approach. Finally, having the adjoint solution vector, the sensitivities of the objective function with respect to the design variables can be computed using

$$\frac{dI}{d\vec{x}} = \frac{\partial I}{\partial \vec{x}} + \psi^T \left(\frac{d\vec{U}}{d\vec{x}}\right) \tag{4.6}$$

In the discrete adjoint method, the nominal CFD solver can be viewed as a series of m function evaluations that are applied to a vector of initialized solution $\vec{U}_0$ based on $\vec{x}$ (the design variables) such that

$$\vec{U}_0 = f_0(\vec{x}), \quad \vec{U}_1 = f_1(\vec{U}_0), \quad \vec{U}_2 = f_2(\vec{U}_1) \quad \dots \quad \vec{U}_m = f_m(\vec{U}_{m-1}) \tag{4.7}$$

where $\vec{U}_m$ is the fully converged solution vector that the CFD solver outputs. Eventually, this final flow solution is used to evaluate an objective function, I, via a vector operator $\vec{v}$ operating on $\vec{U}_m$

$$I = \vec{v}^T \vec{U}_m \tag{4.8}$$

In order to calculate the sensitivities of the objective function to the input variables, the chain rule of differentiation can be applied to Eq. (4.8) resulting in

$$\nabla I = \vec{v}^T \left[\frac{\partial f_m}{\partial \vec{U}_{m-1}}\right] \left[\frac{\partial f_{m-1}}{\partial \vec{U}_{m-2}}\right] \quad \dots \quad \left[\frac{\partial f_1}{\partial \vec{U}_0}\right] \left[\frac{\partial f_0}{\partial \vec{x}}\right] \tag{4.9}$$

In the above equation, the derivatives of the intermediate functions with respect to the design variables propagate from $\vec{U}_0$ toward $\vec{U}_m$ which results in the so-called "forward" mode. This process involves m matrix-matrix multiplications and a single matrix-vector product at the last step. When the number of design variables is large, which happens to be the case in aerodynamic design optimization applications, the cost of evaluating the m matrix-matrix multiplications grows linearly, leading to high computational costs for sensitivity analysis. Alternatively, without loss of generality, one can transpose Eq. (4.9) to obtain

$$\nabla I^T = \left[\frac{\partial f_0}{\partial \vec{x}}\right]^T \left[\frac{\partial f_1}{\partial \vec{U}_0}\right]^T \left[\frac{\partial f_2}{\partial \vec{U}_1}\right]^T \quad \dots \quad \left[\frac{\partial f_m}{\partial \vec{U}_{m-1}}\right]^T \vec{v} \tag{4.10}$$

which requires only m matrix-vector multiplications, resulting in a significantly lower computational cost. It must be noted that the gradients obtained using the discrete adjoint approach are generally consistent with the finite-difference approximations, and therefore, lead to slightly more accurate sensitivity information compared to the continuous approach [139]. Apparently, in the discrete analysis the derivatives are propagated in the reverse direction. However, this requires the complete time history of the flow equations to be stored which can lead to huge memory footprints in a large scale three-dimensional flow

solver. In any case, this issue can be addressed in a simple way, which is one of the novel aspects of the method presented in this dissertation.

4.2 Automatic Differentiation Using FDOT

Since many CFD solvers (such as steady and unsteady harmonic balance or time-spectral) follow an iterative process to reach a converged flow solution, the memory requirements for recording the entire expression tree can easily go out of hand after many iterations, especially with the addition of auxiliary set of equations such as complex turbulence and transition models. However, this issue can be addressed in a very simple way by using the repetitive iterative process inherent in these CFD solvers. Assume that the nominal CFD solver has converged to machine accuracy for a given set of design (or input) variables. At this point, further iterations do not change the values of conservation and primitive variables and the expression tree starts following the exact same path at each iteration since the solution is fully converged. By taking advantage of this feature, the expression tree at the fully converged state can be recorded in a *tape*. This is done for a single sweep of the nominal solver (pre-processing plus one iteration of the CFD solver plus post-processing) by loading a fully converged solution. This leads to a very short tape that can easily fit in the random access memory (RAM). The adjoint solutions are then obtained by repeatedly playing the recorded tape in reverse until the desired level of accuracy in the gradient information is achieved. The reverse nature of these computations makes the method equivalent to the discrete adjoint approach. As already mentioned, this method requires the fully converged nominal solution before the discrete adjoint computations can be carried out. However, this is not a drawback since the same is true for traditional discrete (SCT or hand-written) or continuous adjoint computations. This method can easily handle even the most complicated three-dimensional solvers in a fully-automated fashion. Furthermore, the memory footprint becomes very manageable even on most modern workstations. The novel approach is implemented into a toolbox called FDOT (**F**ast automatic **D**ifferentiation using **O**perator-overloading **T**echnique), and is written for both C/C++ and Fortran programming languages.

In this section, the methodology that constitutes the foundation of all available OO/AD tools will be introduced first. In doing so, the challenges faced in using these tools will be identified and discussed. Next, the novel approach proposed in this work will be explained in detail and the implementation of the toolbox as well as its black-box-type application to a simple CFD solver will be presented.

4.2.1 Reverse Mode of AD Using Operator Overloading

In the reverse mode of automatic differentiation the derivatives of the objective function with respect to all design and intermediate variables, also known as *adjoints*, are calculated via an evaluation process. As has been shown by Wolfe [229], and Baur and Strassen [16], the reverse mode of AD is capable of evaluating the gradient of the output with respect to a large number of design variables (or inputs) with a computational cost that is only a small multiple of that of the primal solver that calculates the value of the function itself. Once again, this is what makes the reverse mode of AD attractive for design optimization problems. However, the implementation of the reverse mode is not as straightforward as the forward mode of AD and, especially for the operator overloading approach, a lot of meta-programming is required. In fact, for the reverse mode of AD to work, the solver has to have access to the entire set of instructions, also known as the expression tree, that are executed from the inputs all the way to the output. Almost all OO/AD tools achieve this by recording the entire expression tree into a derived type class often called the *tape* that is stored in the RAM. This tape not only stores the resulting value of each unary or binary operation together with all assignments, but also stores the index of the first and second (required only for binary operations) arguments, operation type and the adjoint value of the result. The reverse mode of AD defines an adjoint object for each variable involved in any unary or binary operation. As an example, for a binary operation, f, performed on two variables a and b and resulting in another variable c, three adjoint variables, $\overline{a}$, $\overline{b}$, and $\overline{c}$ will be defined. Therefore, the forward and the adjoint modes of this simple operation can be written as

$$c = f(a,b) \quad \Rightarrow \quad \begin{aligned} \overline{a} \ &+= \overline{c} * \frac{\partial f}{\partial a} \\ \overline{b} \ &+= \overline{c} * \frac{\partial f}{\partial b} \end{aligned} \tag{4.11}$$

where, for example, $\overline{a}$ and $\overline{b}$ are the adjoints of a and b, respectively. It is useful to note that, by definition

$$\overline{c} = \frac{\partial c}{\partial f} = 1.0 \tag{4.12}$$

Since the process for the reverse mode calculates all adjoints in the expression tree, one must only set the adjoint of the objective function to 1.0 while initializing the rest of the adjoints to 0.0. As an example for the above binary operation, one can set $\overline{c} = 1.0$ to calculate the adjoints of a and b variables, which in this case would simply be the partial derivatives of the output function with respect to both input variables, i.e.,

$$\overline{c} = 1.0 \quad \Rightarrow \quad \begin{aligned} \overline{a} \ &= \frac{\partial f}{\partial a} \\ \overline{b} \ &= \frac{\partial f}{\partial b} \end{aligned} \tag{4.13}$$

This simple idea is the key rule for the reverse mode of automatic differentiation. Therefore, each unary or binary operation can be overloaded such that the input arguments are indexed along with their values and the type of operation. In the reverse evaluation process, the adjoints of the input variables for each statement – or basically each entry of the expression tree – will be calculated using the concept shown in Eq. (4.11). A very important issue regarding the use of the OO-based AD tools is that, in contrast to the SCT tools where all branches, loops and if blocks are transformed, there are no branches or if blocks in the recorded tape and all of the loops are fully unrolled since the tape actually records only the operations that are being executed at run time. Therefore, in the framework of an OO-based AD tool, all of the data-dependent branches and loops can only be reversed at run time. As can be clearly seen, there are two main challenges that must be addressed in the reverse mode of AD using operator overloading which are summarized below.

1- The memory footprint for the recorded tape can easily reach beyond the available resources as the number of instructions are increased. Since all of the loops in the nominal code are unrolled in the process of recording the tape, the iterative part of the numerical solver will exponentially increase the number of tape entries thus increasing the memory requirements and the computational overhead in writing, reading and evaluating the tape. This can make the use of traditional OO-based AD tools infeasible for full-fledged CFD solvers.

2- Because the tape is a complete record of all instructions and operations that are performed, in the process of iterative convergence of the CFD solver, different branches will be taken and different loops will be unrolled. Therefore, the recorded tape can grow even larger as more iterations are unfolded. This forces the toolbox to use disk files instead of the random access memory which further slows down the read/write process of the tape.

The novel approach that is presented in the FDOT toolbox addresses all these issues and the methodology is explained in the following section.

4.2.2 Discrete Adjoint Sensitivity Analysis Using FDOT

For a numerical solver (or more specifically a CFD solver) with a set of design variables, $\mathbf{x}$, and an objective (or output) variable, c, the entire process can be broken down to a set of pre-iterative (flow initialization), iterative (flow solution) and post-iterative (computation of the cost function) procedures.

To simplify the discussion, let us assume a single design variable, x, and a set of pre-iterative procedures that can be collectively called $u(x)$. Next, an iterative process is recursively performed with only one intermediate variable, y_k, which is updated at each iteration k. This iterative process can be denoted as function $f(y_k, u, x)$ that can have multiple inputs with only one input changing throughout the iterative process. Assuming a convergent iterative process, one can assume $y_k \to y^*$ as $k \to N$ after a certain number of iterations. Finally, a set of post-iterative procedures are performed using the converged solution that leads to the cost function. This process can be viewed as another function,

$g(y^*, u, x)$, that can have multiple inputs. The output of this function would be our objective, c. In order to see how the iterative process evolves as the loops are unrolled and the instructions are being recorded, let us assume the iterative process converges after only 2 iterations. Therefore, this simple algorithm (left side of the following pseudo-code) can be written as:

Primal Solver (forward)	Adjoint Calculations (reverse)
pre-iterative part:	**post-iterative part:**
	$\overline{c} = 1.0$
x	$\overline{y_3} = \overline{y_2} = \overline{y_1} = \overline{u} = \overline{x} = 0.0$
$u = u(x)$	$\overline{u} += \partial g/\partial u * \overline{c}$
y_1 : initial guess	$\overline{x} += \partial g/\partial x * \overline{c}$
	$\overline{y_3} += \partial g/\partial y_3 * \overline{c}$
iterative part:	**iterative part:**
iteration 1: $y_2 = f(y_1, u, x)$	iteration 2: $\overline{y_2} += \partial f/\partial y_2 * \overline{y_3}$
	$\overline{u} += \partial f/\partial u * \overline{y_3}$
	$\overline{x} += \partial f/\partial x * \overline{y_3}$
iteration 2: $y_3 = f(y_2, u, x)$	iteration 1: $\overline{y_1} += \partial f/\partial y_1 * \overline{y_2}$
	$\overline{u} += \partial f/\partial u * \overline{y_2}$
	$\overline{x} += \partial f/\partial x * \overline{y_2}$
post-iterative part:	**pre-iterative part:**
$c = g(y_3, u, x)$	$\overline{x} += \partial u/\partial x * \overline{u}$

Now, using the generalized adjoint formula [Eq. (4.11)], one can calculate all adjoints of the intermediate variables, and propagate the sensitivities in the reverse order (right side of the pseudo-code given above). As discussed earlier, there are no branches or if loops involved in the recorded tape. This is due to the fact that the tape records only intrinsic operations

(e.g., assignments, multiplications, additions, trigonometric functions, etc.) that are being executed at run time. This very important feature necessitates the use of a fully converged solution of the primal solver to ensure that all following iterations go through the exact same expression tree. Therefore, the iterative process, $f(y_k, u, x, ...)$, would not change the forward solution throughout the iterative evaluation process since:

$$y^* = f(y^*, u, x, ...) \text{ with } y \to y^* \text{ as } k \to \infty \tag{4.14}$$

Thus, the iterative tape evaluation process can be generalized as

Iterative Adjoint Calculations (reverse)

for $k = N, 1, -1$

accumulation part:

iteration k: $\overline{u} += \partial f / \partial u * \overline{y_{k+1}}$

$\overline{x} += \partial f / \partial x * \overline{y_{k+1}}$

update part:

iteration k: $\overline{y_k} += \partial f / \partial y_k * \overline{y_{k+1}}$

$\overline{y_{k+1}} \leftarrow \overline{y_k}$

end

where N is normally the total number of iterations that it takes for the primal solver to converge to a certain level of accuracy. Therefore, the adjoint evaluation process depends on the level of accuracy that was prescribed in the nominal solver. Having $\overline{y_{N+1}}$ from the post-iterative process, one can initiate the above iterative procedure that leads to the converged adjoint solution $\overline{y_1}$. However, most of the time $\overline{y_1}$ will not be used in the pre-iterative process while the most important part of the above iterative loop is the accumulation of the $\overline{u}$ and $\overline{x}$ or any other active variable that belongs to the pre-iterative portion of the recorded tape. In fact, performing the iterative adjoint evaluation helps accumulating these adjoints to their correct value. Finally, at the end of the pre-iterative part of the adjoint evaluation, the

correct final adjoint value, $\overline{x}$, is obtained. This adjoint value is, in fact, the sensitivity of the output, c, with respect to the input, x, or $\overline{x} = dc/dx$. It is worth noting that the user needs to execute a checkpointing function right before and after the iterative loop to simply record the index of the tape entries. This way, it is easy to identify which portion of the recorded tape belongs to pre-iterative, iterative and post-iterative stages. Figure 4.1 demonstrates the coupling of the nominal solver to the FDOT toolbox for discrete adjoint analysis.

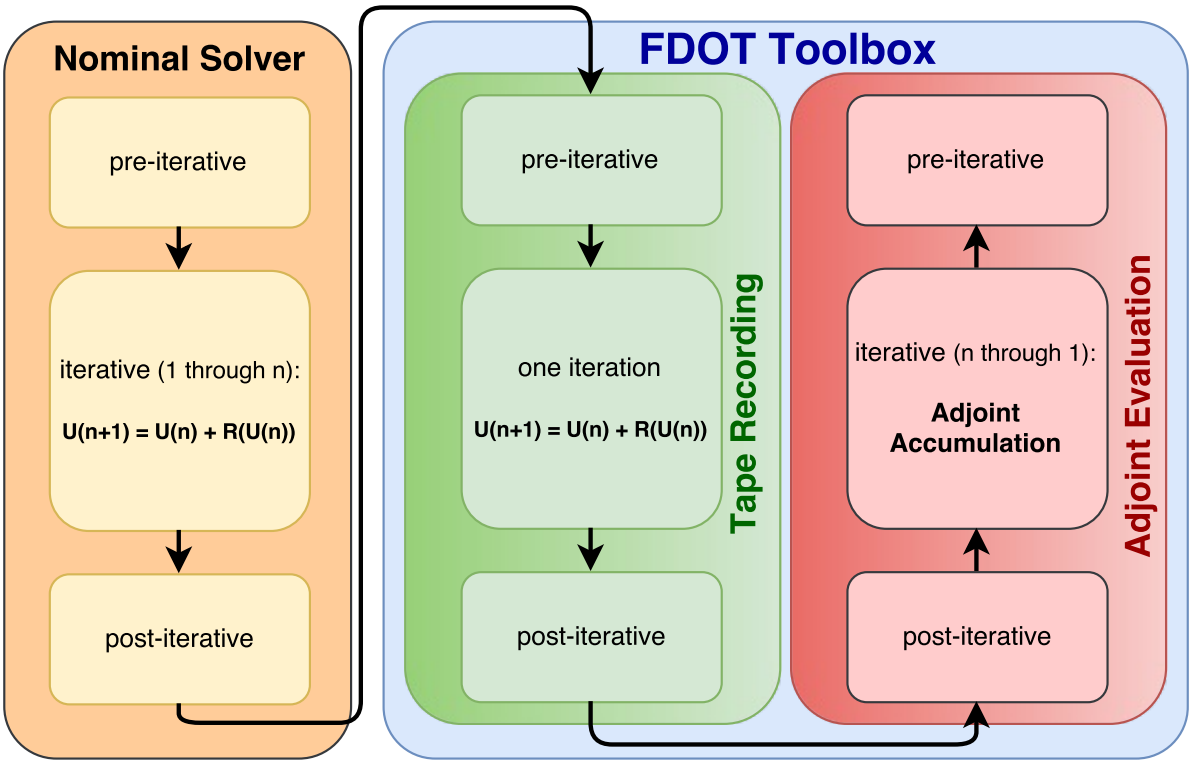


Figure 4.1: Flowchart for the FDOT toolbox and its integration into the nominal CFD solver.

4.2.3 Operator Overloading and Adjoint Evaluation

As demonstrated earlier, the present AD technique takes advantage of the iterative feature of the tape evaluation to greatly reduce the memory footprint and the computational overhead that is involved with writing and reading the recorded tape. Initially, a user defined type is introduced that takes care of the active real variables. A similar type can be defined for the active integer variables. However, the integer variables are often treated as passive

variables except for the cases where there is a combination of integer and real variables. In these situations, the integer variables are usually converted to real variables in-place before running the instructions. As an example, the definition of the new derived type is given below in Pseudo-code 4.1.

```
Type AReal
   integer   :: index
   real      :: value
End Type AReal
```

Pseudo-code 4.1: AReal derived type

As can be seen, for each **AReal** derived type variable, not only its value is stored but also an index is recorded that can be used to identify the variable's location among tape entries. It is worth noting that the precision of the floating point real numbers can be simply set using compile flags if double precision is needed for real-typed variables throughout the solver. Next, a derived type or class for the *tape* is defined which records all operations in the exact order that they are performed. As explained earlier, for each unary, binary or assignment operation this class records (1) an operation tag, (2) index of the first argument, (3) index of the second argument (only for binary operations), (4) an iterative flag, (5) a passive/active flag, (6) the primal value and (7) the adjoint value (see below).

```
Type Tape
   integer   :: optag
   integer   :: arg1
   integer   :: arg2
   logical   :: iterative
   logical   :: passive
   real      :: value
   real      :: adjoint
End Type Tape
```

Pseudo-code 4.2: Tape class

As an example, for the multiplication operator, i.e., $c = a * b$, not only the value of c is stored but also a tag is recorded to indicate that a multiplication operation has been performed. In addition, the indices of variables a and b are stored if any of the three variables is active, i.e., of **AReal** type. As explained in Section 4.2.2, the iterative adjoint procedure requires the values of the adjoints to be updated after each iteration by replacing the old adjoint values $\overline{y_{k+1}}$ by the newly evaluated ones $\overline{y_k}$. This requires flagging those iterative variables such that their indices could be stored in an array that determines which ones should have their values swapped. Therefore, an *iterative* flag is used that will be enabled for all tape entries of iterative variables and disabled for the rest of the tape.

An important feature of our OO-based AD tool is to overload all intrinsic operations. For this reason, the following operators are overloaded in Fortran to achieve in-situ tape recording while calculating each overloaded operation:

1- **Unary operators:** absolute ($|\cdot|$), square-root ($\sqrt{\cdot}$), sin, cos, tan, arcsin, arccos, arctan, log, $\log_{10}$, exp, sinh, cosh, tanh

2- **Binary operators:** $+$, $-$, $*$, $/$, $**$, ATAN2, max, min

3- **Logical operators:** $=$, $\neq$, $>$, $\geq$, $<$, $\leq$

4- **Special operators:** MATMUL, DOT_PRODUCT, MINVAL, MAXVAL. [Intrinsic functions in Fortran and their equivalents in C++ standard template library (STL).]

As examples of unary and binary operations, the overloaded versions of sine and multiplication operators in Fortran are provided below.

```
1   Function FDOT_SIN(X) RESULT (Y)
2      Type(AReal), intent(in)   :: X
3      Type(AReal)               :: Y
4
5      Tape(index_counter).optag  = SINTAG
6      Tape(index_counter).arg1   = X.index
7      Tape(index_counter).value  = sin(X.value)
8
9      Y.value = sin(X.value)
10     Y.index = index_counter
11     index_counter = index_counter + 1
12
13  End Function FDOT_SIN
```

Pseudo-code 4.3: Overloaded Sine (unary) operator (line 9) plus recording a new tape entry

As all operations are overloaded for the derived type arguments, one must also ensure that various combinations of **Real** and **AReal** as well as **Integer** and **AReal** operations are handled properly. The important issue here is that a tape entry for the passive real or integer variable must be created before continuing with the overloaded operation in order to have the correct set of instructions recorded in the tape. However, these additional tape entries are passive and their adjoints are not needed. Therefore, a logical flag is enabled for these entries so that in the process of evaluating the tape, they can be skipped. This process can greatly increase the efficiency of adjoint evaluations. It is worth mentioning that a similar procedure is performed for assignment operations for cases where the right-hand-side is of type **Real** or **Integer** while the left-hand-side is of type **AReal**.

```
1   Function FDOT_MUL(X,Y) RESULT (XY)
2      Type(AReal), intent(in)   :: X, Y
3      Type(AReal)               :: XY
4
5      Tape(index_counter).optag  = MULTAG
6      Tape(index_counter).arg1   = X.index
7      Tape(index_counter).arg2   = Y.index
8      Tape(index_counter).value  = X.value * Y.value
9
10     XY.value = X.value * Y.value
11     XY.index = index_counter
12     index_counter = index_counter + 1
13
14  End Function FDOT_MUL
```

Pseudo-code 4.4: Overloaded multiplication (binary) operator (line 10) plus recording a new tape entry

One of the attractive features of the present method is that the integration of the FDOT module into any solver requires minimal changes to the primal code and no additional code developement is needed. These steps are listed below:

1- Change all **Real** types to the user-defined type **AReal**.

2- In case there is an iterative update loop, flag all the iterative variables by simply storing their indices.

3- Call the **set_checkpoint** function before and after the main iterative loop.

4- Set the adjoint of the cost function to one, **Tape**(cost.**index**).**adjoint** = 1.0.

5- Call the **evaluate_tape** function to calculate/accumulate adjoints.

Since in most modern CFD solvers, the use of global data structure is common, changing all the **Real** types to the user-defined type **AReal** would be easy as the user needs to simply replace all instances of **Real** with **Type(AReal)** throughout the data structure. In order to flag the iterative variables, user can simply call a function that handles this operation by receiving the index of the iterative variable as an input argument. A similar function is called right after the update formula. In fact, aside from changing the **Real** types to **AReal**s, user has to only add less than 10 lines of code, i.e., two function calls for flagging iterative variables, two calls to the **set_checkpoint** function, one line setting the adjoint of the cost function to unity and one call to the **evaluate_tape** function. Also, in the adjoint version of the code, the iteration loop has to be executed for only one iteration using the fully converged solution [see Fig. (4.1)]. This is the main advantage of the proposed technique which greatly reduces the memory footprint and subsequently the computational overhead in writing, reading and evaluating the recorded tape.

The last step of the adjoint code is to call **evaluate_tape** function. This function reads in the recorded tape and starts evaluating the adjoints by using the key rule that was defined earlier in Eq. (4.11). This is mainly based on the derivatives of each operation with respect to its argument(s). Since the recorded tape is needed to be assessed in the reverse order, the process starts by evaluating the adjoints for the post-iterative portion of the tape first. Next, the proposed iterative process that was explained in Sections 4.2.1 and 4.2.2 is performed to evaluate the adjoints of the iterative portion of the tape. While consistently updating the adjoints of the iterative variables, adjoints of the post- or pre-iterative stages will be updated by getting accumulated. Note that the user can specify the number of iterations or a convergence tolerance for the iterative adjoint evaluation process. Finally, the adjoints of all intermediate as well as the input (design) variables are obtained. The results are the sensitivities of the cost (objective) function with respect to all inputs. As an example, the adjoint evaluation process is shown below for the multiplication and sine operators. Here, a select/case is used that switches over different operation tags to execute the corresponding adjoint formula [Eq. (4.11)].

```
Do I = stage_end, stage_start,-1
. . .
Select (Tape(I).optag)
   Case (MULTAG)
      IF (.NOT. Tape(Tape(I).arg1).passive) &
      Tape(Tape(I).arg1).adjoint =
      Tape(Tape(I).arg1).adjoint + &
      Tape(Tape(I).arg2).value * &
      Tape(I).adjoint
      IF (.NOT. Tape(Tape(I).arg2).passive) &
      Tape(Tape(I).arg2).adjoint = &
      Tape(Tape(I).arg2).adjoint + &
      Tape(Tape(I).arg1).value * &
      Tape(I).adjoint
   . . .
   Case (SINTAG)
      IF (.NOT. Tape(Tape(I).arg1).passive) &
      Tape(Tape(I).arg1).adjoint = &
      Tape(Tape(I).arg1).adjoint + &
      cos(Tape(Tape(I).arg1).value) * &
      Tape(I).adjoint
   . . .
End Select
```

Pseudo-code 4.5: Adjoint evaluation: select/case

One final note is that the user can simply use a set of pause/unpause functions to temporarily stop recording the tape in cases where there are check procedures or passive function calls that should not be included in the tape. The inclusion of these passive function

calls would not affect the results of the FDOT toolbox but their exclusion can decrease the length of the tape which results in higher computational efficiency.

Chapter 5

Validation and Verification

In this chapter, validation test cases are presented for verifying the numerical results obtained using the UNPAC solver. Additionally, the sensitivity analysis results using the FDOT toolbox are presented. Initially, 2D steady and unsteady flow cases are considered. For this purpose, inviscid transonic flow around the NACA0012 airfoil and turbulent transonic flow over the RAE2822 airfoil are presented. Additionally, the steady subsonic flow around the S809 airfoil is investigated to demonstrate the importance of modeling transitional effects for certain cases. Next, unsteady flows around pitching NACA0012 airfoils are studied to verify the implementation of the harmonic balance method. Following 2D validation results, the attention is shifted to 3D flow cases. First, 3D flow around an extruded airfoil is studied and the numerical results are compared to the 2D solutions. Next, the steady flow around a 3D ONERA M6 wing is simulated. Finally, the flow around a rotor in hover is studied using the rotating frame of reference capability of the UNPAC solver. Throughout this chapter, r-adaptation as well as the ROM-based convergence acceleration methods are applied to selected test cases in order to demonstrate the effectiveness and robustness of the proposed methodologies.

5.1 2D Steady: Inviscid Flow Around NACA0012 Airfoil

The first test case studied here is the steady inviscid flow over the NACA0012 airfoil. The free-stream Mach number is $M = 0.8$ with the angle of attack set to $\alpha = 1.25$ degree. These settings lead to a transonic flow with shocks forming on both the pressure and the suction sides of the airfoil. The relatively weak shock on the pressure side is located at around 35% chord length while the strong shock on the suction side is situated at about 65% of the chord length from the leading edge of the airfoil.

The "baseline" grid used for this case is made of 129×129 nodes and is generated using the Karman-Trefftz conformal transformation [216]. In the framework of our unstructured solver and in the absence of a branch-cut, the baseline grid has $128 \times 129 = 16512$ nodes, $32,896$ edges, and $16,384$ quadrilateral cells as shown in Figure 5.1.

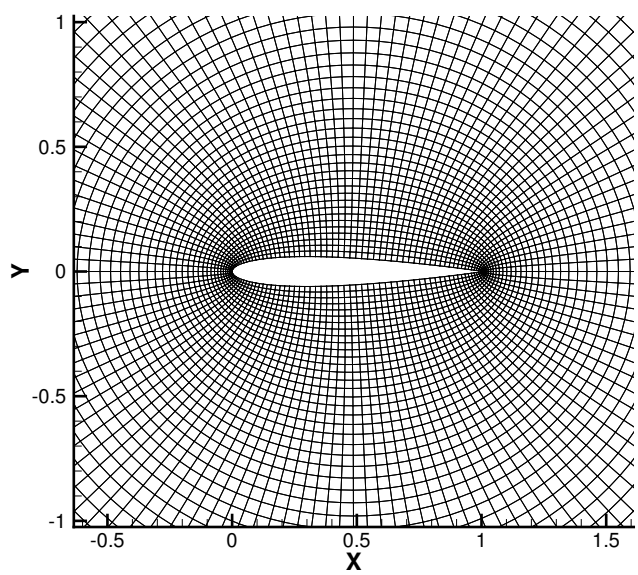
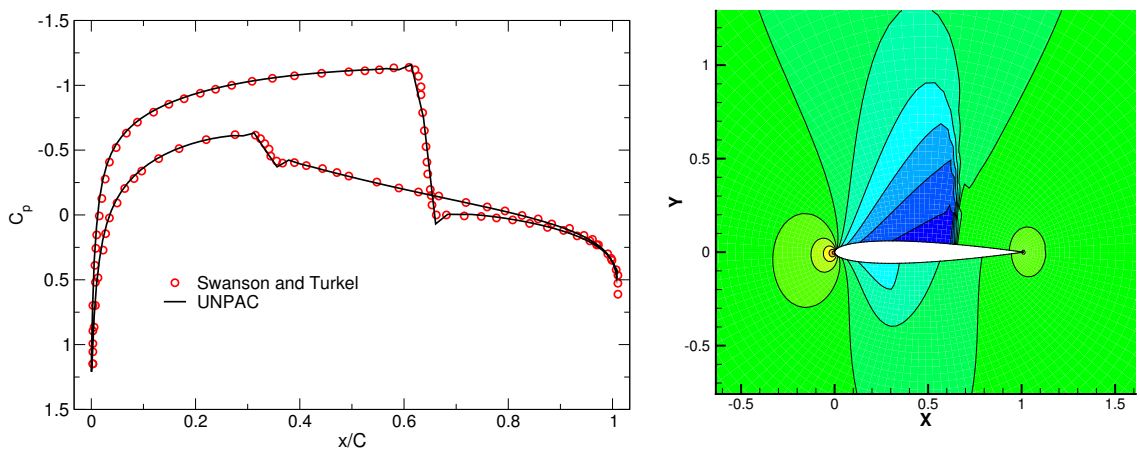


Figure 5.1: Near-field view of the baseline grid used for NACA0012 airfoil with 16512 nodes.

The second-order Roe scheme with Venkatakrishnan's limiter is used for the calculation of convective fluxes. The pressure coefficient distribution on the surface of the airfoil as well as the pressure contour field are shown in Figure 5.2. Also, the C_p results are compared

against the numerical results of Swanson and Turkel [201] obtained using a comparable flux method in a structured solver. As can be seen, shocks on both sides of the airfoil are captured with a good resolution and there is a good agreement between the results from UNPAC and those reported in the literature.

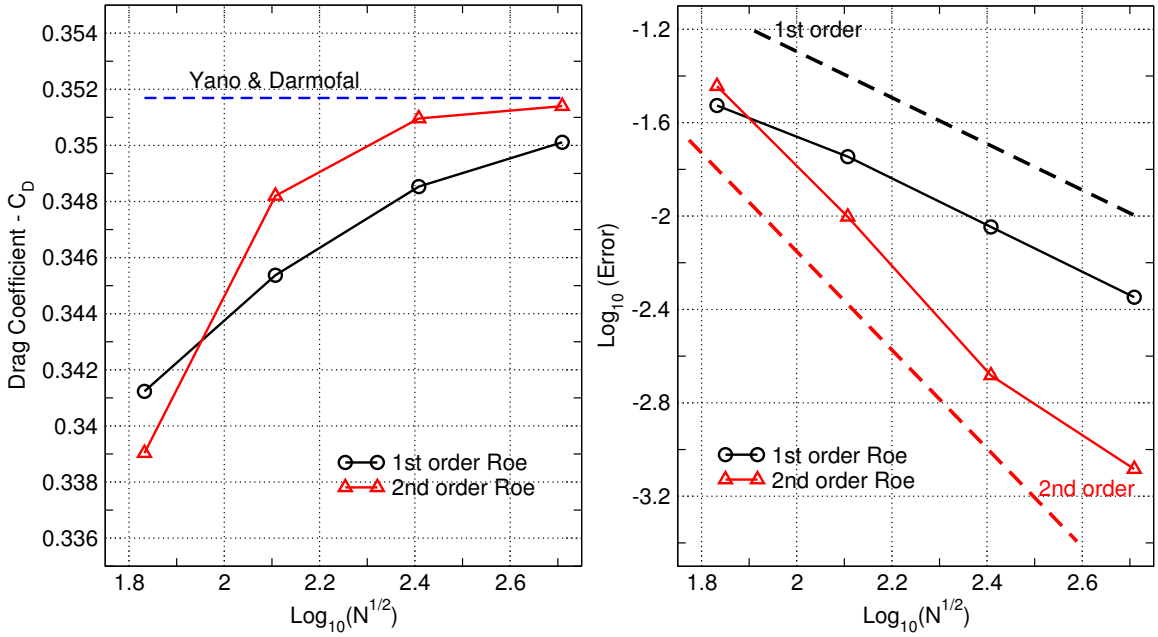


(a) C_p vs x/C **(b)** pressure field

Figure 5.2: Surface pressure distributions and the pressure contour field for the inviscid transonic flow (steady) past NACA0012 airfoil.

In order to make sure that the results are grid-independent, a grid convergence study is performed. Therefore, four different grids with successively increased resolutions, namely with 65×65, 129×129, 257×257, and 513×513 computational nodes, are considered. To quantify solution errors, the drag coefficient is chosen and the result of Yano and Darmofal [233] based on an adaptive discontinuous Galerkin-FE method is considered as the most accurate solution and the convergence results are shown in Figure 5.3.

Here, first-order and second-order Roe fluxes are considered and the rest of the solver settings are kept to be identical. For grid convergence analysis, the predicted drag coefficients using UNPAC are compared to the results of Yano and Darmofal [233] while progressively increasing the grid resolution. Additionally, in order to predict the convergence rate, the calculated errors for the first- and second-order methods are presented in a *log-log* plot during the grid refinement process.



(a) Drag vs. grid resolution **(b)** Error vs. grid resolution

Figure 5.3: Convergence rate analysis for the inviscid transonic flow past NACA0012 airfoil. Theoretical 1st and 2nd order convergence rates are shown with dashed lines.

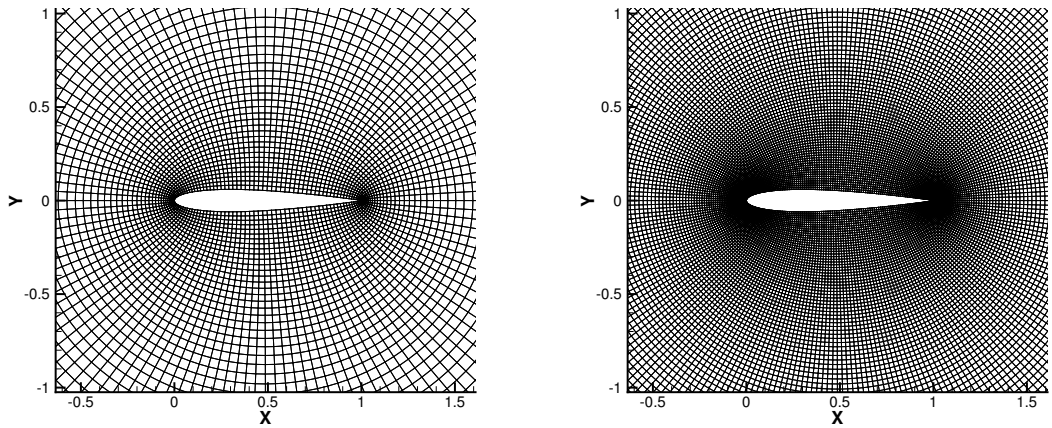
As can be seen in Figure 5.3, the first-order results exhibit a semi-linear convergence which agrees well with the truncation errors of the upwind Roe scheme. On the other hand, the convergence of the second-order method is initially quadratic while it gradually approaches to a first order convergence as the grid resolution is increased. This can be associated with the fact that the Roe scheme switches to a first-order upwind scheme at the discontinuities due to the Venkatakrishnan's limiter function [136].

5.1.1 Grid Adaptation using AMR[1]

In order to study the performance of the proposed framework for the r-adaptive AMR technique, the present case is considered. The goal here is to determine the capability of the r-adaptive approach in redistributing grid nodes by clustering them around regions of high

[1]This section, in part, is a reprint of the material as it appears in AIAA Paper 2018-3245 titled "*An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers*" (2018). Authors: **Reza Djeddi** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi** and Kivanc Ekici.

gradients. Therefore, two sets of grids with 128×129 (baseline) and 256×257 (fully-refined) nodes are used [216]. Both of these grids are shown in Figure 5.4.



(a) Baseline Grid **(b)** Fully-Refined Grid

Figure 5.4: Baseline and fully-refined grids for NACA 0012 airfoil with 128×129 and 256×257 nodes, respectively.

Our goal in using the r-adaptive AMR approach is to increase the accuracy of our CFD solver by clustering the nodes around regions with large flow gradients and curvatures while keeping the number of grid nodes exactly the same. Therefore, the AMR technique is applied to the original grid and its performance is assessed by comparing the flow field, surface pressure distributions as well as the force coefficients to those reported in the literature [233, 217]. For comparison purposes, the results from the fully-refined grid are also included.

In order to define the driving force for the r-adaptive AMR technique presented in this work, static pressure value and its gradients are used. The AMR threshold is setup such that the r-adaptive process is initiated after 5 orders of magnitude drop in the flow residual. Initially, the pressure field and its gradients are smoothed using 2 Laplacian smoother iterations with the smoothing coefficient set to 0.4. Here, the gradient and curvature force coefficients are taken to be $C_{f,\,\text{gradient}} = 2.0$ and $C_{f,\,\text{curvature}} = 6.0$. Our numerical tests have shown that usually a ratio of $C_{f,\,\text{gradient}}/C_{f,\,\text{curvature}} \approx 1/3$ leads to an efficient node clustering which has also been reported by Jones [119]. In addition, smaller force coefficients normally lead to a gradual node clustering. The AMR iterations are continued until 4

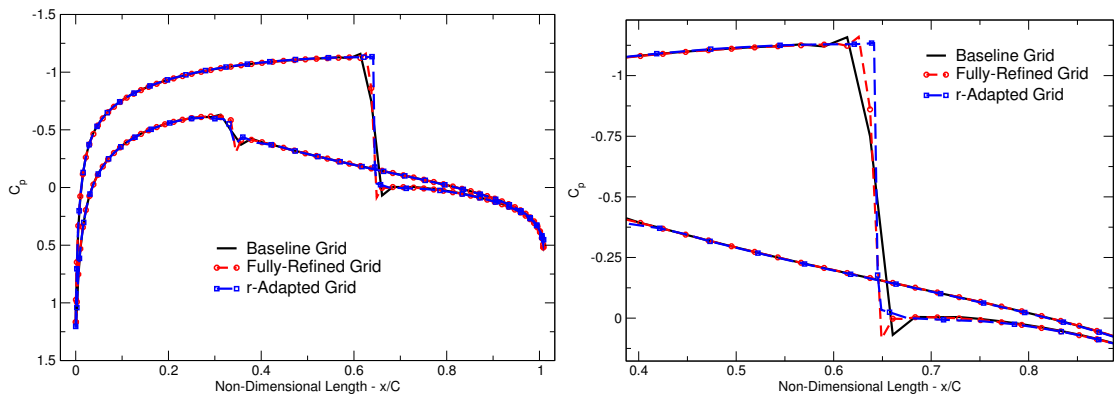
orders of magnitude drop in the nodal displacements have been achieved. Moreover, the AMR process is repeated every 500 iterations for 10 cycles. Finally, the r-adapted grid is smoothed using a Laplacian smoother with 2 passes and a smoothing coefficient of 0.5. The geometry preserving shape parameterization process described in Section 3.11 is also applied while keeping the trailing edge node fixed.

Table 5.1: Lift and drag coefficients for the inviscid transonic flow past NACA 0012 at $M = 0.8$ and $\alpha = 1.25$ deg using baseline, fully-refined, and r-adapted grids as well as those of Yano and Darmofal [233].

Grid	C_L	Error	Error (rel.)	C_D	Error	Error (rel.)
Yano & Darmofal [233]	0.35169	-	-	0.02262	-	-
Fully-Refined	0.35096	0.20%	-	0.02349	3.84%	-
Baseline	0.34820	0.99%	1.64%	0.02445	8.09%	4.08%
r-adapted (AMR)	0.35024	0.41%	0.20%	0.02302	1.76%	2.00%

The numerical results in terms of lift and drag coefficients are presented in Table 5.1 for the cases using the baseline, fully-refined, and r-adapted grids and are compared against those of Yano and Darmofal [233]. Here, the error values are once again calculated based on the numerical results of Yano and Darmofal [233]. Additionally, the errors are calculated by comparing the results of the baseline and r-adaptive grids to those obtained using the same solver but with the fully-refined grid (referred to as relative errors). As can be seen in Table 5.1, the application of the r-adaptive AMR leads to a significant increase in the accuracy of the numerical results. In fact, there is more than four-fold decrease in the numerical errors for the lift coefficient when applying the r-adaptive approach to the baseline mesh compared to the results presented in the literature [233]. This drop in the numerical error is even larger when comparing the r-adaptive results to the ones obtained from the same solver but using the fully-refined grid. Furthermore, the over-prediction of the drag coefficient is significantly reduced with the application of the AMR technique. In fact, comparing to the results of Yano and Darmofal [233], a much better agreement is obtained using the r-adapted grid than the fully-refined grid.

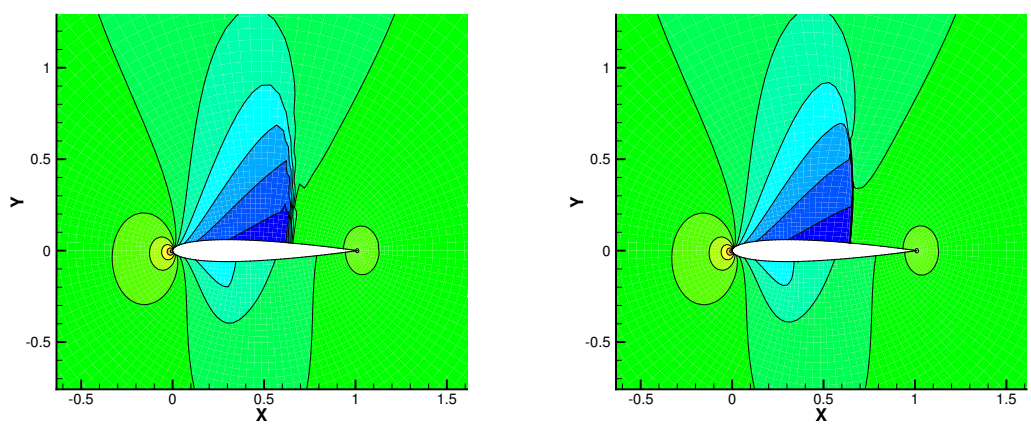
Next, the surface pressure coefficients are compared for the cases studied using the UNPAC solver with three different grid resolutions, i.e., baseline, fully-refined, and r-adapted grids. These results are shown in Figure 5.5. It can be easily noticed that the surface pressure



(a) C_p vs x/C **(b)** close-up at the strong shock (suction side)

Figure 5.5: Surface pressure distributions for the inviscid transonic flow (steady) past NACA0012 airfoil using the baseline, fully-refined, and r-adapted grids.

coefficient results for the r-adapted case exhibit a much sharper shock on the suction side compared to the results obtained using the same number of nodes but with the baseline grid resolution. In the close-up view at the location of the strong shock, it can be seen that there is a very good agreement between the results using the r-adapted and fully-refined grids while the captured shock using the AMR grid is sharper than that obtained using the fully-refined grid.

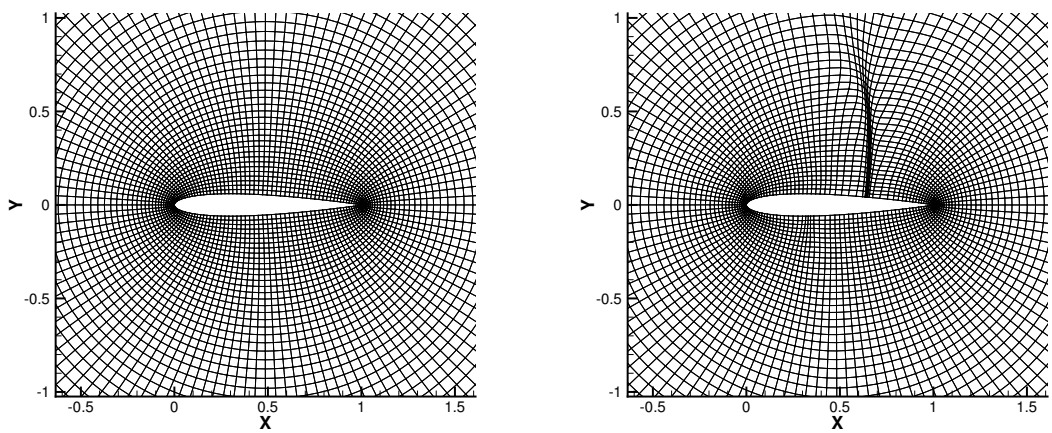


(a) Baseline Grid **(b)** r-adapted Grid

Figure 5.6: Pressure contour plots for the inviscid transonic flow (steady) past NACA 0012 airfoil using the baseline and r-adapted grids.

It is also possible to visually compare the flow solution in terms of the pressure contours as well as the numerical grid for cases with the baseline and the r-adapted grids. These results

are shown in Figures 5.6 and 5.7. As can be seen in Figure 5.7, the present r-adaptive AMR approach is capable of efficiently clustering the nodes around the strong shock on the suction side as well as the weak shock on the pressure side. Additionally, there is a significant improvement in the shock capturing using the r-adapted grid with the AMR flow results leading to a much higher shock resolution [see Figure 5.6].



(a) Baseline Grid

(b) r-adapted Grid

Figure 5.7: Baseline and r-adapted grids for the inviscid transonic flow (steady) past NACA 0012 airfoil.

So far, the presented numerical results using the r-adaptive approach indicate that the proposed AMR technique is capable of efficiently increasing the accuracy of the numerical solver by node redistribution and clustering. However, another important aspect of the r-adaptive approach is that the improved accuracy is generally achieved without increasing the size of the grid data structure. Therefore, in order to further study the performance of the AMR technique, the computational cost of the numerical solver for the different grids studied here are compared. In this regard, the CPU times for the cases with the baseline, fully-refined, and r-adapted grids are compared and presented in Table 5.2. Also, the convergence histories for the cases with the baseline, fully-refined, and the r-adapted grids are presented in Figure 5.8.

Since for this particular case 10 AMR cycles are applied every 500 iterations, there will be an increase in CPU times for the case with the r-adapted grid. Also, the AMR iterative

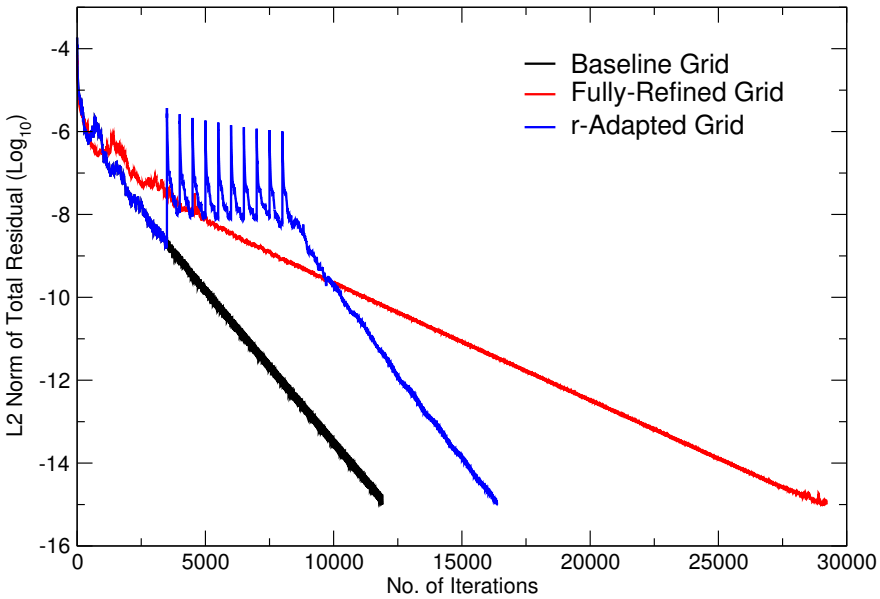


Figure 5.8: Convergence histories for the inviscid transonic flow (steady) past NACA0012 airfoil using the baseline, fully-refined, and r-adapted grids.

Table 5.2: CPU times for the three different grid resolutions (baseline, fully-refined, and r-adapted) used for the inviscid transonic (steady) flow over NACA0012 airfoil.

Grid	CPU Time (s)	Normalized CPU Time
Baseline	215.58	1.00
Fully-Refined	2037.60	9.45
***r*-adapted (AMR)**	**306.27**	**1.42**

process, post-AMR grid smoothing and grid pre-processing also contribute to a CPU time overhead. However, compared to the computational cost of the fully-refined grid case, there are significant savings in the CPU time. Note that, there is a decrease in the convergence rate for the fully-refined grid case, which is expected. Moreover, this translates into an almost quadrupled CPU time per iteration. In fact, with the application of the present r-adaptive AMR technique, a significantly higher level of accuracy has been achieved compared to the baseline grid at a much lower computational cost compared to the fully-refined grid.

5.1.2 Convergence Acceleration using ROM-based approach[2]

As discussed in Chapter 3, one the main contributions of the present work is a novel ROM-based convergence acceleration technique. In this section, the proposed technique is used to accelerate the convergence of the UNPAC solver to steady-state solution. Due to the high non-linearity of the present transonic case, a significant number of solution snapshots should be considered. As explained in Section 3.10, the effectiveness and robustness of the proposed acceleration technique are directly related to the amount of information that the collected snapshots provide. Therefore, the performance of the acceleration process can be greatly enhanced by taking the snapshots at the best possible stages during the convergence of the CFD solver. One way of achieving this goal is to follow the guidelines for snapshot selection presented in Section 3.10.3. Additionally, one of the attractions of the present technique is that increasing the number of snapshots over a fixed cycle, thus reducing the snapshot interval, can be also very helpful in achieving enhanced performance.

Here, the covariance-based acceleration technique is used over a cycle of 2,000 iterations. Also, the snapshot collection is lagged by 1,000 iterations to make sure that the system is linearly convergent. In order to study the effects of oversampling, the number of snapshots is successively increased and four cases with 10, 20, 40, and 80 equally-spaced snapshots are considered. Obviously, the snapshot intervals for these oversampling cases will be 200, 100, 50 and 25 iterations, respectively. The convergence history plots are compared against each other and are shown in Figure 5.9.

It can be seen in Figure 5.9 that the performance of the proposed acceleration technique can be greatly improved by simply increasing the number of snapshots over a fixed cycle. This way, not only more snapshots are included in the process which can increase the amount of useful information for extrapolation, but also the chances of having the best possible set of snapshots for a more effective projection will be increased. The reductions achieved in the required number of iterations to reach machine accuracy as well as those in CPU times are presented in Table 5.3. The acceleration results exhibit a steady improvement in the

[2]This section, in part, is a reprint of the material as it appears in AIAA Journal 55 (9), 3059-3071 titled "*Convergence Acceleration of Fluid Dynamics Solvers Using a Reduced–Order Model*" (2017). Authors: **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici.

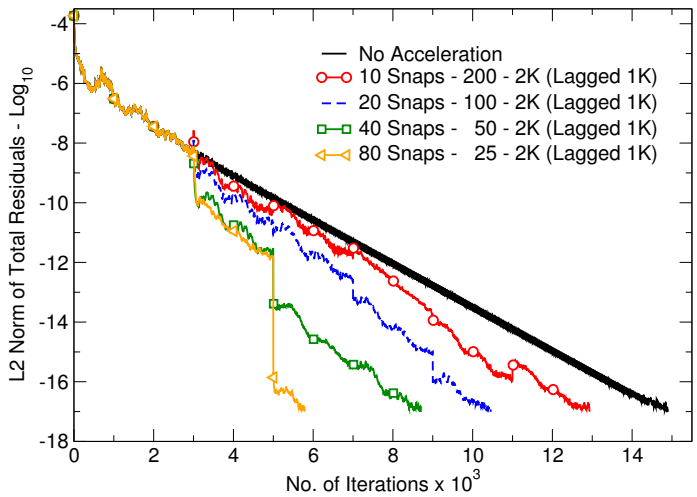


Figure 5.9: Convergence history plots for the original and accelerated solutions with oversampling approach for the inviscid transonic flow past NACA0012 airfoil.

Table 5.3: Reductions in the number of iterations and CPU times for Case 3 with oversampling of covariance-based acceleration.

Test	Iterations	Reduction	CPU time (s)	Reduction
No Acceleration	14,899	-	215.58	-
10 - 200 - 2K (L-1K)[a]	12,930	13.21%	188.09	12.75%
20 - 100 - 2K (L-1K)	10,451	29.86%	154.64	28.27%
40 - 50 - 2K (L-1K)	8,699	41.61%	131.59	38.96%
80 - 25 - 2K (L-1K)	5,788	61.15%	93.28	56.73%

[a] 10 snapshots every 200 iterations during a cycle of 2,000 itrs. and lagged by 1,000 itrs.

performance as the number of snapshots is increased. Also, the CPU time results show that the speed-ups of up to about 57% are possible when more snapshots are included in the same acceleration cycle.

One can always argue about the memory affordability especially in the case of oversampling. However, in general, the proposed technique can be performed using two different approaches: (1) Input/Output (I/O) efficient and (2) memory efficient. In the first approach, which has been used primarily in this work, the entire solution and residuals are written into external files at each snapshot. When the sanpshot collection cycle ends and the acceleration procedure is started, the recorded solutions and residuals are loaded into random-access-memory (RAM) to start the computations. This process requires the storage of $N \times (M + 2)$ values into the RAM where N is the number of degrees of freedom (DOF),

i.e., number of computational nodes times the number of equations, and M is the number of snapshots. Therefore, as an example for a case with 10 million DOF and 10 snapshots using double precision, only 915 MBytes of RAM is required which is easily affordable in a standard workstation. This memory footprint will be increased to about 1.7 GBytes for 20 snapshots, 3.2 GBytes for 40 snapshots, and 6.2 GBytes for 80 snapshots which would still be affordable. Moreover, the number of I/O operations is only $2M$ in this case which only includes one set of write-out and another set of read-in per snapshot. An alternative approach would be to load snapshots one at a time during the formation of the reduced-order model and the corresponding linear system. This approach which is called "memory efficient" only requires the storage of $5N$ values in the RAM and is independent of the number of snapshots which makes it suitable especially for cases with a large number of snapshots. However, the number of I/O operations in this case would be $M^2 + 2M$ and there would be a trade-off between the memory footprint and I/O operations. It is worth noting that using the direct access and buffered I/O capabilities of the programming language, the efficiency of I/O operations can be further enhanced.

5.2 2D Steady: Turbulent Flow Around RAE2822 Airfoil

In order further validate the UNPAC solver, turbulent and transonic flow past the RAE2822 airfoil (Case-9) is considered [43] next. This test case exhibits a strong shock wave and boundary layer interaction which makes it a challenging flow problem for any CFD solver [54]. The freestream Mach number is set to 0.734 and the angle of attack based on the wind tunnel correction is taken to be 2.79 degrees. Also, the Reynolds number based on the chord length is Re = 6.5 million. The computational grid used for this case has $258 \times 128 = 33,024$ nodes and $32,768$ quadrilateral elements and it extends to a far-field boundary approximately 150 chord lengths away from the airfoil. A close-up view of the grid is shown in Figure 5.10 and, although not presented here, grid independence studies have shown less than 2%

variation in the lift and moment coefficients when the grid resolution is doubled indicating grid convergence.

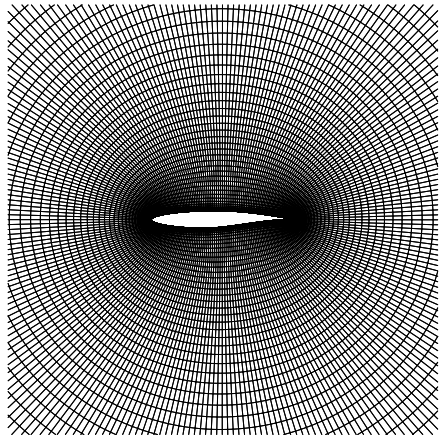


Figure 5.10: Viscous grid used for the turbulent and transonic flow over the RAE2822 airfoil.

The contour plots of the Mach number as well as the pressure are presented in Figure 5.11 for the fully converged flow field. In order to further validate the obtained numerical results, the surface pressure coefficients are compared to the experimental data [43] and the results are shown in Figure 5.12. As can be seen, the numerical results prove that there is good agreement between the present results and those reported in the literature [201].

5.2.1 Convergence Acceleration using POD[3]

As discussed earlier in detail for the previous flow problem, in the case of complex flow features with strong non-linearities and discontinuities such as those seen in transonic and turbulent flows, finding the set of the best possible snapshots for the projection can be challenging. However, as shown in Section 5.1.2, the proposed ROM-based convergence acceleration approach is found to be a simple but elegant tool to accelerate convergence to steady state solution. For this case, the nominal solver requires about 90,000 iterations to

[3]This section, in part, is a reprint of the material as it appears in AIAA Journal 55 (9), 3059-3071 titled "*Convergence Acceleration of Fluid Dynamics Solvers Using a Reduced–Order Model*" (2017). Authors: **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi**, Andrew Kaminsky, and Kivanc Ekici.

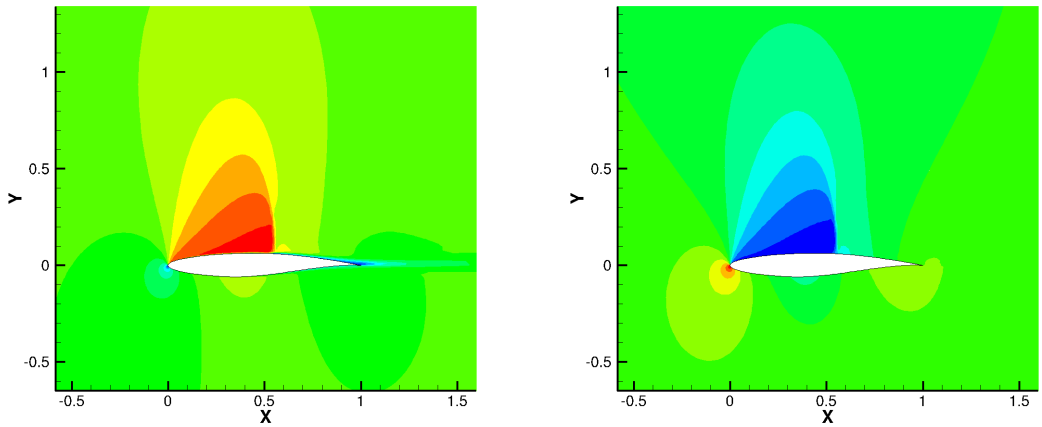


Figure 5.11: Contour plots of Mach number (left) and pressure for the turbulent and transonic flow past the RAE2822 airfoil: M = 0.734, $\alpha = 2.79°$, Re = 6.5 million.

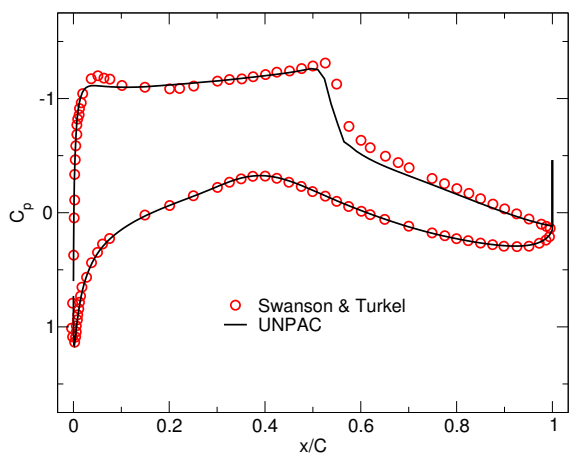


Figure 5.12: Comparison of surface pressure coefficients with the experimental results [43] for the turbulent and transonic flow past the RAE2822 airfoil: M = 0.734, $\alpha = 2.79°$, Re = 6.5 million.

converge to machine accuracy. Once again, following the rule of thumb that was discussed earlier, the acceleration process is lagged for 5000 iterations in order to have 2 orders of magnitude drop in the maximum residual in the entire flow field. To get started, 10 snapshots are taken every 1500 iterations for a cycle of 15,000 iterations. Next, the duration of the convergence cycle is fixed while increasing the number of snapshots by gradually reducing the snapshot interval. Therefore, cases with 20, 40 and 80 snapshots are considered that use

snapshots recorded every 750, 375 and 187 (next integer number) iterations, respectively. The convergence acceleration results for these cases are shown in Figure 5.13.

Table 5.4: Reductions in the number of iterations and CPU times for the RAE2822 case with oversampling of covariance-based acceleration.

Test	Iterations	Reduction	CPU time (s)	Reduction
No Acceleration	87,681	-	517.32	-
10 - 1500 - 1.5K (L-5K)[a]	67,113	23.45%	424.56	17.93%
20 - 750 - 1.5K (L-5K)	55,894	36.25%	372.42	28.01%
40 - 357 - 1.5K (L-5K)	50,135	42.82%	336.67	34.92%
80 - 187 - 1.5K (L-5K)	47,926	45.34%	334.55	35.33%

[a] 10 snapshots every 1500 iterations during a cycle of 15,000 itrs. and lagged by 5,000 itrs.

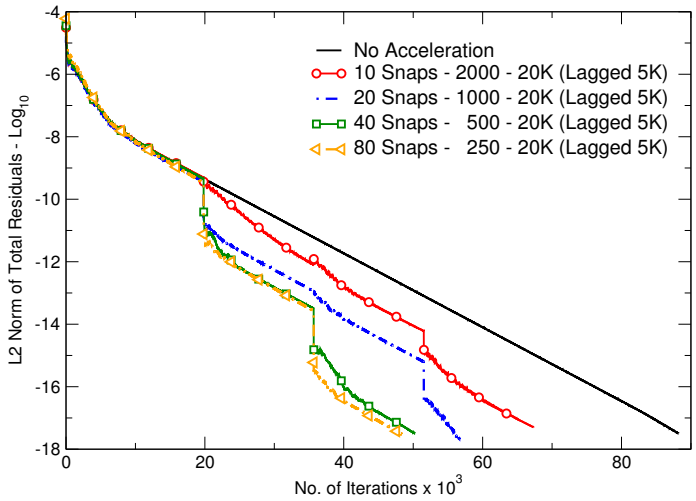


Figure 5.13: Convergence history plots for the original and accelerated solutions with oversampling approach for the turbulent and transonic flow past the RAE2822 airfoil: M = 0.734, $\alpha = 2.79°$, Re = 6.5 million.

As can be seen, the performance of the convergence acceleration technique is incrementally improved as the number of snapshots included in the projection is increased. These results once again prove the earlier assertion regarding the effectiveness of oversampling process for complex flow problems. Moreover, the reductions that are obtained in terms of the required number of iterations to reach machine accuracy as well as CPU times are presented in Table 5.4.

The presented results show a steady improvement in the performance of the convergence acceleration process as the number of snapshots is increased. Additionally, reductions of

Table 5.5: Memory footprints of the acceleration technique using I/O efficient approach for the RAE2822 case with oversampling of covariance-based acceleration.

No. of Snapshots	memory (MB)	scale-up	I/O count	scale-up
10	15.11	-	20	-
20	27.71	1.8x	40	2x
40	52.91	1.9x	80	2x
80	103.3	2.0x	160	2x

Table 5.6: Memory footprints of the acceleration technique using memory efficient approach for the RAE2822 case with oversampling of covariance-based acceleration.

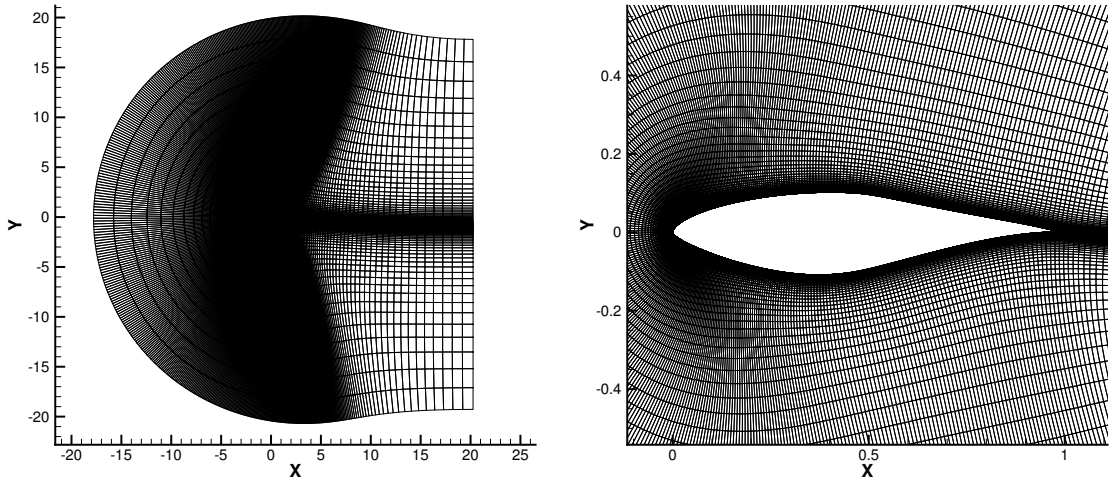
No. of Snapshots	memory (MB)	scale-up	I/O count	scale-up
10	6.30	-	120	-
20	6.30	1x	440	3.7x
40	6.30	1x	1680	3.8x
80	6.30	1x	6560	3.9x

23% to 45% in the number of iterations and 17% to 35% in the CPU time are achieved. Finally, the memory footprints (in MBytes) and I/O counts for the proposed acceleration technique are given in Tables 5.5 and 5.6 using the two approaches discussed in the previous section. The present convergence acceleration technique is relatively affordable in terms of the memory footprint while the memory efficient approach can also be used as an alternative for cases with degrees of freedom in the order of $\mathcal{O}(10^8)$ or higher with a large number of snapshots.

5.3 2D Steady: Transitional Flow Around S809 Airfoil

So far, the UNPAC solver has been validated for inviscid and viscous transonic flow cases. As for the turbulence transonic flow past the RAE2822 airfoil, a fully-turbulent flow assumption was used in the solution of the RANS-SA equations. However, as discussed previously in Chapter 2, the use of a transition model can be useful in improving the stall predictions for cases involving flow separation. This was shown by Howison [105] where a two-equation γ–$\mathrm{Re}_{\theta t}$ transition model [129, 104] was used to study the transitional flow past a S809 airfoil. Howsion and Ekici [104] have shown that the lift force in the post-stall regime can be over-predicted to a large extent in the absence of a transition model.

In order to study the effects of the transition model in enhancing the eddy viscosity predictions, a transitional flow is considered. The S809 airfoil is selected for this reason as it is one of the most well-known airfoils developed by the National Renewable Energy Laboratory (NREL) for the horizontal-axis wind turbine (HAWT) applications. This case is studied for a Mach number of $M = 0.1$ and Reynolds number of one million. Here, two cases are analyzed for angles of attack set to AoA $= 4.1°$ and AoA $= 12.2°$. The data of Ramsay et al. [171] from wind tunnel tests at the Ohio State University are used for comparison. A computational grid with $53,146$ nodes and $52,600$ quadrilateral elements is considered that has 366 nodes on the surface of the airfoil (shown in Figure 5.14). This grid has a minimum spacing of 2×10^{-6} near the wall, which corresponds to $y^+ \approx 0.1$ which is small enough to capture all turbulent flow features inside the boundary layer in the absence of a wall function. Although not shown here, our numerical results have proven to be grid-independent.

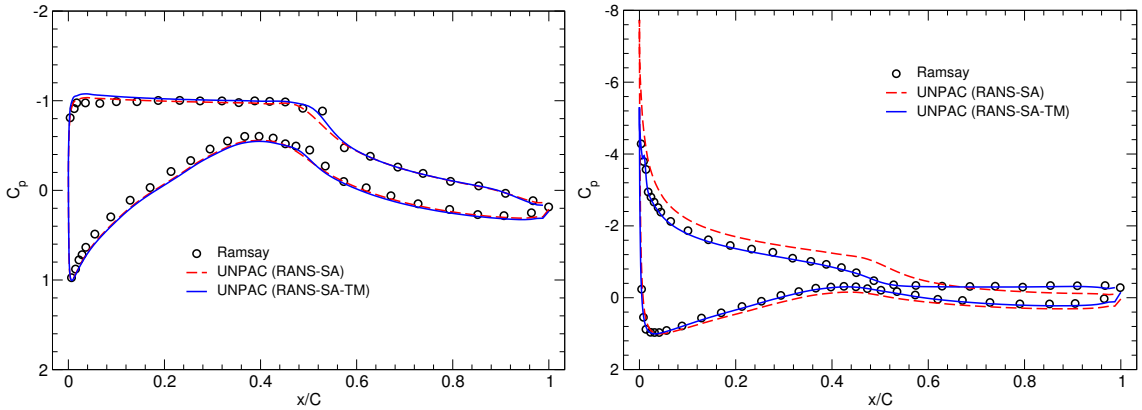


(a) far-field grid **(b)** near-field grid

Figure 5.14: The far-field and near-field views of the computational grid used for the S809 airfoil case with $53,146$ nodes.

In order to validate the obtained numerical results, a comparison of the computed surface pressure coefficients with the experimental data is presented in Figure 5.15.

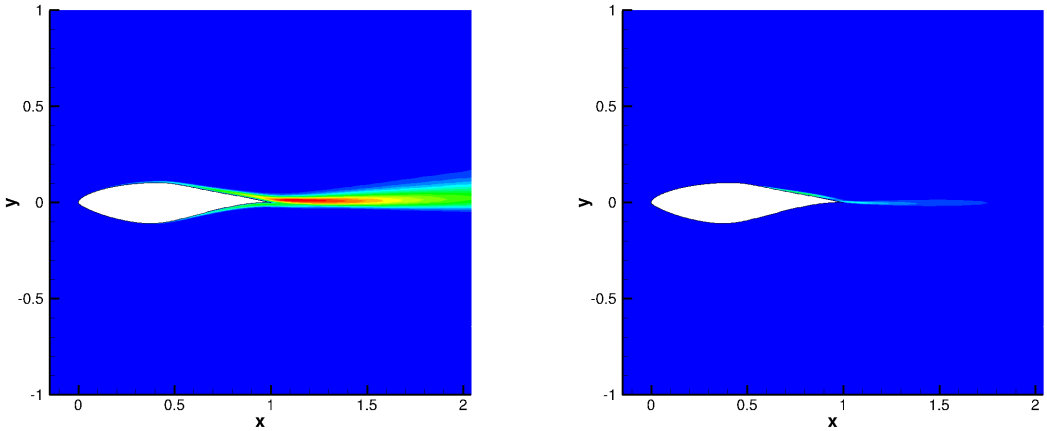
It is necessary to carefully investigate the surface pressure results presented in Figure 5.15 in order to have a good understanding and interpretation of the transition phenomenon. In the regions close to the leading edge of the airfoil, the flow remains attached, and therefore,



(a) AoA = 4.1° **(b)** AoA = 12.2°

Figure 5.15: Surface pressure coefficients for steady transitional flow past S809 airfoil with fully-turbulent assumption (RANS-SA) and enhanced with transition model (RANS-SA-TM).

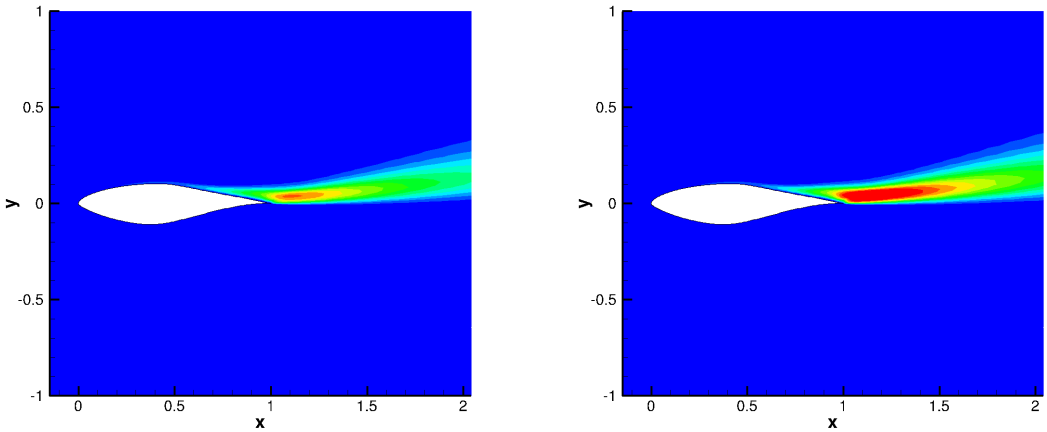
RANS-SA and RANS-SA-TM results agree well. The fully turbulent boundary layer on both sides of the airfoil remains attached up to about half of the chord after which the transition model predicts the laminar separation bubble and the following turbulent reattachment. This phenomenon can be clearly seen in Figure 5.15a on the lower surface for the case at AoA = 4.1° [105, 128]. In fact, it is downstream of this location where the transition model clearly exhibits its effects.



(a) fully-turbulent **(b)** transitional

Figure 5.16: The near-field contour plot of SA turbulent eddy viscosity, $\tilde{\nu}$ for steady transitional flow past S809 airfoil (AoA = 4.1°) with (a) fully-turbulent assumption (RANS-SA) and (b) enhanced with transition model (RANS-SA-TM).

When the flow separates, the surface pressure coefficient using the transition model gets much closer to those from the experimental data. This is even more pronounced in the case of AOA = 12.2° where there is a very good agreement between the UNPAC results using RANS-SA-TM model and the experimental data of Ramsay et al. [171]. In general, the transition model controls the amount of turbulent eddy viscosity mostly based on the local flow features including pressure acceleration, vorticity magnitude, etc.



(a) fully-turbulent **(b)** transitional

Figure 5.17: The near-field contour plot of SA turbulent eddy viscosity, $\tilde{\nu}$ for steady transitional flow past S809 airfoil (AoA = 12.2°) with (a) fully-turbulent assumption (RANS-SA) and (b) enhanced with transition model (RANS-SA-TM).

For the AoA = 4.1° flow case, due to the mostly-laminar flow features, this process leads to a reduction of the eddy viscosity compared to the fully-turbulent boundary layer assumption (see Figure 5.16). On the other hand, as the angle of attack is bumped to 12.2 degree, an opposite trend is observed where the turbulent eddy viscosity is increased due to the transition phenomenon. In fact, a fully-turbulent assumption for this case leads to an under-prediction of the flow separation and thus weaker eddy viscosity field compared to the transitional flow case (RANS-SA-TM) [104] (see Figure 5.17).

5.4 2D Unsteady: Inviscid AGARD-702-CT5 Case[4]

In order to validate the implementation of the harmonic balance (HB) method in UNPAC, Euler-HB solutions are sought for the inviscid flow past pitching NACA0012 airfoil. The fluid flow settings are according to AGARD-702 (Landon) [127]. Here, case CT5 from Landon's report is selected and the flow conditions are given in Table 5.7 where α_0 and α_p are the mean angle of attack and the pitching amplitude, respectively. Also, k is reduced frequency defined based on the half-chord length according to

$$k = \omega c / 2U_\infty$$

where ω is the fundamental frequency of excitation (pitching frequency). It must be noted that the airfoil undergoes a sinusoidal oscillation with amplitude α_p.

Table 5.7: Description of the AGARD-702-CT5 conditions for the NACA0012 airfoil [127].

Mach_∞	α_0 (deg)	α_p (deg)	k
0.755	0.016	2.51	0.0814

The AGARD-702-CT5 case involves a non-linear flow field with an oscillating shock on both sides of the airfoil. This shock moves downstream on the suction side with the increase in the angle of attack and a similar but reversed process is observed on the pressure side. The oscillating shock can travel as far as 0.45-c downstream of the leading edge before moving back toward the leading edge. This non-linear process makes CT5 case a suitable candidate for the validation and verification of the high-dimensional harmonic balance (HDHB) method implemented in UNPAC.

As discussed in Chapters 2 and 3, the harmonic balance method uses the solutions at $2N+1$ equally-spaced sub-time levels coupled together via the HB pseudo-spectral operator. Therefore, different number of harmonics, N, can be retained in the truncated Fourier series to study the time-periodic flow problem. For this case, a fully unstructured grid is used which

[4]This section, in part, is a reprint of the material as it appears in AIAA Paper 2018-3245 titled "*An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers*" (2018). Authors: **Reza Djeddi** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi** and Kivanc Ekici.

includes $5,233$ nodes where 200 of those are on the surface of the airfoil. The unstructured grid used, which has $10,216$ triangular elements, is shown in Figure 5.18. It must be noted that, the RBF method with a support radius equal to the radius of the far-field boundary is used to model the mesh motion. Therefore, in the grid shown in Figure 5.18, airfoil is at the maximum pitching amplitude where the incidence flow angle is $\alpha_\infty = \alpha_0 + \alpha_p = 2.526$ deg.

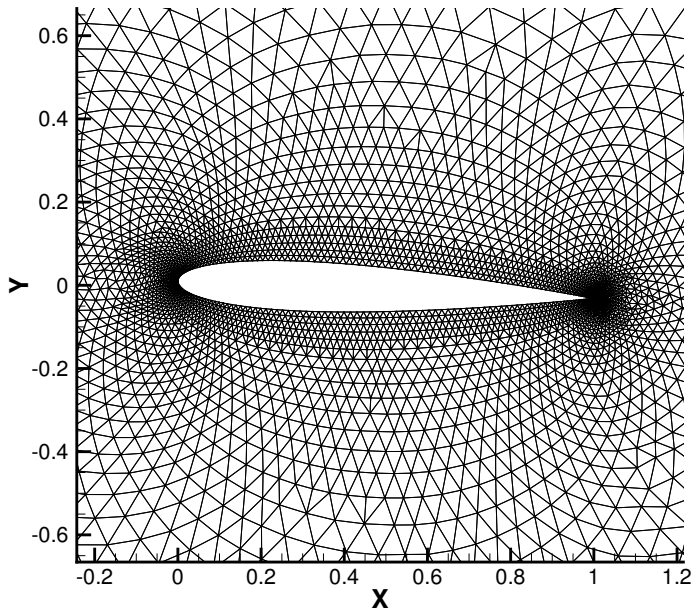
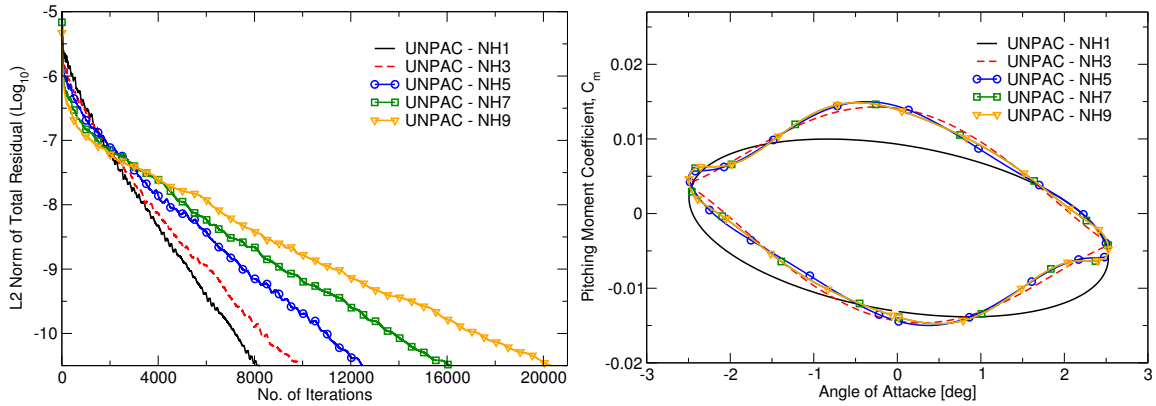


Figure 5.18: The near-field view of the unstructured grid used for the pitching NACA0012 airfoil case with $5,233$ nodes (first sub-time level, ST1).

As the first step, the HB solutions are required to be independent of the grid resolution and the number of harmonics used. While not shown here, the grid convergence studies have shown less than 2% variation in the mean lift and moment coefficients when the grid resolution is doubled. Next, a mode convergence study needs to be performed in order to make sure that the HB results are harmonic-independent meaning that increasing the number of harmonics any further would not change the HB solutions. For the CT5 case studied here, the number of harmonics retained in the HB solver is varied between $N = 1,\ 3,\ 5,\ 7$, and 9 (namingly NH1 through NH9).

In Figure 5.19, the convergence history of the HB solver and the integrated moment coefficients are presented for five different runs that have successively increasing number

of harmonics. As expected, the inclusion of more harmonics in the HB system leads to numerical stiffness which decreases the convergence rate. On the other hand, from Figure 5.19b, it is clear that more harmonics are required to capture the non-linearities in the flow field. This is mainly due to a strong shock that forms on the suction and pressure sides of the airfoil. Also, it can be seen that 5 to 7 harmonics are enough to achieve mode-converged HB solutions for this case.



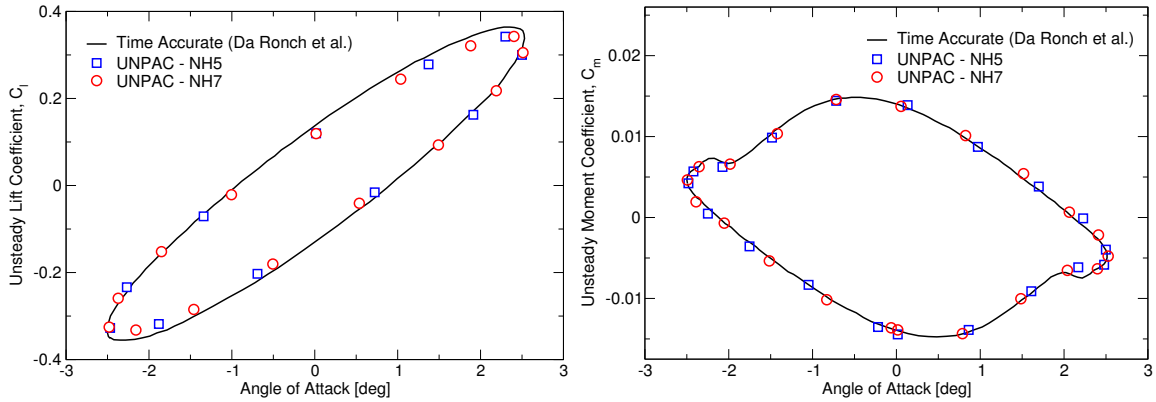
(a) convergence histories **(b)** unsteady pitching moment, C_m

Figure 5.19: Convergence history and unsteady pitching moment coefficients for different numbers of harmonics retained in the HB solver.

Following the mode convergence study presented, numerical results obtained using the HB method will be investigated next. For verification purposes, the time accurate results of Da Ronch et al. [44] are compared to the present results with 5 and 7 harmonics retained in the model. The unsteady lift and moment coefficients are presented in Figure 5.20 and HB results show a good agreement with the available time-accurate solutions [44].

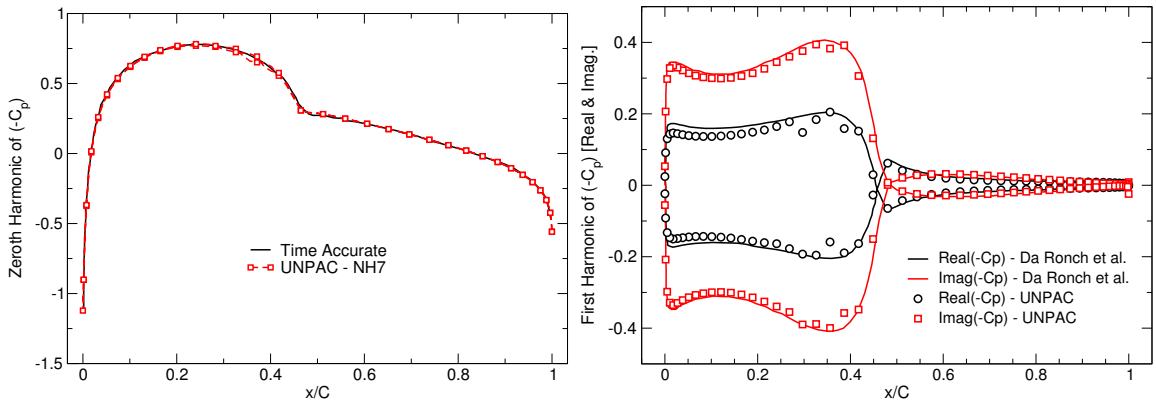
To gain further insight on the performance and accuracy of the current HB implementation, pressure coefficient C_p distributions on the surface of the airfoil are studied. In this regard, the zeroth and first harmonic of the unsteady C_p are plotted in Figure 5.21 and compared to the computations of Da Ronch et al. [44]. As can be seen, there is a very good agreement between the HB results obtained using UNPAC and those reported in the literature which verifies the HB solver implementation in this work.

Finally, the flow field solutions are examined for the instantaneous Mach number contour plots at various sub-time levels during one period. These results, as plotted in Figure 5.22, are



(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

Figure 5.20: Unsteady lift and moment coefficient results of the HB method for AGARD-702-CT5 case. UNPAC results are obtained using 5 and 7 harmonics.



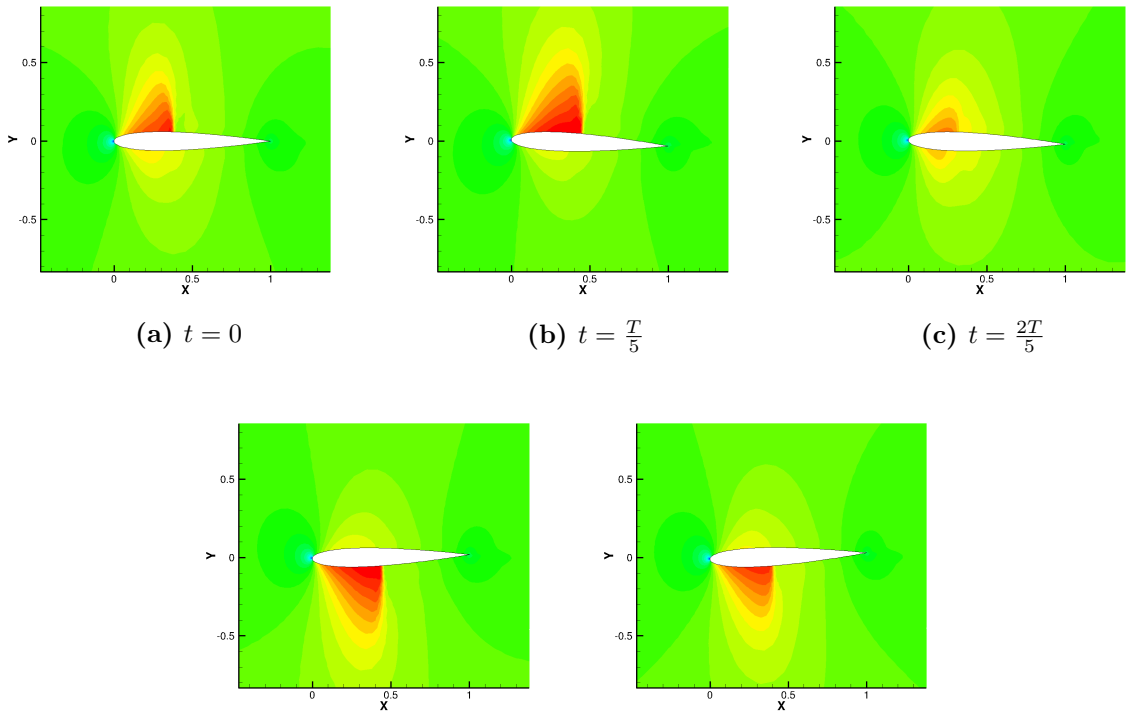
(a) zeroth harmonic **(b)** first harmonic

Figure 5.21: Zeroth and first harmonic (real and imaginary parts) of the unsteady surface pressure coefficient for AGARD-702-CT5 case. UNPAC results are obtained using 7 harmonics.

obtained using 7 harmonics retained in the HB solver and clearly depict the shock oscillation over the pitching period.

5.4.1 Grid Adaptation using AMR

As demonstrated earlier, the AGARD-702-CT5 case involves a non-linear flow field with an oscillating shock on both sides of the airfoil. The shock moves downstream on the suction side with an increase in the angle of attack, and travels as far as 45% of the chord before moving back towards the leading edge and later appearing on the pressure side of the airfoil. This



(a) $t = 0$ **(b)** $t = \frac{T}{5}$ **(c)** $t = \frac{2T}{5}$

(d) $t = \frac{3T}{5}$ **(e)** $t = \frac{4T}{5}$

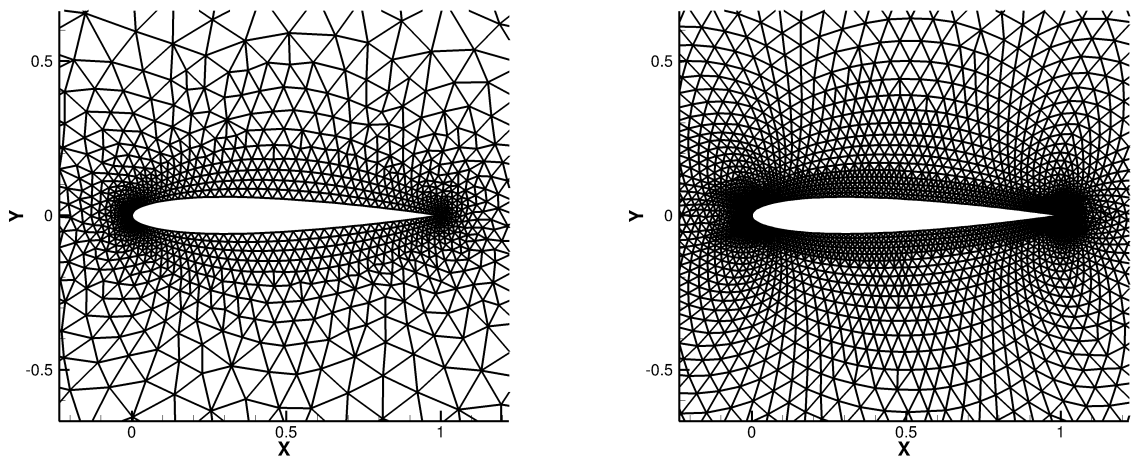
Figure 5.22: Instantaneous Mach number contour plots at 5 different sub-time levels during a single period of the unsteady flow past pitching NACA0012 airfoil (AGARD-702-CT5) case. Results are obtained using 7 harmonics.

highly non-linear phenomenon makes CT5 a suitable candidate for validation and verification of the harmonic balance-based AMR procedure.

As discussed previously, the harmonic balance method uses the solutions at $2N + 1$ equally-spaced sub-time levels coupled together via the HB pseudo-spectral operator. Therefore, different number of harmonics, N, can be retained in the truncated Fourier series to study the time-periodic flow problem. The CT5 case studied here has a relatively high free-stream Mach number which leads to the formation of strong dynamic shocks. Additionally, numerical results show that there is shocked flow around the pitching airfoil for almost 90% of the oscillation period.

The computational grid used in the previous section is considered the "fully-refined" (or fine grid) here. The strategy is to apply the AMR approach to a coarser grid with the goal of improving the numerical accuracy by reducing the discretization errors at a low

computational cost overhead. Therefore, the "baseline" (or coarse grid) used for this test case has $1,837$ nodes and $3,542$ triangular elements. The baseline (coarse) and the fully-refined (fine) grids used for the CT5 case are shown in Figure 5.23.



(a) "baseline" grid

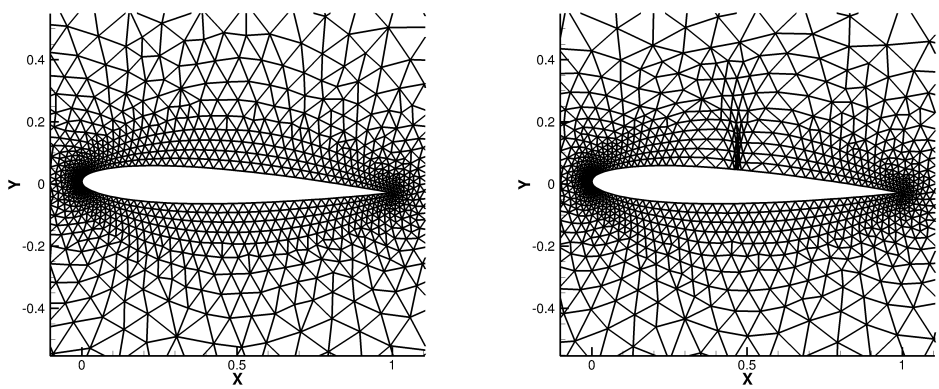
(b) "fully-refined" grid

Figure 5.23: The "baseline" and "fully-refined" unstructured grids used for the CT5 case.

In order to study the effects of the proposed AMR technique in the framework of the harmonic balance solver, the "baseline" grid is adapted to increase the solution accuracy by clustering grid nodes around regions of large flow gradients. Similar to the steady grid adaptation case that was studied earlier, the static pressure value and its gradients are used as the driving force for r-adaptation. The AMR threshold is set such that the adaptation is initiated after about 5 orders of magnitude drop in the flow residual. A 2-pass pre-smoothing is applied to the pressure field and its gradients. Unlike the steady case, the gradient and curvature force coefficients are taken to be $C_{f,\,\text{gradient}} = 10$, and $C_{f,\,\text{curvature}} = 30$ since the goal here is to have rapid clustering in a single AMR cycle. The AMR iterations are continued until 4 orders of magnitude drop in the nodal displacements have been achieved. As mentioned earlier, the desired clustering can be achieved in a single cycle when larger force coefficients are considered. Additionally, the AMR process is followed by a Laplacian grid smoothing process with a smoothing coefficient of 0.5. Finally, the geometry preserving

shape parameterization described in Section 3.11 is applied to each sub-time level grid to ensure that the surface topology is not altered.

Next, the baseline and r-adapted grids are shown in Figure 5.24 for the sixth sub-time level corresponding to $t = \frac{T}{3}$, where T is the period of excitation. Additionally, the r-adapted grids for different sub-time levels of case CT5 using seven harmonics are presented in Figure 5.25. As can be seen, the nodes are efficiently clustered around the dynamic shock as it oscillates between the suction and pressure sides of the airfoil.

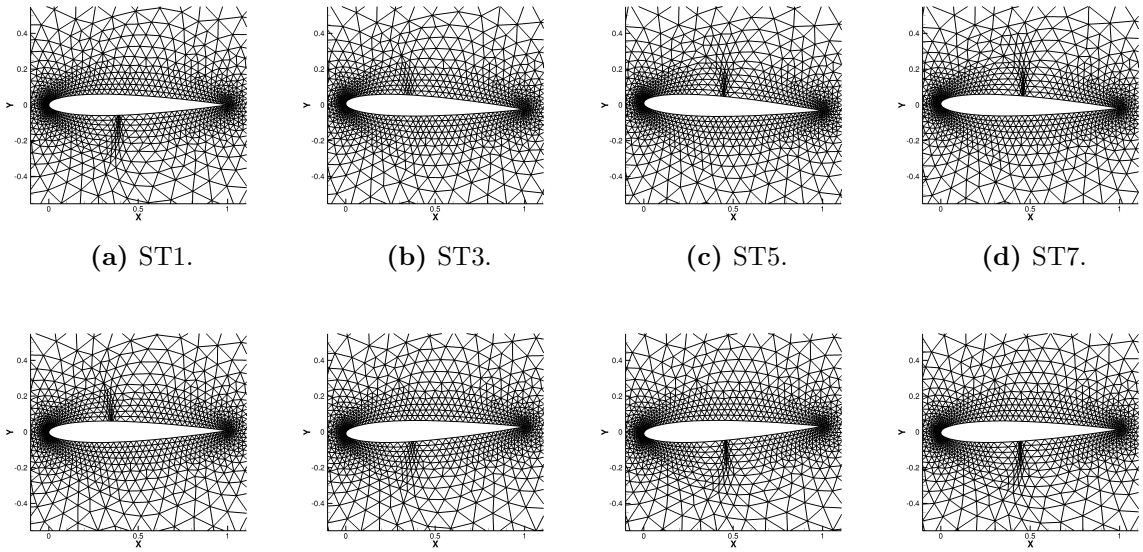


(a) Baseline Grid ($t = \frac{T}{3}$) **(b)** r-adapted Grid ($t = \frac{T}{3}$)

Figure 5.24: Close-up view of the leading edge region for the baseline and r-adapted grids used for the CT5 case at the sixth sub-time level corresponding to $t = \frac{T}{3}$.

Next, the Mach number contour plots for the baseline, r-adapted, and fully-refined grid cases are shown in Figures 5.26 and 5.27 for the sixth and twelfth sub-time levels corresponding to $t = \frac{T}{3}$ and $t = \frac{11T}{15}$. Once again, based on the flow solutions, it can be inferred that the r-adaptive technique leads to a higher resolution of the dynamic shock without increasing the computational cost of the CFD solver per iteration.

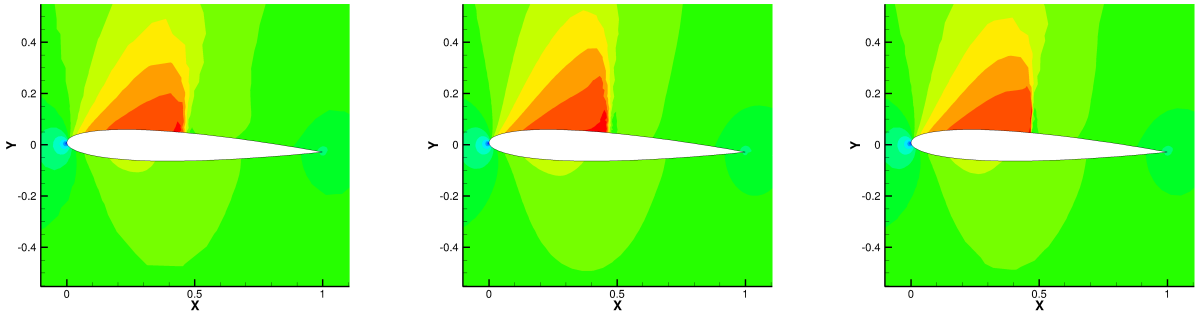
The CPU times for the cases with the baseline, fully-refined, and r-adapted grids are compared and presented in Table 5.8. Also, the convergence histories for the three different grid settings are presented in Figure 5.28. As can be seen, the AMR process leads to only about 31% increase in the computational cost compared to about ten-fold increase in CPU time for the case of the fully-refined grid.



(a) ST1. **(b)** ST3. **(c)** ST5. **(d)** ST7.

(e) ST9. **(f)** ST11. **(g)** ST13. **(h)** ST15.

Figure 5.25: r-adapted grids at different sub-time (ST) levels for the AGARD-702-CT5 case.

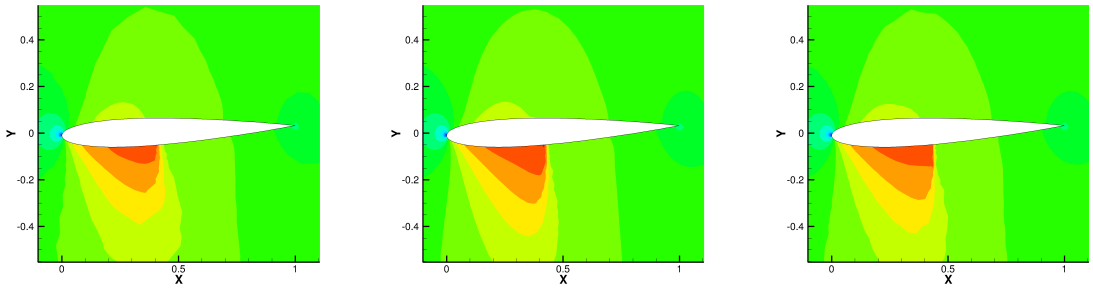


(a) Baseline Grid **(b)** Fully-refined Grid **(c)** r-adapted Grid

Figure 5.26: Mach number contours obtained using the baseline, fully-refined, and the r-adapted grids for the CT5 case at the sixth sub-time level, i.e., $t = \frac{T}{3}$.

Table 5.8: CPU times for the three different grid resolutions (baseline, fully-refined, and r-adapted) used for the AGARD-702 CT5 case.

Grid	CPU Time (s)	Normalized CPU Time
Baseline	1,749.1	1.00
Fully-Refined	12,671.2	7.24
***r*-adapted (AMR)**	**2,207.2**	**1.26**



(a) Baseline Grid **(b)** Fully-refined Grid **(c)** r-adapted Grid

Figure 5.27: Mach number contours obtained using the baseline, fully-refined, and the r-adapted grids for the CT5 case at the twelfth sub-time level, i.e., $t = \frac{11T}{15}$.

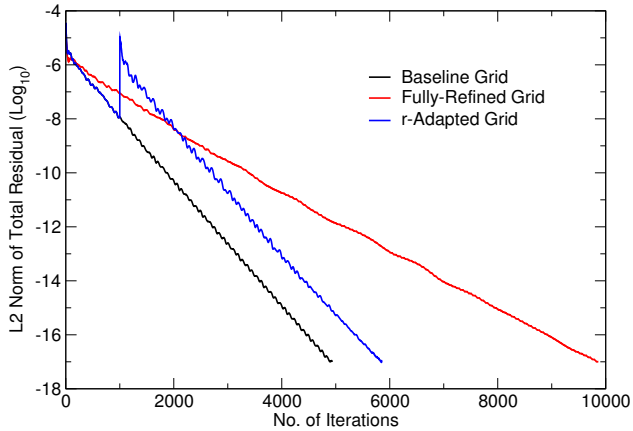
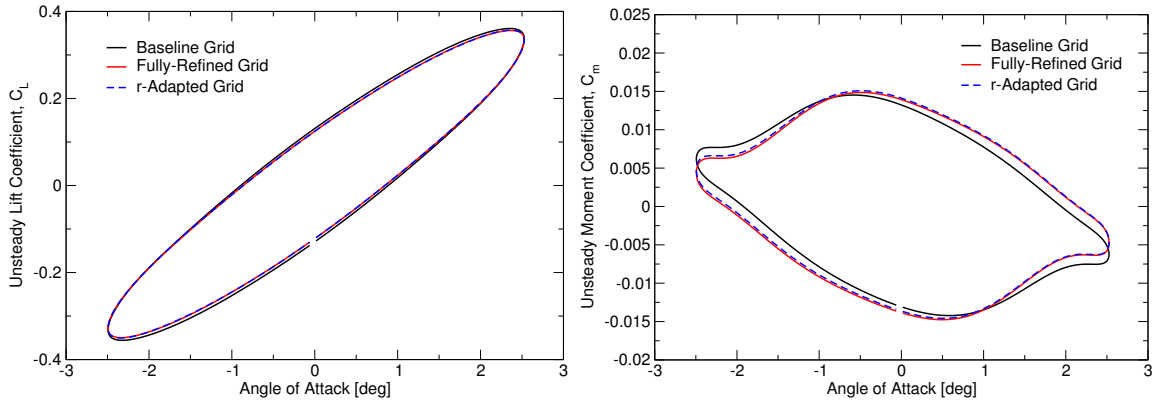


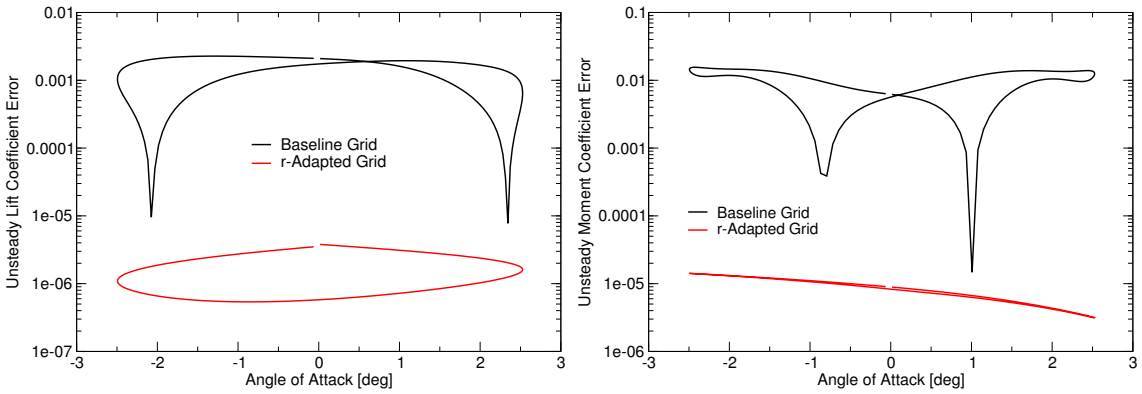
Figure 5.28: Convergence histories for the AGARD-702-CT5 case using the baseline, fully-refined, and r-adapted grids.



(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

Figure 5.29: Unsteady lift and moment coefficient results for the AGARD-702-CT5 case using the baseline, fully-refined, and r-adapted grids.

The improvements in the accuracy of the results obtained using the coarse baseline grid can be demonstrated in more detail by considering the unsteady lift and moment coefficients over the entire cycle of pitching. Here, the numerical results obtained using the baseline and the r-adapted grids are compared to those from the fully-refined grid for the CT5 case. As demonstrated in Figure 5.29, the AMR clustering can lead to significantly higher numerical accuracy without increasing the number of nodes in the computational grid. It is clear that the r-adapted grid results have a much better agreement with the results of the fully-refined grid although the number of nodes are not changed and only node clustering is performed using the AMR approach.



(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

Figure 5.30: Numerical errors for the unsteady lift and moment coefficients for the AGARD-702-CT5 case using the baseline and r-adapted grids (compared to the fully-refined grid results).

Here, the numerical results obtained using the fully-refined grid are considered as the benchmark and the errors for the baseline and r-adapted grid results are calculated for both unsteady lift and moment coefficients. These errors are presented in Figure 5.30 and clearly exhibit that the numerical errors for the r-adapted grid are orders of magnitude lower than those of the baseline grid with the same number of grid nodes. Additionally, the L2 norm of these unsteady errors are given in Table 5.9.

As shown here, the application of the AMR approach to the CT5 case has enabled us to obtain more accurate numerical results from the HB solver at the cost of a small computational overhead.

Table 5.9: L2 norm of the errors (Log_{10}) for the unsteady lift and moment coefficients obtained using the baseline and r-adapted grids compared to those of the fully-refined grid for the AGARD-702 CT5 case.

Coefficient	Baseline Grid	r-Adapted Grid
C_l	-2.878065	**-5.792785**
C_m	-1.992387	**-5.063434**

5.5 2D Unsteady: AGARD-702-CT1 Case[5]

In the previous section, the HB method was used to study the unsteady periodic flow past the pitching NACA0012 airfoil. As discussed, the problem of transonic flow over a pitching airfoil includes a strong non-linearity due to shock oscillations. According to McCroskey [149], shock oscillation is the main source of non-linearity in periodic transonic flow cases in the absence of boundary layer separation. When the two are combined, i.e., turbulent transonic flow past pitching airfoils, the non-linearities get even stronger and high fidelity methods are vital for solving the governing equations.

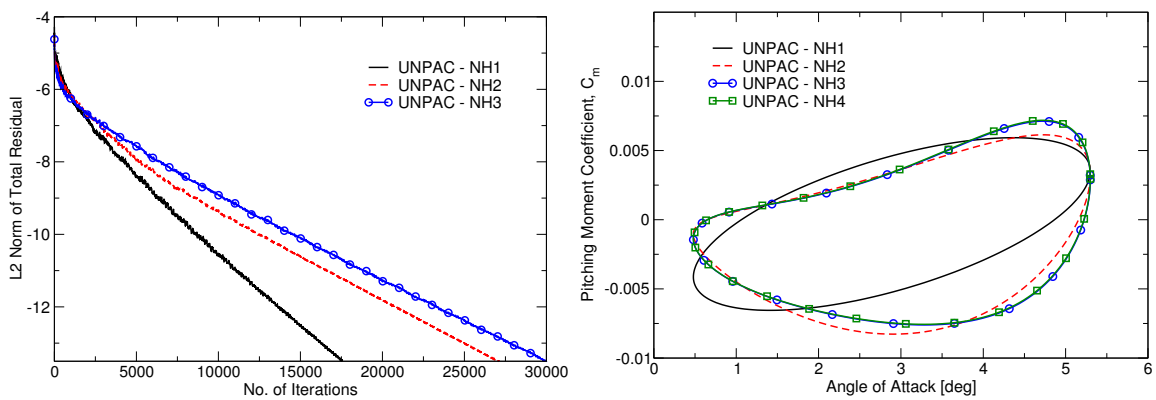
In this section, the CT1 case from AGARD-702 report is considered with the experimental results due to Landon [127]. According to the non-linear frequency domain (NLFD) results of McMullen [150] for the CT1 case, the movement of the shock in this test case is about 7.9% of the chord length. In general, the transition from linear to non-linear regime in problems involving shock oscillations happens when the movement of the shock has exceeded about 0.05-c (or 5% chord-length) according to Dowell et al. [56]. This puts the flow field of the CT1 case well into the non-linear regime. The movement of the shock varies as a function of the pitching amplitude, α_p, and the reduced frequency, k. Generally speaking, the increase in the pitching amplitude or the decrease in the reduced frequency can lead to stronger non-linearities such that linear flow assumptions can be made at very small rotation angles or very large reduced frequencies [150]. The flow settings for the case AGARD-702-CT1 are presented in Table 5.10.

[5]This section, in part, is a reprint of the material as it appears in AIAA Paper 2018-3245 titled "*An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers*" (2018). Authors: **Reza Djeddi** and Kivanc Ekici. The dissertation author was the primary investigator and author of this paper. Copyright is held by **Reza Djeddi** and Kivanc Ekici.

Table 5.10: Description of the AGARD-702-CT1 conditions for the NACA0012 airfoil [127].

Mach_∞	α_0 (deg)	α_p (deg)	k	Reynolds number
0.6	2.89	2.41	0.0808	4.8×10^6

Compared to the CT5 case studied previously, the CT1 case leads to a maximum flow incidence angle of 5.3 degree which leads to boundary layer separation downstream. Due to the shock-boundary-layer interaction that happens for the CT1 case, the HB-RANS-SA capability of the UNPAC solver has been employed and the Euler and Navier-Stokes solutions are compared. Additionally, observations are made regarding the effects of the solver fidelity on the numerical results compared to the available experimental data.



(a) convergence histories **(b)** unsteady pitching moment, C_m

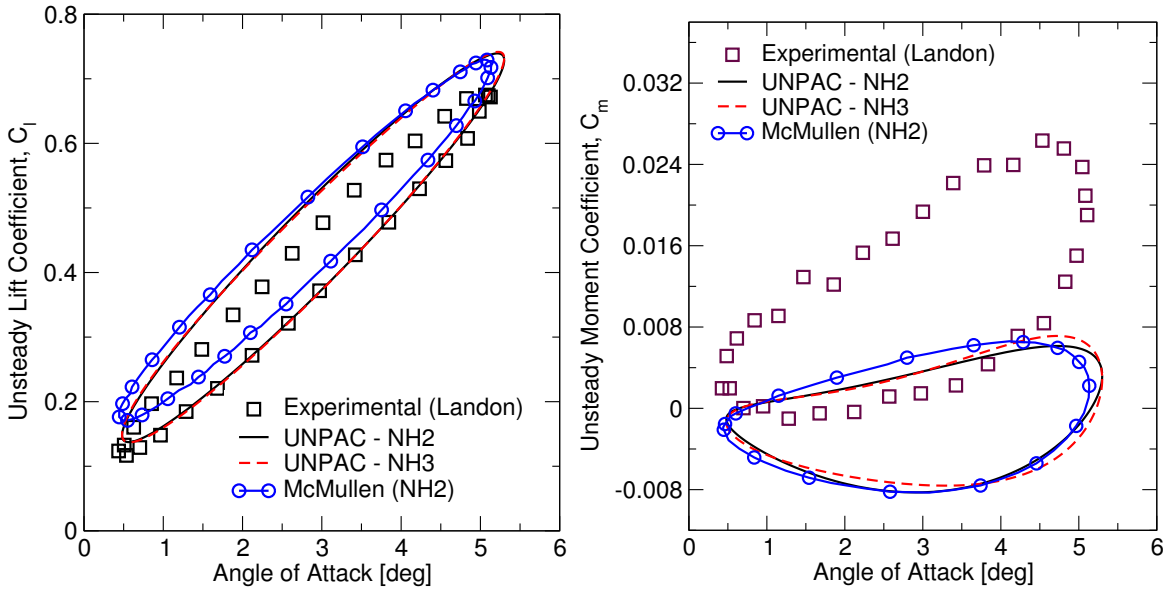
Figure 5.31: Convergence history and unsteady pitching moment coefficients for different numbers of harmonics retained in the HB solver (case AGARD-702-CT1 Euler).

5.5.1 Euler Solutions

Inviscid periodic flow past the pitching NACA0012 airfoil is considered according to the CT1 case [127]. The same unstructured grid used for the CT5 case is utilized here and a similar mode convergence analysis has been pursued. In this regard, the HB solver has been setup using 1, 2, and 3 harmonics and the pitching moment coefficient results for these three cases are compared to the NH4 results to verify mode-converged solutions. These results are presented in Figure 5.31.

As can be seen in Figure 5.31b, the unsteady pitching coefficient is no longer changing when 4 harmonics are retained in the HB solver which hints to the mode convergence of

the harmonic balance solutions at NH3. Next, the unsteady lift and moment coefficients are plotted at different pitching angles and the NH2 and NH3 results obtained using UNPAC are compared to the experimental results [127] as well as the numerical results of McMullen [150] using the NLFD method 2 harmonics (NH2).



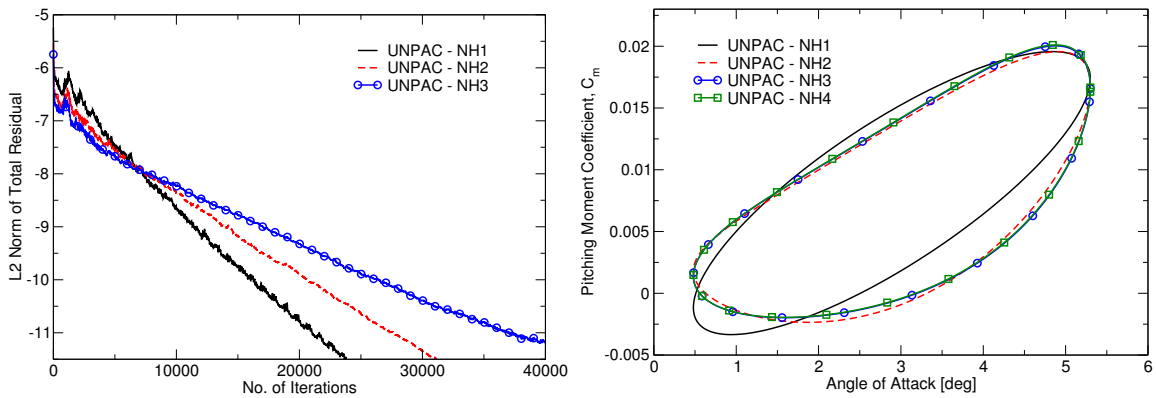
(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

Figure 5.32: Unsteady lift and moment coefficient results of the HB method for the AGARD-702-CT1 case (Euler solutions).

The Euler solutions for the unsteady lift and moments coefficient are shown in Figure 5.32. It can be seen that while the lift coefficients obtained from UNPAC using Euler solutions agree well with the available experimental data, the agreements between the moment coefficients for the present results and the experimental measurements [127] are not acceptable. This was also shown by the numerical results of McMullen [150] using the NLFD method. Based on the steady simulations of this airfoil at pre- and post-stall conditions, McMullen [150] has suggested that this agreement can be associated with the lack of viscous effects in the unsteady simulations. Therefore, focus is now shifted to obtaining HB-RANS-SA solutions for the AGARD-702-CT1 case.

5.5.2 Navier-Stokes Solutions

Steady turbulent flow tests of McMullen [150] have shown that a small separation zone appears right before the shock as the angle of attack is increased. While flow reattachment occurs aft of the shock, it separates once again at the trailing edge (TE). This separation zone at the trailing edge is relatively large and it can be seen on the suction side during half of the oscillation period [150]. Therefore, the same case (AGARD-702-CT1) is now studied using the RANS-SA solver enhanced with the HB method to handle the unsteady periodic flow.

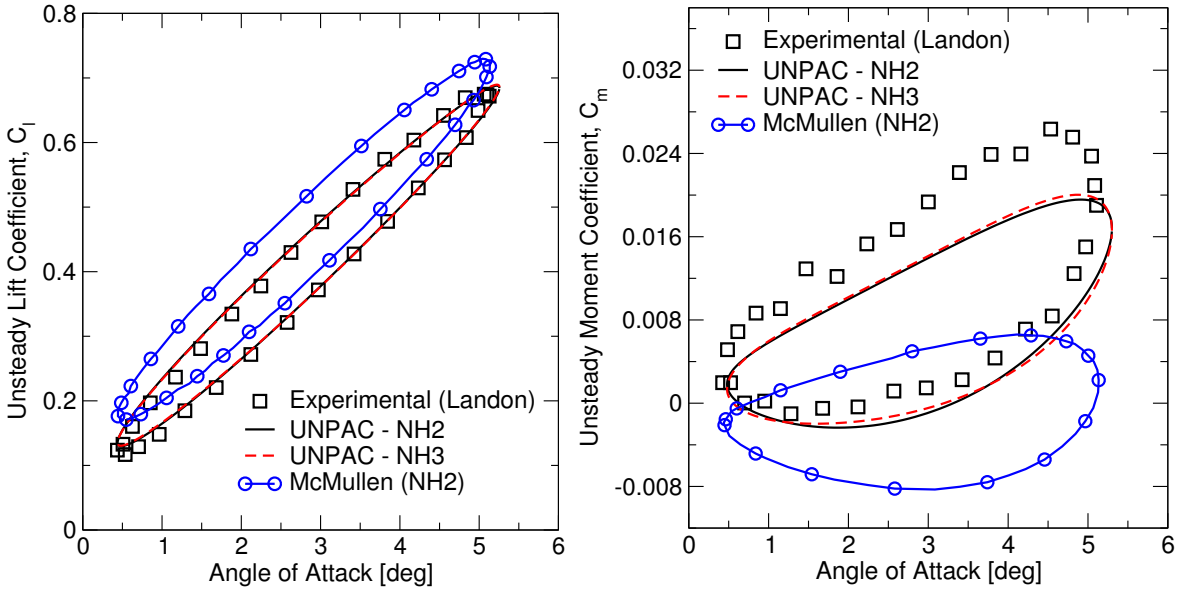


(a) convergence histories **(b)** unsteady pitching moment, C_m

Figure 5.33: Convergence history and unsteady pitching moment coefficients for different numbers of harmonics retained in the HB solver (case AGARD-702-CT1 Navier-Stokes).

The computational grid used for this work has 14,576 nodes and 14,336 quadrilateral elements with 128 nodes on the surface of the airfoil. Also, the minimum wall spacing is set such that $y^+ \approx 0.5$. Although not shown here, this grid provides results that are independent of the grid resolution. Next, the mode convergence study is performed with the results shown in Figure 5.33. As can be seen, mode-converged results are obtained with only 3 harmonics retained in the HB solver.

Next, the unsteady lift and moment coefficients are compared against the experimental results of Landon (AGARD Report) [127] as well as the numerical results of McMullen [150]. These results are presented in Figure 5.34 and suggest that while the agreements for the lift coefficient are similar to those using Euler solutions, the prediction of the pitching moment coefficients are greatly improved when viscous effects are accounted for. Also from Figure



(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

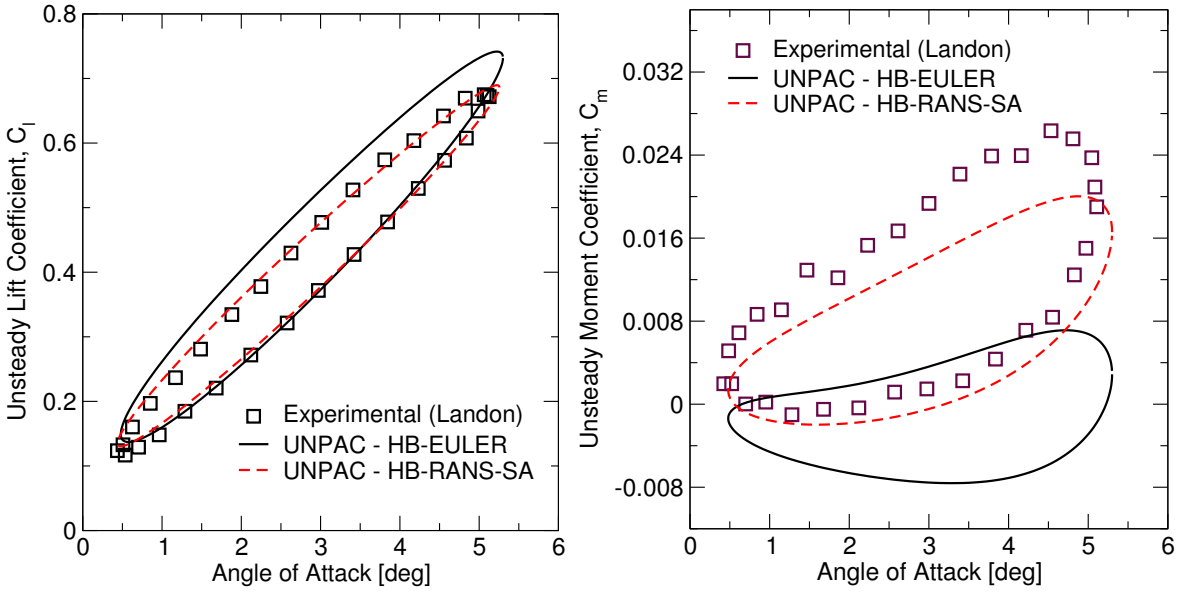
Figure 5.34: Unsteady lift and moment coefficient results of the HB method for the AGARD-702-CT1 case (Navier-Stokes solutions).

5.34b, it can be seen that there is a much better agreement between the UNPAC results and the experimental results [127] compared to the NLFD results of McMullen [150].

5.5.3 Effects of Solver Fidelity

Finally, the HB-Euler and HB-RANS-SA solutions are compared and the effects of using a high fidelity solver (HB-RANS-SA) are studied for highly non-linear unsteady transonic flow fields involving shock-boundary-layer interaction. Therefore, the unsteady lift and moment coefficient results are compared using Euler and Navier-Stokes solutions and the results are shown in Figure 5.35.

As it was shown in Section 5.4, Euler solutions for the AGARD CT5 case were sufficient to capture the flow features and a good agreement between the UNPAC results using HB model and the experimental data was achieved. However, as discussed previously, the CT1 case exhibits flow separation and reattachment as well as a shock-boundary-layer interaction which requires a higher fidelity CFD solution. As shown in Figure 5.35, the agreement between the UNPAC results and those calculated experimentally is significantly improved when HB-RANS-SA solutions are considered.



(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

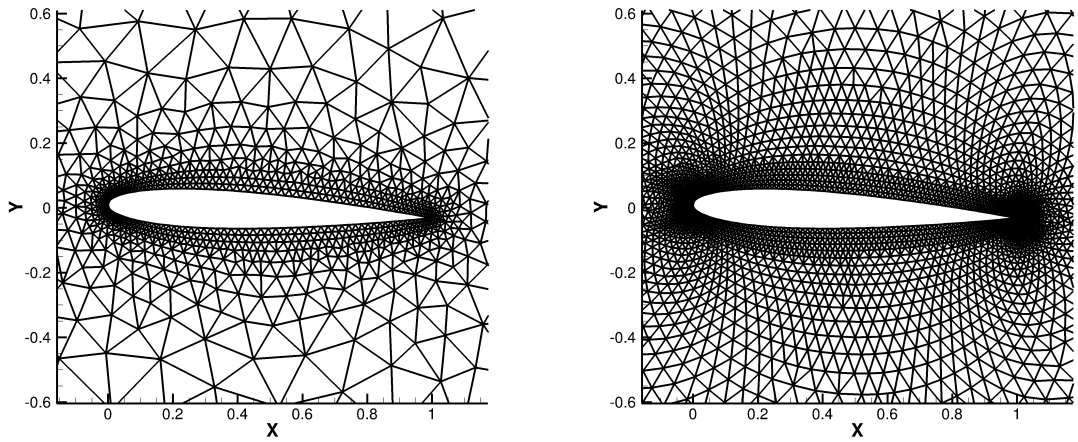
Figure 5.35: Comparison of Euler and Navier-Stokes solutions in terms of the unsteady lift and moment coefficient for the AGARD-702-CT1 case. Note that the UNPAC results are obtained using 3 harmonics and McMullen NLFD results include 2 harmonics.

McMullen [150] has also reported similar trends between the Euler and Navier-Stokes equations. However, the agreement of the HB-RANS-SA results obtained using UNPAC solver with the experimental data of Landon [127] are much better compared to McMullen's results and the improvements achieved by switching to HB-RANS-SA solutions are much more pronounced that those reported by McMullen [150]. This can be associated with the fact that the zero-equation (algebraic) turbulence model of Baldwin-Lomax [9] has been used by McMullen [150] who also states that the predictions of the eddy viscosity using the Baldwin-Lomax turbulence model are not reliable in cases involving flow separation and reattachment.

5.5.4 Grid Adaptation using AMR

Finally, the performance of the r-adaptive AMR technique in the ALE framework is assessed. For this reason, the Euler solutions for the CT1 case are reconsidered and the "fully-refined" unstructured grid with $10,216$ triangles ($5,233$ nodes and $15,449$ edges) is used here. Due to the cosine motion of the airfoil, a dynamic shock is formed at the first sub-time level that

corresponds to a 5.3 degree angle of attack with the airfoil pitching downward. Previously, it was shown that retaining 3 harmonics in the HB solver leads to mode-converged results and therefore, the exact same flow and solver settings are considered herein.



(a) "baseline" grid

(b) "fully-refined" grid

Figure 5.36: The "baseline" and "fully-refined" unstructured grids used for the CT1 case.

Additionally, a coarsened version of the "fully-refined" grid with $2,450$ triangles ($1,308$ nodes) is generated where the edge lengths are doubled. Therefore, the number of nodes in the "baseline" grid is almost quadrupled to obtain the "fully-refined" grid. The unstructured fully-refined and the baseline grids at the first sub-time level with $\alpha = 5.3$ deg are shown in Figure 5.36. It must be noted that for all unsteady grid results presented here, the mean angle of attack is prescribed at the free-stream flow while the airfoil is allowed to pitch according to the unsteady angle of attack. Therefore, in the case of the first sub-time level grids presented in Figure 5.36, the airfoil is only rotated for 2.41 degrees while the incidence flow angle is kept at 2.89 degrees. Once again, the coarse baseline grid is used with the goal of increasing numerical accuracy by clustering nodes around important regions using the AMR approach.

Similar to AGARD-702-CT5 case that was presented earlier, the coarse baseline grid is used with the goal of increasing numerical accuracy by clustering nodes around important regions using the AMR approach. Here, the exact same settings of the AMR process used

earlier for the CT5 case are considered and the grid adaptation is initiated after about five orders of magnitude drop in the flow residual.

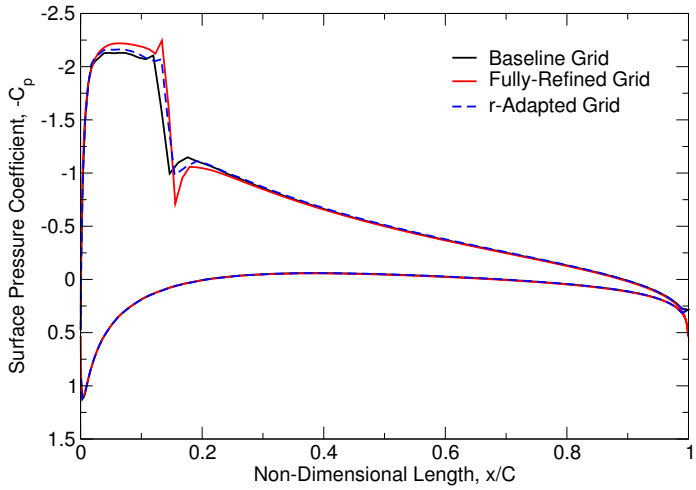
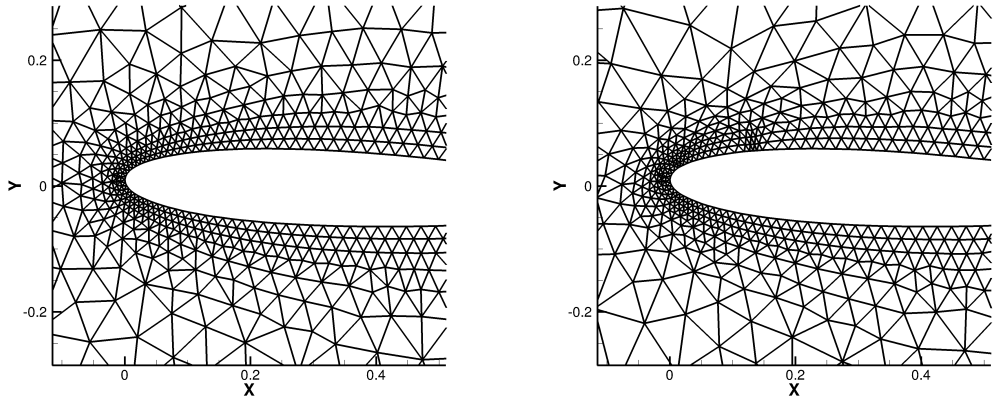


Figure 5.37: Surface pressure distributions ($-C_p$) for the CT1 case (NH3) using the baseline, fully-refined, and r-adapted grids at the first sub-time level, i.e., $\alpha = 5.3° \downarrow$.

According to the NH3 results obtained using the fully-refined grid, the first sub-time level solution includes a dynamic shock on the suction side at around 13% chord length from the leading edge. Thus, after applying the r-adaptive technique, the surface pressure coefficient distributions are presented for the first sub-time level using the baseline, r-adapted, and the fully-refined grids as shown in Figure 5.37. As can be seen, the application of the r-adaptive AMR leads to a sharper shock, which is comparable to that obtained using the fully-refined grid.

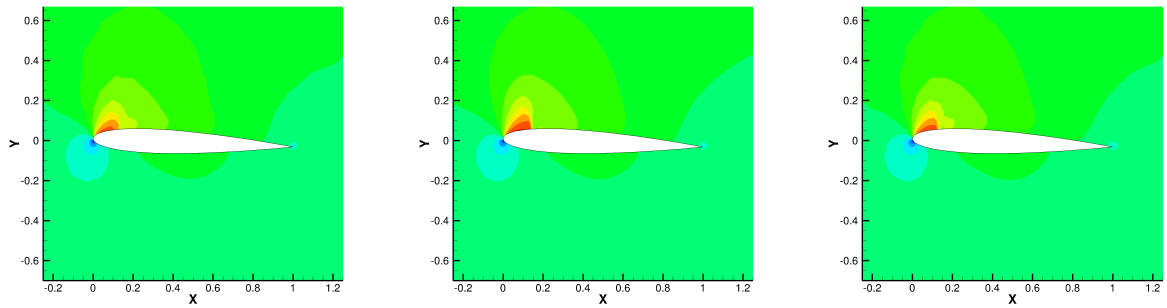
Next, the close-up view of the leading edge region for the baseline grid as well as the r-adapted grid are provided in Figure 5.38 for the first sub-time level of the HB solver. As can be seen, the nodes are efficiently clustered around the shock on the suction side of the airfoil. Also, the Mach number contour plots for the baseline, r-adapted, and fully-refined grid cases are shown in Figure 5.39 for the same sub-time level.

The visual (qualitative) comparison of the flow solutions shown in Figure 5.39 proves that the r-adaptive technique leads to a higher resolution of the dynamic shock. This is achieved without increasing the number of grid nodes which results in identical CPU times per iteration. Obviously, there is a CPU time overhead for the application of the AMR technique and the necessary processes that follow in addition to a slow down due to a jump



(a) Baseline Grid **(b)** r-adapted Grid

Figure 5.38: Close-up view of the leading edge region for the baseline and r-adapted grids used for the CT1 case at the first sub-time level ($\alpha = 5.3° \downarrow$).



(a) Baseline Grid **(b)** Fully-refined Grid **(c)** r-adapted Grid

Figure 5.39: Mach number contours obtained using the baseline, fully-refined, and the r-adapted grids for the CT1 case at the first sub-time level ($\alpha = 5.3° \downarrow$).

in the residual right after AMR is applied. However, as shown previously for the steady case, this overhead is relatively small compared to the significant accuracy improvement gained from the r-adaptive AMR approach. The CPU times for the cases with the baseline, fully-refined, and r-adapted grids are compared and presented in Table 5.11. Also, the convergence histories for the cases with the baseline, fully-refined, and the r-adapted grids are presented in Figure 5.40. As can be seen, the AMR process is initiated after about five to six orders of magnitude drop in the flow residual and a single cycle of AMR is applied to achieve the desired node clustering.

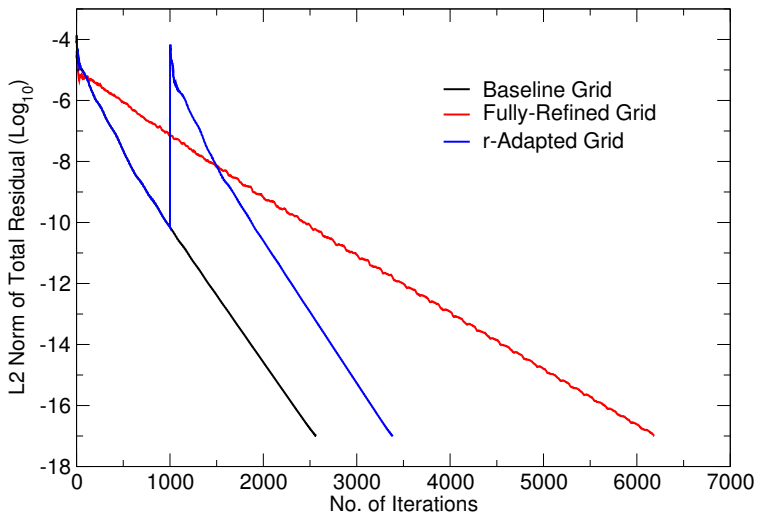
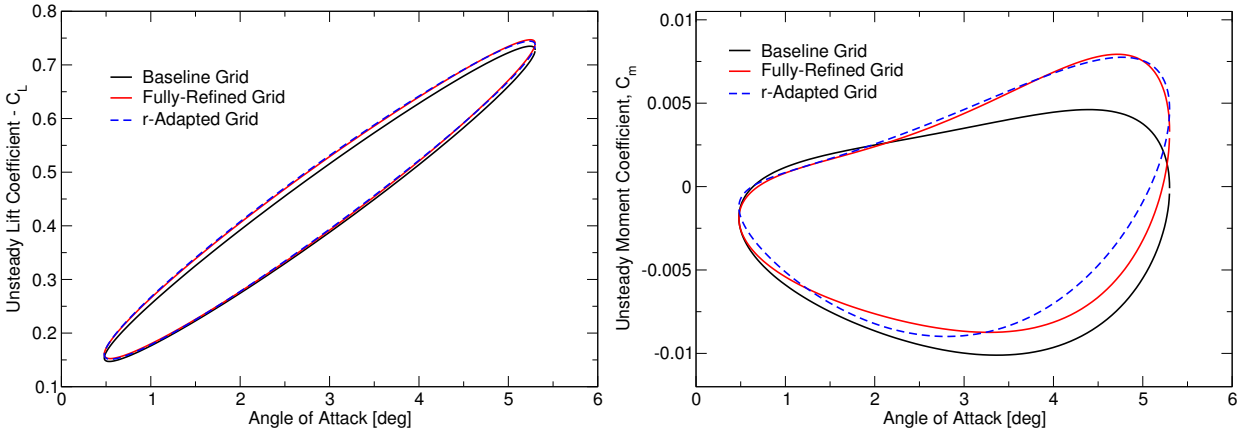


Figure 5.40: Convergence histories for the AGARD-702 CT1 case using the baseline, fully-refined, and r-adapted grids.

Table 5.11: CPU times for the three different grid resolutions (baseline, fully-refined, and r-adapted) used for the AGARD-702 CT1 case.

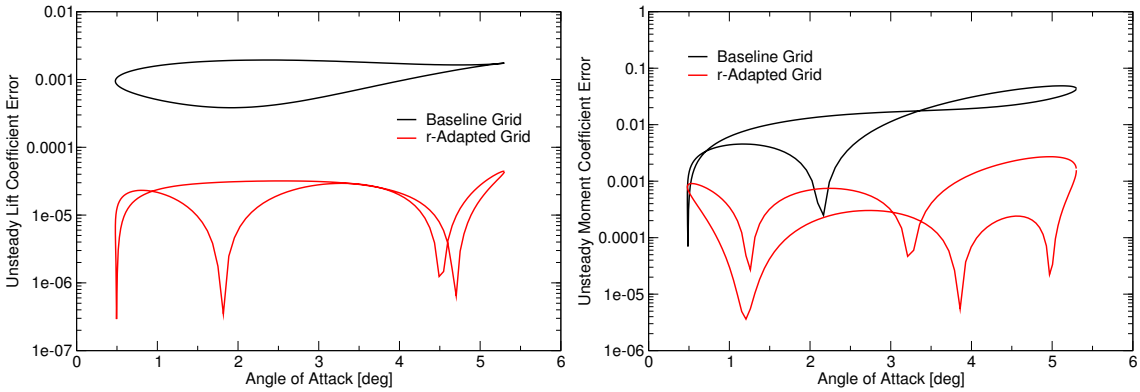
Grid	CPU Time (s)	Normalized CPU Time
Baseline	801.9	1.00
Fully-Refined (Fine)	7,447.5	9.28
***r*-adapted (AMR)**	**1,109.9**	**1.38**



(a) lift coefficient, C_l **(b)** pitching moment coefficient, C_m

Figure 5.41: Unsteady lift and moment coefficient results for the AGARD-702-CT1 case using the baseline, fully-refined, and r-adapted grids.

The improvements in the accuracy can be shown in more detail by considering the unsteady lift and moment coefficients. Here, the numerical results obtained using the baseline and the r-adapted grids are compared to those from the fully-refined grid in order to show that the node clustering achieved using AMR technique can lead to significantly higher numerical accuracy without increasing the number of nodes in the computational grid. These comparisons are provided in Figure 5.41. As can be seen, the r-adapted grid results have a much better agreement with the results of the fully-refined grid although the number of nodes are not changed and only node clustering is achieved using the AMR approach.



(a) unsteady errors of lift coefficient, C_l

(b) unsteady errors of pitching moment coefficient, C_m

Figure 5.42: Numerical errors for the unsteady lift and moment coefficients for the AGARD-702-CT1 case using the baseline and r-adapted grids (compared to the fully-refined grid results).

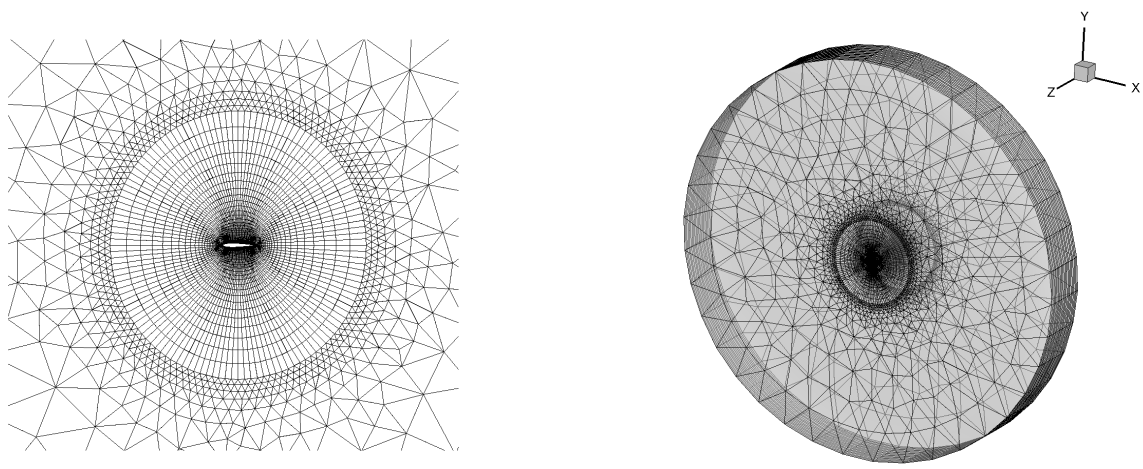
Next, the numerical results obtained using the fully-refined grid are considered as the benchmark and the baseline and r-adapted grid results are used to calculate the numerical errors for each case which are provided in Table 5.12. These unsteady lift and moment coefficient errors are provided in Figure 5.42. Clearly, the numerical errors for the r-adapted grid are orders of magnitude lower than those of the baseline grid with the same number of grid nodes. Also, for both unsteady lift and the unsteady moment coefficients, the numerical errors are significantly lower when grid adaptation is used.

Table 5.12: L2 norm of the errors (Log_{10}) for the unsteady lift and moment coefficients obtained using the baseline and r-adapted grids compared to those of the fully-refined grid for the AGARD-702 CT1 case.

Coefficient	Baseline Grid	r-Adapted Grid
C_l	-2.894424	**-4.667339**
C_m	-1.734874	**-3.138296**

5.6 3D Steady: Flow Around Extruded NACA0012 Airfoil

As the first three-dimensional test case studied in this work, the flow around an extruded NACA0012 airfoil is considered. In fact, the flow around extruded 2D airfoils can be used as an elegant tool to verify the implementation of the 3D solver as well as a mean for validating the obtained numerical results.



(a) near-field grid (2D) **(b)** far-field grid (3D)

Figure 5.43: Near-field view of the hybrid 2D and far-field view of the hybrid 3D grids used for the NACA0012 (2D and extruded 3D) airfoil case.

For this reason, the same case studied initially in Section 5.1 is reconsidered. However, this time a new and hybrid grid is utilized that can serve as an example of the mixed-grid capabilities and grid-transparency of the UNPAC solver. The initial 2D grid around the NACA0012 airfoil consists of a structured domain with 128×35 nodes and $4,480$ quadrilateral elements that is extended about 5 chord lengths in normal direction. Following this inner

structured block, the computational domain is extended further for about 20 chord lengths in an unstructured region with 930 nodes and $1,694$ triangular elements. Finally, the 2D grid is extruded in z-direction for 5 chord lengths with 20 extrusion levels (21 nodes in z-direction). This extrusion process transforms the quadrilateral and triangular elements of the 2D grid into hexahedral and prism (wedge) elements in 3D, respectively. Overall, the 3D grid for this case has $113,610$ nodes and a total of $123,480$ cells ($89,600$ hexahedral and $33,880$ prism elements). The near-field view of the 2D grid as well as the far-field view of the 3D grid are shown in Figure 5.43.

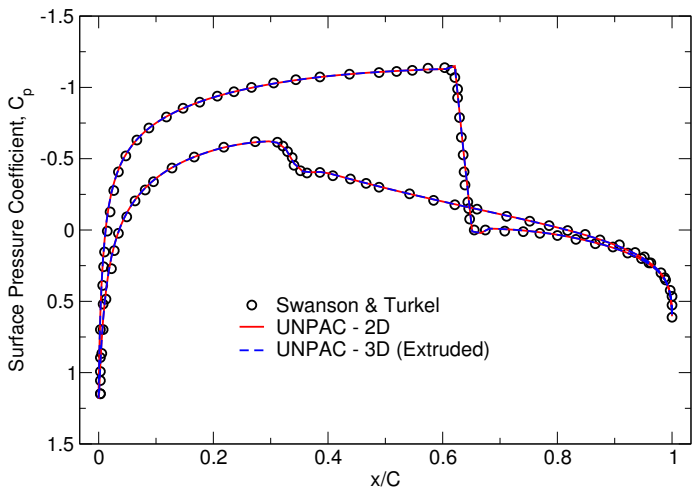
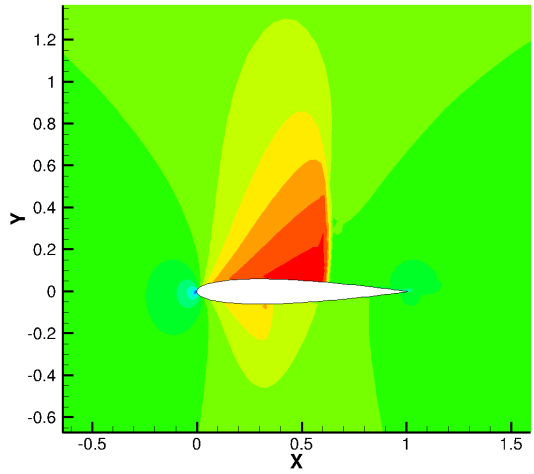


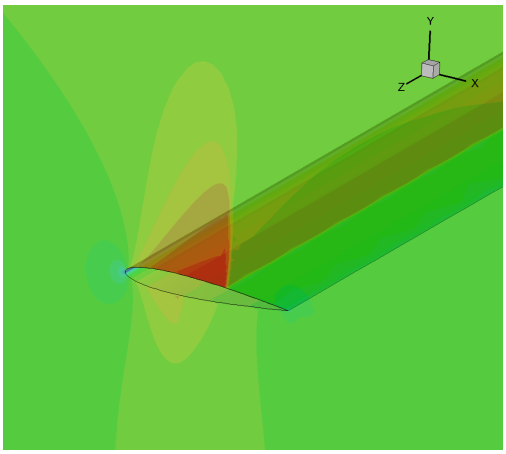
Figure 5.44: Surface pressure coefficient distributions for the inviscid transonic flow past NACA0012 airfoil (2D and extruded 3D).

Here, the inviscid transonic flow over the NACA0012 airfoil and the NACA0012 extruded wing are considered (see section 5.1) and the computed C_p distributions are compared to each other for the 2D and 3D cases. This comparison is shown in Figure 5.44 which proves that the numerical results for the 2D airfoil and the extruded wing (3D) are identical. It must be noted that for this case, symmetry boundary conditions are imposed at the ends of the wing to ensure that the 3D simulations "mimic" the 2D case.

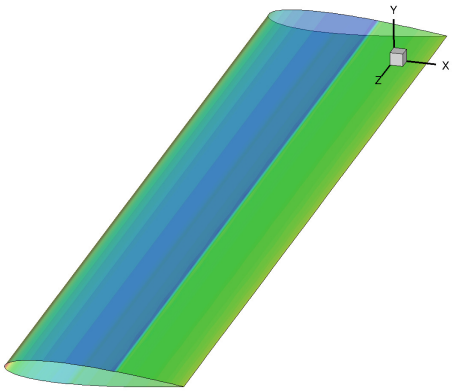
Finally, the Mach number contours around the 2D airfoil, iso-value contour planes of Mach number around the 3D wing, as well as the pressure contours on the surface of the wing are shown in Figure 5.45. These results clearly exhibit the formation of the strong and weak shock on the suction and pressure sides of the NACA0012 extruded wing, respectively.



(a) Mach number contours (2D)



(b) Mach number contour field (3D)



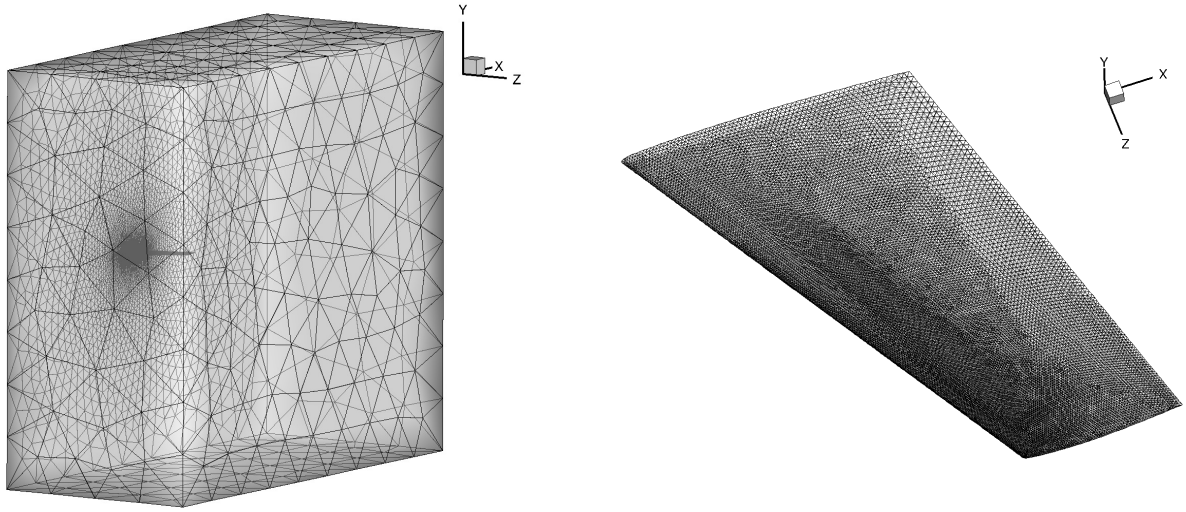
(c) Pressure contour field (3D)

Figure 5.45: Mach number contours for the inviscid transonic flow past the 2D NACA0012 airfoil and the 3D extruded NACA0012 wing.

5.7 3D Steady: Flow Around ONERA M6 Wing

The flow past the ONERA M6 wing is the first 3D test case investigated for validation and verification of the UNPAC solver. This configuration has been studied extensively and often used as a classical benchmark test case to validate three-dimensional CFD solvers. The aerodynamics of this wing involve a region of supersonic flow as well as a special shock formation known as the lambda shock [61, 105]. The geometry of the transonic M6 wing is based on the symmetric airfoil sections of type ONERA D which have a maximum of 10%

thickness-to-chord ratio. The M6 wing has a sweep angle of 30 degrees at the leading edge and an aspect ratio of 3.8 and is also tapered with a ratio of 0.562. The flow conditions are set according to the experiments carried out by Schmitt and Charpin [180] with a free-stream Mach number of 0.8395 and an angle of attack of 3.06°. Also, the Reynolds number based on the mean aerodynamic chord length for this case is 11.72 million.



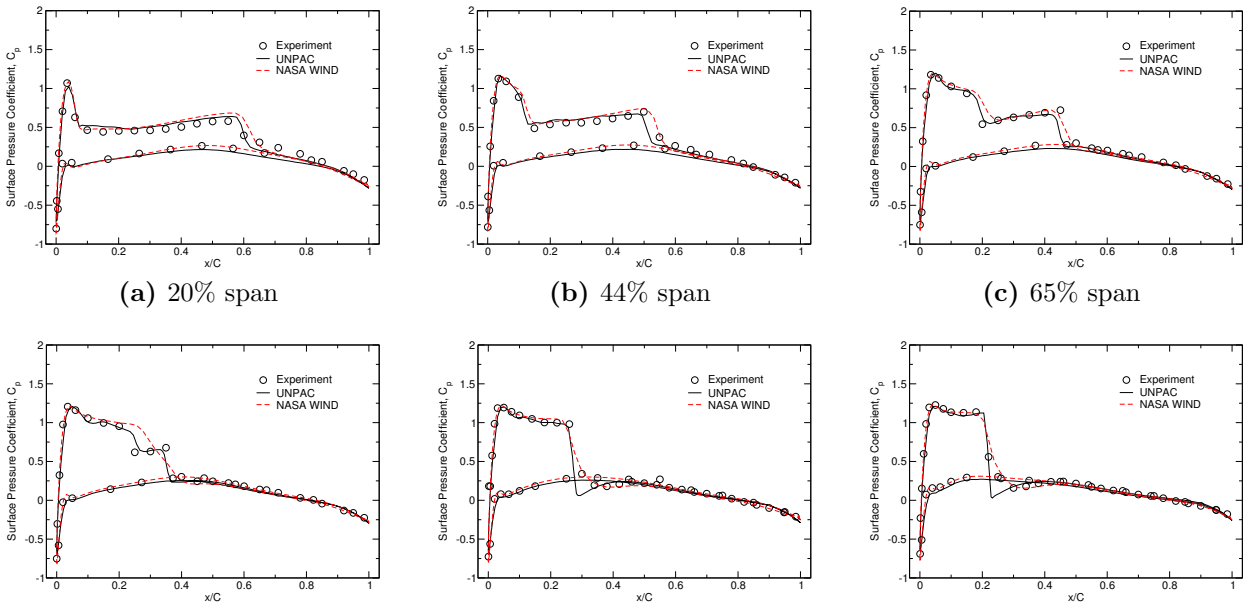
(a) Computational domain with $341,797$ tetrahedra **(b)** Surface mesh with $52,856$ triangles

Figure 5.46: Volume and surface meshes used for the transonic flow past ONERA M6 wing.

Here, a rectangular block computational domain is used which extends about 15 chord lengths on each side in the cross-sectional planes and for about 5 chord lengths in the span-wise direction. The far-field and near-field views of the grid used for the ONERA M6 wing case is shown in Figure 5.46. This grid is made of $72,791$ nodes and $341,797$ tetrahedral elements. Also, $52,856$ triangular faces are defined on the surface of the wing with $y^+ \approx 1.0$. A symmetry boundary condition is used on the root-plane and far-field boundary conditions are used for the rest of the outer boundaries. Here, the viscous (no-slip) wall boundary condition is imposed on the surface of the wing. Also, while not shown here, the obtained numerical results are grid-converged.

Numerical results obtained using the UNPAC solver are compared against the experimental data of Schmitt and Charpin [180] as well as the RANS solutions obtained using the NASA WIND solver [188]. These solutions are reported at 6 different sections along the

span of the wing and the distribution of the surface pressure coefficient at each section are shown in Figure 5.47 and compared to the benchmark data.



(a) 20% span **(b)** 44% span **(c)** 65% span

(d) 80% span **(e)** 90% span **(f)** 95% span

Figure 5.47: Surface pressure coefficients for turbulent transonic flow past ONERA M6 wing compared to experimental data [180] and results of NASA WIND [188].

As can be seen in Figure 5.47, there is a very good agreement between the UNPAC solver results and the experimental data. It must be noted that in many CFD solutions [206, 61, 105] reported in the literature including those with NASA WIND solver [188], the double shock at the 80% span is not captured accurately or there are some disagreements with the experimental results. However, the numerical results using the UNPAC solver have proven to be highly accurate with reasonable agreements between UNPAC, experimental, and NASA WIND results.

Finally, the pressure contours on the surface of the wing as well as on the symmetry boundary are presented in Figure 5.48. The pressure distribution on the suction side of the wing clearly depicts the formation of the lambda shock as well as the supersonic flow regime. Also, at the root of the wing, a strong and high resolution shock is captured which agrees well with the CFD results available in the literature [61, 105].

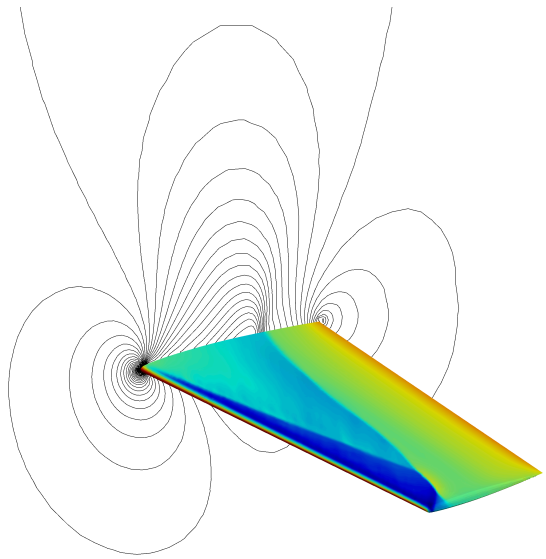


Figure 5.48: Pressure contours on the wing surface and the symmetry boundary for the turbulent transonic flow past ONERA M6 wing.

5.8 3D Steady: Caradonna-Tung Rotor

As the last test case and in order to validate the relative frame of reference feature of the UNPAC solver, the flow around a helicopter rotor in hover is considered. Based on the experiments carried out by Caradonna and Tung [38], the rotor geometry consists of two untapered and untwisted blades with NACA0012 profile. Here, two lifting cases are considered with a collective pitch angle of $\theta_c = 8$ degrees and a pre-cone angle of $\beta = 0.5$ degrees as shown in Figure 5.49. Also, the rotor blades have an aspect ratio of 6 and a unit span.

The axis of rotation is aligned with the x-axis with the rotor blades spanning in z-direction. Two rotational speeds of $\Omega = 1250$ rpm and $\Omega = 2500$ rpm are considered which correspond to tip Mach numbers of $M_{\text{tip}} = 0.439$ and $M_{\text{tip}} = 0.877$, respectively. The rotor is in hover mode which means that the free-stream Mach number in the inertial frame of reference would be zero.

A cylindrical computational domain is chosen for this case which spans four units in each direction. A hybrid mesh with $4,692$ quadrilateral cells on the top and bottom surfaces and 408 triangular cells on the root and tip surfaces of each blade is used. The hybrid volume mesh consists of $9,384$ pyramid and $458,375$ tetrahedral cells for a total of $467,759$ grid cells

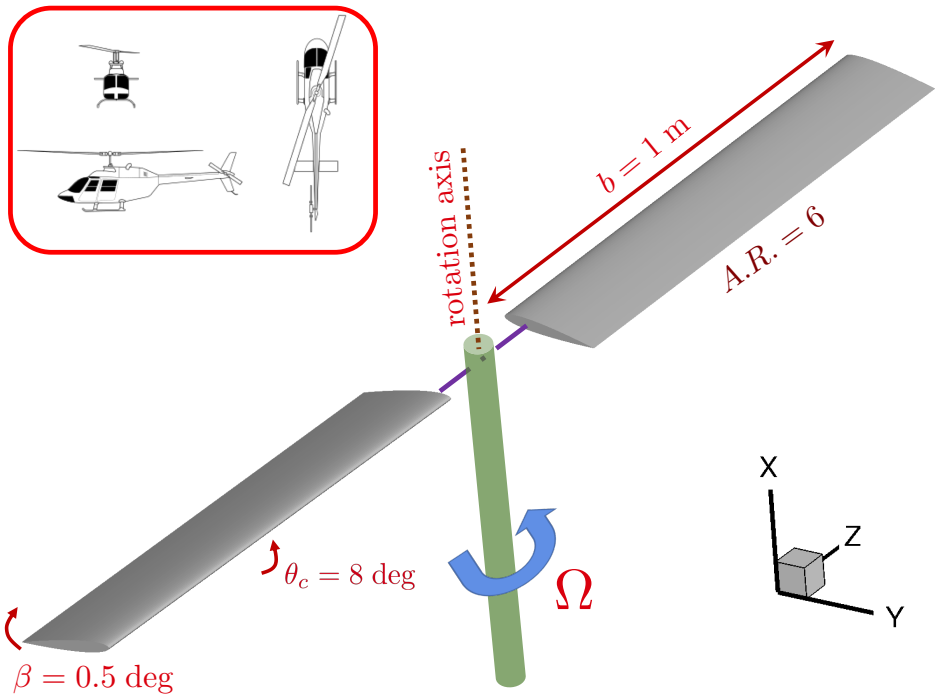
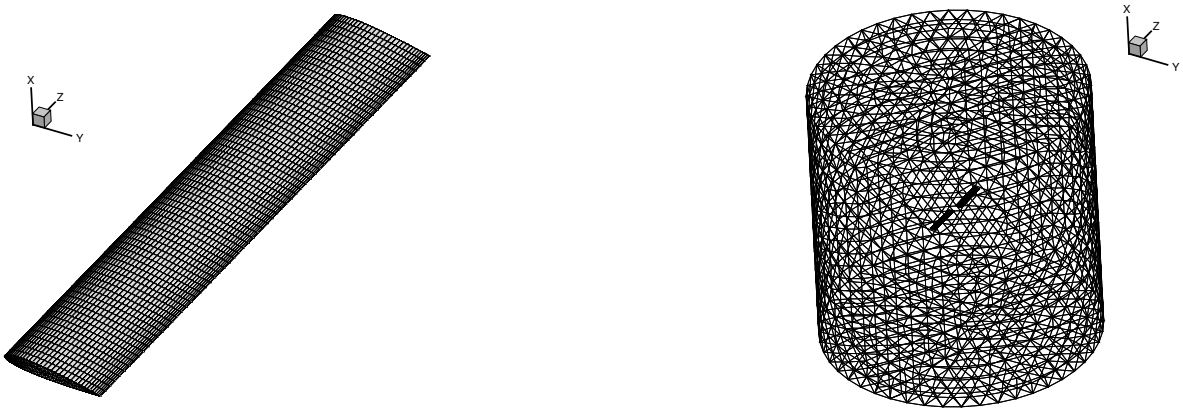


Figure 5.49: Schematic of the Caradonna-Tung rotor with twin blades.

(84, 953 nodes and 563, 978 edges). The near-field and far-field views of the computational mesh are shown in Figure 5.50. Although not shown here, the numerical results are grid independent with less than 5% difference in the predicted lift and drag coefficients when the grid resolution is doubled.

Here, only Euler solutions are considered and the contours of static pressure on the top and bottom surfaces of each blade are shown in Figures 5.51 and 5.52. As can be seen, the flow remains subsonic on the entire blade for the 1250 rpm case. On the other hand, for the 2500 rpm case, the flow is subsonic for almost 80% of the span before becoming transonic in the region closer to the tip of the blade.

In order to validate the numerical results obtained using the UNPAC solver, the surface pressure coefficients, C_p, at different spanwise locations are compared to the experimental data. Here, the dynamic pressure used for calculating the pressure coefficient is defined as $p_{\text{dynamic}} = \frac{1}{2}\rho_\infty(\Omega r)^2$ where r is the radial distance of each cross-section from the axis of rotation. The results in terms of the C_p distributions at three spanwise locations, i.e., 80%,



(a) Surface mesh with $4,692$ quadrilateral and 408 triangular cells

(b) Computational domain with $9,384$ pyramid and $458,375$ tetrahedral cells

Figure 5.50: Near-field and far-field views of the computational domain depicting the surface and volume meshes for the Caradonna-Tung rotor case.

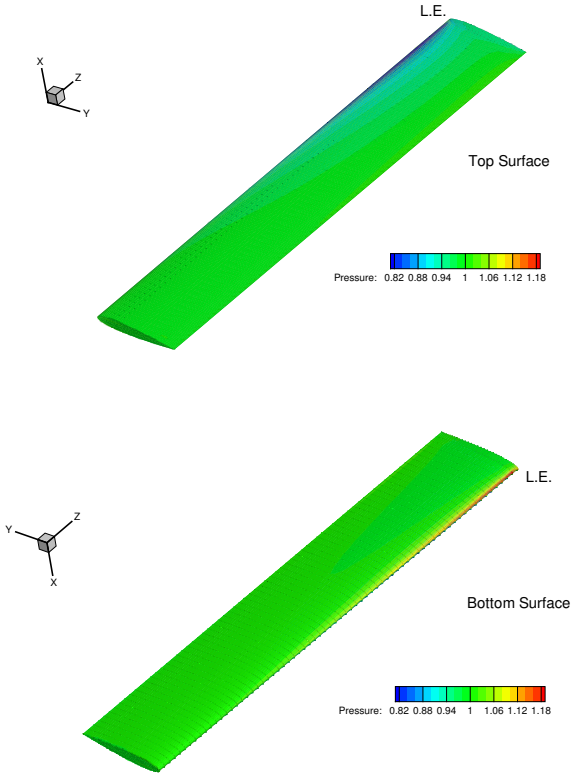


Figure 5.51: Contours of static pressure on the suction (top) and pressure (bottom) sides of the Caradonna-Tung rotor blade for the 1250 rpm case.

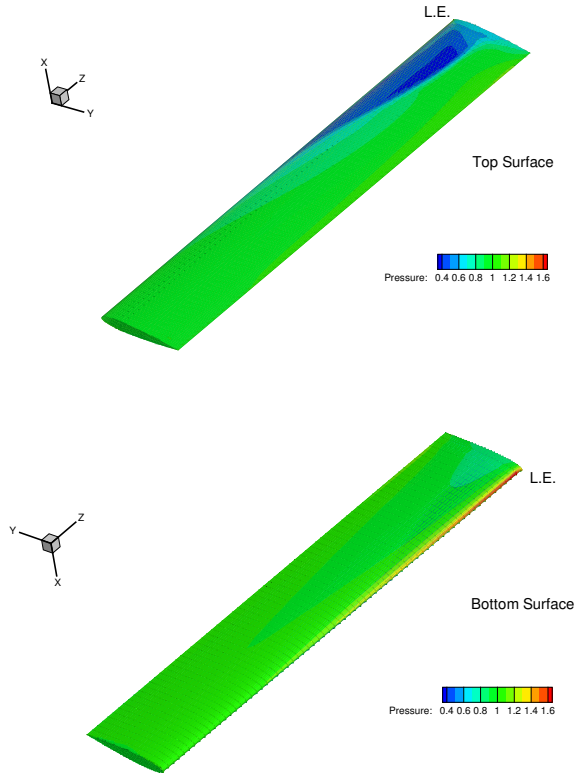
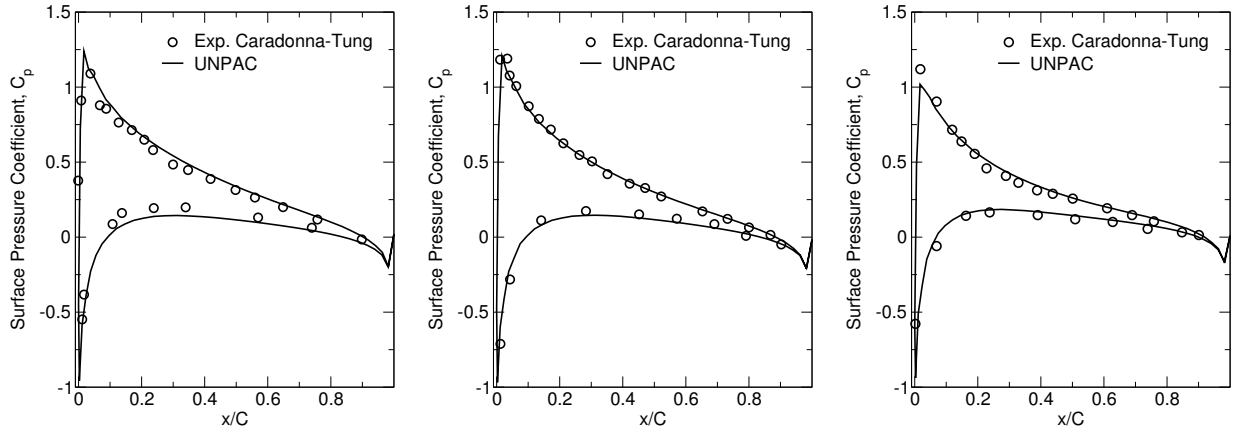


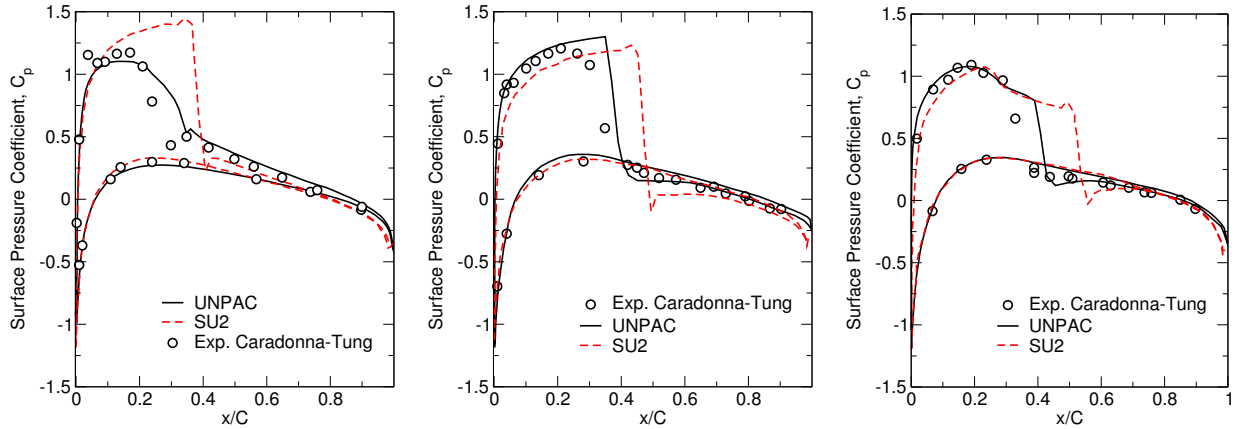
Figure 5.52: Contours of static pressure on the suction (top) and pressure (bottom) sides of the Caradonna-Tung rotor blade for the 2500 rpm case.

89%, and 96% span, are shown in Figures 5.53 and 5.54 for the 1250 rpm and 2500 rpm cases, respectively.



(a) 80% span **(b)** 89% span **(c)** 96% span

Figure 5.53: Coefficient of pressure distribution for the Caradonna-Tung rotor in hover at 1250 rpm (tip Mach number of 0.439).



(a) 80% span **(b)** 89% span **(c)** 96% span

Figure 5.54: Coefficient of pressure distribution for the Caradonna-Tung rotor in hover at 2500 rpm (tip Mach number of 0.877).

As can be seen, for the hover case at the lower rotational speed, the flow remains subsonic on the entire blade and a very good agreement is obtained between the UNPAC results and the experimental data [38]. As for the 2500 rpm case with the tip Mach number of $M_{\text{tip}} = 0.877$, the flow becomes transonic near 80% span. Once again, a reasonable agreement is achieved for the Euler solutions obtained using the UNPAC solver and the experimental results. The only exception is in the vicinity of the shock which is captured at a further downstream location. This is consistent with the other Euler solutions reported in the literature [61, 58]. Additionally, a comparison is made between the UNPAC results and those using the Stanford University Unstructured (SU2) solver with the same mesh. As can be seen in Figure 5.54, the UNPAC results show a better agreement with the experimental data.

Chapter 6

Sensitivity Analysis Results

In this chapter, the proposed FDOT toolbox is applied to a few different numerical solvers to demonstrate its advantages. The goal here is to calculate the sensitivities of the cost function with respect to the design variables using FDOT. Different test cases based on various numerical solvers are presented here. Herein, the validation of the sensitivity analysis results obtained using the FDOT toolbox applied to several different test cases with various levels of complexity is sought.

As the first test case, a simple iterative process based on Newton's method is considered. A non-linear objective function is defined and the entire process is differentiated algorithmically using FDOT. Additionally, other test cases including two-dimensional heat diffusion and quasi-1D inviscid compressible flow through a nozzle are investigated. For the case involving compressible flow through a converging-diverging nozzle, the sensitivities calculated with the present method are compared to those obtained using a continuous adjoint approach for verification purposes. In all cases, the efficiency and the ease of use/integration of the FDOT module are demonstrated. Finally, the FDOT toolbox is coupled with a 2D structured RANS/Euler solver and adjoint solutions are obtained for an inviscid transonic flow past NACA0012 airfoil.

6.1 Newton's Iteration

Here, a simple iterative algorithm is considered to demonstrate our proposed automatic differentiation approach based on operator overloading. As discussed earlier, most CFD procedures can be viewed as a three-step process involving an iterative core that potentially converges to a final numerical solution used in the calculation of the objective or the cost function. Therefore, in general the entire solver would consist of (1) pre-iterative, (2) iterative and (3) post-iterative procedures followed one after the other. For the first test case, a simple pre-iterative process is assumed that consists of a non-linear function, u, operating on the input variable, x, such that

$$\begin{aligned} & x \Rightarrow \text{input variable} \\ & \textbf{pre-iterative:} \quad u = u(x) = x^2 \end{aligned} \tag{6.1}$$

Next, an iterative process working on variable y whose fully converged solution is assumed available, i.e., y^*, along with x and u are used to define an objective. For simplicity, the fully converged solution of the iterative variable is assumed which is basically the solution to nonlinear equation:

$$f(y, u) = (y - u)^2 = 0 \tag{6.2}$$

The above equation can be solved using Newton's method with the initial guess y_1. That is,

$$y_{k+1} = y_k - \frac{f(y_k, u)}{f'(y_k, u)} \quad \text{with} \;\; k = 1, 2, ... \tag{6.3}$$

so that

$$y_{k+1} = y_k - \frac{(y_k - u)^2}{2(y_k - u)} = y_k - \frac{1}{2}(y_k - u) = \frac{1}{2}y_k + \frac{1}{2}u \tag{6.4}$$

The analytical solution of Eq. (6.2) is clearly $y = u$ and the Newton's method will converge to this solution assuming that the initial guess, y_1, is close enough to u. Finally,

the iterative process is followed by a post-iterative procedure that calculates the arbitrarily chosen nonlinear cost function given as

$$c = g(y^*, u, x) = \sin(y^*) + u \tag{6.5}$$

where y^* is the fully converged solution.

Based on the definition of the pre-iterative function and the solution of $f(y, u) = 0$, one can show that the final cost function will only depend on the input variables such that

$$c = \sin(y^* = u) + u = \sin(u) + u = \sin(x^2) + x^2 \tag{6.6}$$

The goal here is to find the gradient of the cost function with respect to the input variable, x. Using the above equation, the exact derivative is

$$\frac{dc}{dx} = 2x\left[\cos(x^2) + 1\right] \tag{6.7}$$

Further assuming that $x = 1.0$, the exact gradient value can be computed as

$$u = y^* = 1.0 \;\Rightarrow\; \frac{dc}{dx} = 2\left[\cos(1) + 1\right] = \mathbf{3.080604611736280}$$

The entire process explained here can be simply coded using Fortran (or C++) as the primal solver where the Newton's method is performed for N iterations. One of the many attractive features of the proposed technique is the ease of implementation. Since all operators are overloaded to handle the derived-type variable computations, the *tape* automatically records the expression tree for all instructions. Therefore, one only needs to follow the steps discussed in Section 4.2.3 to use FDOT for adjoint computations. First, all real-typed variables are replaced with variables of type **AReal**. Next, the iterative variable, y, is flagged and the checkpoints are set before and after the iterative loop. Finally, after the post-iterative process, the adjoint value of the cost function is set to unity and the recorded tape is evaluated in reverse by accumulating all adjoint variables. The nominal code for this simple problem is provided below in its entirety.

```
integer :: k, N
real    :: x, u, y, c

x = 1.0
u = x**2
y = 2.0  ! initial guess for Newton's iterations

do k = 1,N
   y = 0.5*y + 0.5*u
enddo

c = sin(y) + u
```

Pseudo-code 6.1: Simple iterative process (primal code)

There are two important issues that must be noted here. First, as mentioned earlier, the adjoint code must be initiated using the fully converged solution from the primal solver. Second, the iterative loop only needs to be executed for a single pass. This is the biggest advantage of the proposed technique to significantly reduce the size of the recorded tape, thus decreasing the memory footprint and increasing the computational efficiency. As a result, the adjoint code based on the nominal solver needs only the minor modifications that are presented below.

During the adjoint computations (reverse tape evaluation), changes in the adjoints of the solution variable ($\overline{y}$) are monitored for convergence. Here, the initial condition for the Newton's solver is taken to be $y_1 = 2.0$. As a result, the nominal solver takes about 50 iterations to converge to machine accuracy. For adjoint computations, a similar convergence limit is used, i.e., $\log_{10}(|\overline{y_k} - \overline{y_{k+1}}|) \approx -15$. Figure 6.1 presents the convergence histories for the primal and adjoint solvers for the simple problem considered here. While theoretically the Newton's method should have a quadratic convergence rate, since the solution has a double root, the solver exhibits a linear convergence.

```
   integer       :: k
   type(areal)   :: x, u, y, c

   x = 1.0
   u = x**2
   y = 1.0 ! solution from the primal solver
   call set_input_variable(y.index)
!
   call set_checkpoint
!
   do k = 1,1
      y = 0.5*y + 0.5*u
      call set_output_variable(y.index)
   enddo
!
   call set_checkpoint
!
   c = sin(y) + u
!
   tape(c.index).adjoint = 1.0
!
   call evaluate_tape
```

Pseudo-code 6.2: Simple iterative process (FDOT-enabled code)

In any case, the adjoint solver has the same convergence rate as the nominal solver, which is expected.

Table 6.1 shows that the sensitivity value obtained using the FDOT toolbox is in close agreement (up to 13 decimal places) with the exact value. In order to further analyze the robustness of the proposed technique, the same case is automatically differentiated using

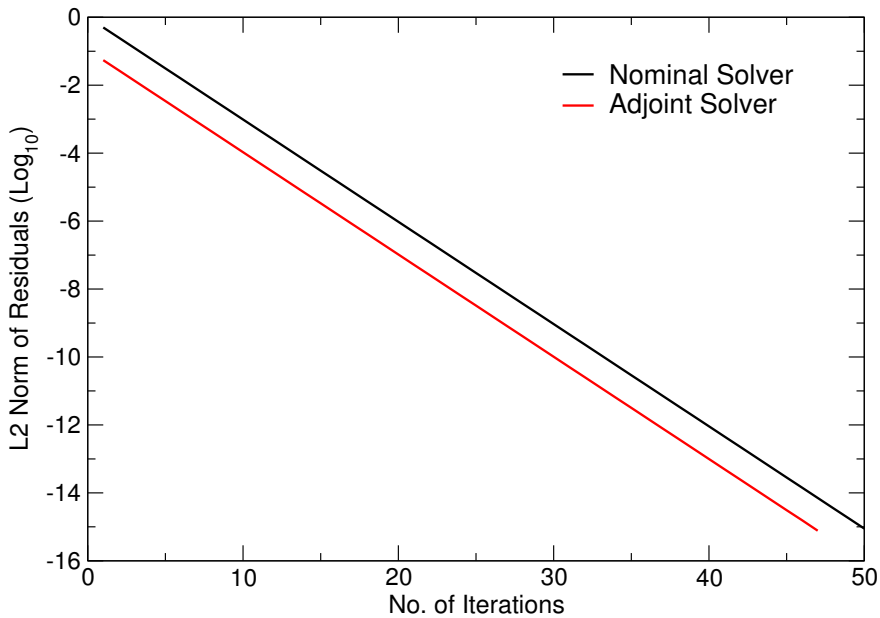


Figure 6.1: Convergence histories of the nominal and adjoint solvers for the simple iterative problem

Table 6.1: Comparison of sensitivity values between FDOT and exact solutions for the simple iterative case.

Sensitivity	Exact	FDOT
$\frac{dc}{dx}$	**3.080604611736280**	**3.080604611736272**

the conventional operator overloading approach. In the conventional approach, the entire iterative convergence loop is unrolled and all the instructions are recorded in the tape. This significantly increases the memory footprint as well as the overhead in writing and reading the tape. Timing results for the nominal solver and two different versions of the OO/AD-based adjoint solver are given in Table 6.2. Also presented are the memory requirements for recording the tape. Note that in the current implementation, each tape entry requires 32 bytes of memory (integers and reals only).

For the case considered here, it is seen that the proposed technique can greatly reduce the memory footprint (for more than 20 times compared to a traditional OO/AD approach) and the computational cost of adjoint computations is about 1.1 times the cost of the nominal solver. In contrast, the computational cost of the traditional OO/AD approach is around 2.8 times the cost of the nominal solver. One thing to note here is that for most problems

Table 6.2: CPU timings and memory footprints for the nominal and adjoint solvers using conventional and the proposed approaches; simple iterative case.

Solver Type	CPU Time (ms)	Normalized CPU Time	Tape Memory	Normalized Memory
Nominal Solver	1.68	1.0	-	-
Conventional OO	4.74	2.82	9888 Bytes	**20.6**
FDOT	1.81	**1.08**	480 Bytes	1.0

of interest there is no need for sensitivities to be accurate to machine precision. Generally speaking, sensitivities that are accurate up to six or seven significant digits would be enough to be used in the gradient-based optimization process. In such cases, the computational times using FDOT would be even lower than what is presented.

6.2 Heat Diffusion

The second test case studied is the two-dimensional, steady, constant property heat diffusion problem that is governed by an elliptic partial differential equation (Laplace equation). Thus, using Dirichlet boundary conditions and no internal heat generation, the governing equation is given as

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \ , \quad 0 < x < L \ ; \ 0 < y < H \tag{6.8}$$

subject to

$$T(x = 0, y) = T(x = L, y) = T(x, y = 0) = T_{side}$$
$$T(x, y = H) = T_{top}$$

where T is the temperature, L is the length and H is the height of the plate.

First, using an equally-spaced grid in each direction, a uniform mesh of size M is created for the physical domain with $L = H = 1.0$ unit length. Next, central finite-difference approximations are used to discretize the governing equation over the computational domain.

Finally, a Jacobi iteration approach is used to solve the penta-diagonal system of equations. Assuming that $T_{side} = 100$ and $T_{top} = 0$, one can determine the temperature distribution at the interior nodes. Finally, using the fully converged temperature distribution along the horizontal axis a cost function, C_T (in unit temperature times unit length) is defined as

$$C_T = \int_0^L T(x, y = \frac{H}{2})\, dx \tag{6.9}$$

The goal here is to calculate the sensitivities of this cost function with respect to the top and side face temperature values, i.e., $\partial C_T/\partial T_{top}$ and $\partial C_T/\partial T_{side}$. Next, FDOT is used to iteratively evaluate the discrete adjoint sensitivities. Similar to the first case, the main changes to the primal code can be listed as:

1- Change real-typed variables to type **AReal**.

2- Read in the fully converged solution (from the primal solver).

3- Flag the iterative variables whose values change during iterations (temperature at the interior nodes for the present case).

4- Set checkpoints before and after the iterative loop.

5- Set the adjoint of the cost function to unity and evaluate the tape.

To study the effects of grid size on the timing results and memory footprints, three different grid resolutions of 10×10, 50×50, and 100×100, are considered. It is useful to note again that the computational effort to obtain all sensitivities is almost independent of the number of design variables. However, only two sensitivities ($\partial C_T/\partial T_{top}$ and $\partial C_T/\partial T_{side}$) are considered here for the purpose of verification. Note that the sensitivities can also be calculated using finite differences. This can be done either with a 1st order forward difference or a 2nd order central difference approximation. The perturbation parameter for both schemes is taken to be 10^{-8} (unit temperature). The sensitivity values from FDOT toolbox are compared against those from finite difference (FD) approximations and the obtained results are shown in Table 6.3 for all three grid resolutions. As can be seen, the sensitivity values obtained using FDOT agree very well with the finite difference approximations for

Table 6.3: Comparison of sensitivity values obtained from FD approximations (1st and 2nd order) and FDOT toolbox for the 2D heat diffusion problem.

M	Sensitivity	FD (1st order)	FD (2nd order)	FDOT
10	$\partial C_T/\partial T_{top}$	**0.11338314651**	**0.11338304934**	**0.11338304969**
	$\partial C_T/\partial T_{side}$	**0.88661691002**	**0.88661693185**	**0.88661693184**
50	$\partial C_T/\partial T_{top}$	**0.15141975823**	**0.15141974288**	**0.15141974285**
	$\partial C_T/\partial T_{side}$	**0.84854972047**	**0.84854972204**	**0.84854972201**
100	$\partial C_T/\partial T_{top}$	**0.15663158512**	**0.15663158601**	**0.15663158608**
	$\partial C_T/\partial T_{side}$	**0.84265874150**	**0.84265874049**	**0.84265874047**

all grid resolutions. As noted in the introduction, the finite difference approximations are highly dependent on the value of the perturbation parameter while AD-based adjoints are accurate to machine precision.

Next, the CPU timings are investigated for the nominal and adjoint solvers and compare the memory footprints for the conventional OO-based adjoint evaluations and the novel approach introduced in this work (see Table 6.4). The results presented clearly show that

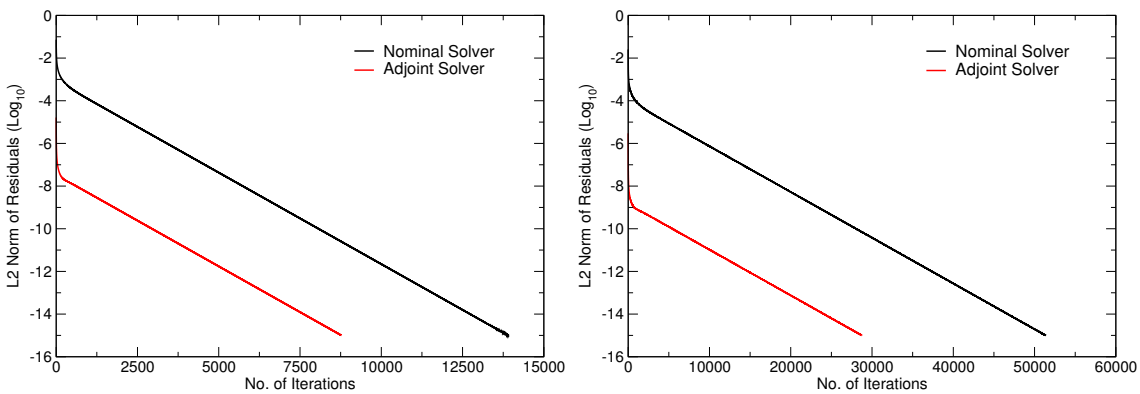
Table 6.4: CPU timings and memory footprints for the nominal and adjoint solvers using conventional and the proposed approaches; 2D heat diffusion case.

Grid Size M	Solver Type	CPU Time (ms)	Normalized CPU Time	Memory Footprint (Tape only)	Normalized Memory Footprint
10	Nominal CFD	9.51	1.0	-	-
	Conventional OO	54.6	5.74	11.25 MBytes	**443**
	FDOT	10.85	**1.141**	26 KBytes	1.0
50	Nominal CFD	385	1.0	-	-
	Conventional OO	2853	7.41	7.5 GBytes	**11,331**
	FDOT	565	**1.467**	694 KBytes	1.0
100	Nominal CFD	2,895	1.0	-	-
	Conventional OO	25,436	8.78	60.1 GBytes	**22,625**
	FDOT	5,535	**1.911**	2.72 MBytes	1.0

the approach is not only efficient computationally but it also offers huge memory savings. For this particular example, the total adjoint computation time is not more than 2 times that of the nominal PDE solver, while the memory requirement can be as little as 1/20,000

of the traditional OO adjoint approach. These features are essential for application of this technique to larger scale CFD solvers.

Next, convergence histories for the nominal and the adjoint solvers are compared in Figure 6.2 for cases with $M = 50$ and $M = 100$. Both solvers have the same convergence rate as expected.



(a) $M = 50$ **(b)** $M = 100$

Figure 6.2: Convergence histories of the nominal and adjoint solvers for the 2D heat diffusion problem.

6.3 Quasi-1D Euler Solver

Next, compressible flow inside a nozzle is investigated which is governed by the quasi-1D Euler equations given in their conservation form as

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}}{\partial x} = \vec{S} \tag{6.10}$$

where $\vec{U}$ is the vector of conservation variables, $\vec{F}$ is the convective flux vector and $\vec{S}$ is the vector of source terms defined as

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho E \end{bmatrix} \quad ; \quad \vec{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u h \end{bmatrix} \quad ; \quad \vec{S} = - \begin{bmatrix} \rho u \\ \rho u^2 \\ \rho u h \end{bmatrix} \frac{1}{A} \frac{dA}{dx}$$

where ρ is the fluid density, u is the fluid velocity, p is the pressure, E is the total energy and h is the specific enthalpy. Using the ideal gas assumption, the pressure can be related to conservation variables through

$$p = (\gamma - 1)\left[\rho E - \frac{\rho u^2}{2}\right] \tag{6.11}$$

where γ is the specific heat ratio. The specific enthalpy is also defined as

$$h = \frac{\rho E + p}{\rho} \tag{6.12}$$

The cross-sectional area of the nozzle A defines the geometry. The governing equations given in Eq. (6.10) are first discretized using a cell-centered finite-volume approach over an equally-spaced computational grid with M cells. To obtain the steady-state solution, the governing equations are marched in pseudo-time using a 4-stage explicit Runge-Kutta scheme. To eliminate the odd-even decoupling due to the central difference approximation and to capture shocks, the artificial viscosity of Jameson-Schmidt-Turkel [117] is added to the inviscid flux terms. Additionally, the convergence is accelerated using local time stepping. For the purpose of sensitivity analysis, an objective function is defined which is taken to be the integral of pressure along the nozzle

$$I = \int_{\Omega} p \, dx \tag{6.13}$$

Adjoint Sensitivity Analysis for a Diverging Nozzle

As the first nozzle flow test case studied here, a diverging nozzle with length $L = 10$ is considered with its cross-sectional area given by

$$A(x) = 1.398 + 0.347 \tanh(0.8x - 3.2) \; ; \qquad 0 \leq x \leq 10 \tag{6.14}$$

Also the following physical boundary conditions for this problem are considered

$$M_{in} = 1.25 \quad \text{at} \quad (x = 0.0) \qquad M_{exit} = 0.45 \quad \text{at} \quad (x = 10.0)$$

Although the optimization process is not pursued here, the derivatives of the cost function with respect to the design variables are required in the framework of a gradient-based optimization algorithm. Needless to say, for the 1D case with M grid cells, $M + 2$ grid nodes are defined and two of these nodes are at the fixed boundary locations which leaves us with M interior nodes. Thus, the goal here is to find the derivatives of the pressure integral [see Eq. (6.13)] with respect to all interior grid nodes, x_i, with $i = 1, \ldots, M$.

Table 6.5: Comparison of sensitivity values obtained from FD approximation (2nd order) and FDOT toolbox for the quasi-1D flow through a divergent nozzle.

M	Sensitivity	FD (2nd order)	FDOT
10	$\partial I/\partial p_1$	**-0.139789975**12287	**-0.139789975**73139
	$\partial I/\partial p_2$	**-1.356323834**140714	**-1.356323834**370082
	$\partial I/\partial p_3$	**1.696427703**508123	**1.696427703**869594
100	$\partial I/\partial p_1$	**-0.133166153**68953	**-0.133166153**53033
	$\partial I/\partial p_2$	**0.220898422**92250	**0.220898422**49454
	$\partial I/\partial p_3$	**0.009573344**32422	**0.009573344**10879
500	$\partial I/\partial p_1$	**-0.031511299293**331	**-0.031511299293**867
	$\partial I/\partial p_2$	**0.051404115**18564	**0.051404115**92566
	$\partial I/\partial p_3$	**0.001922663**915125	**0.001922663**910520

Once again, in order to study the effects of the grid size on the performance of the AD toolbox, three different grid resolutions with $M = 10,\ 100,$ and 500 are considered. As explained earlier, the adjoint method is capable of evaluating the derivatives of the cost function with respect to all input as well as intermediate variables in a single evaluation of the recorded tape in the reverse order. However, the sensitivity results of the objective function at three equally spaced locations along the nozzle length, specifically p_1 at $x = L/4$, p_2 at $x = L/2$, and p_3 at $x = 3L/4$, are presented here. Again, for verification purposes second-order finite difference approximations are calculated and compared to those from the FDOT toolbox. Here, the discrete adjoint runs are stopped when the adjoint residuals reach 10^{-12}. Numerical tests have shown that such tolerance guarantees sensitivities that are adequate for gradient-based aerodynamic optimization. As can be seen in Table 6.5, the obtained results from FDOT are in excellent agreement with those from finite difference approximations for all three grid resolutions.

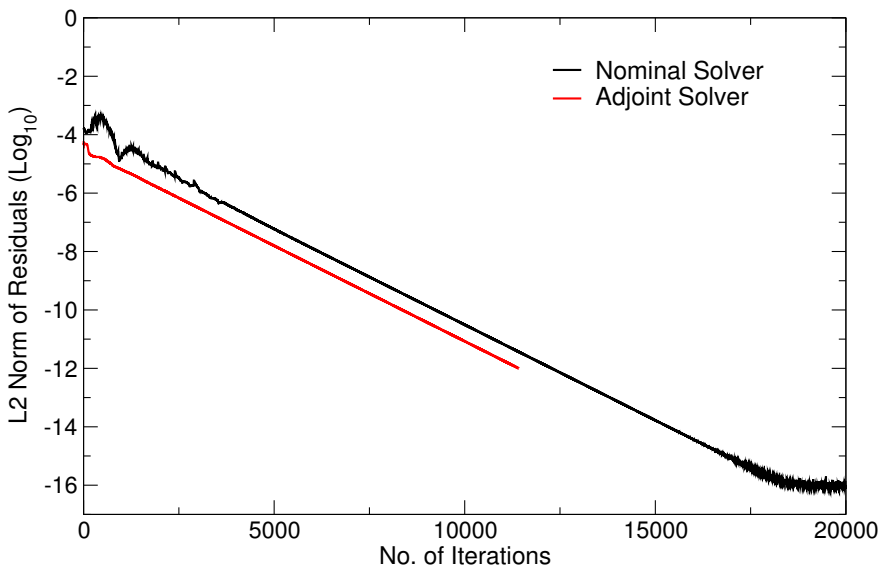


Figure 6.3: Convergence histories of the nominal and adjoint solvers for the quasi-1D nozzle flow with $M = 500$

Next, the convergence of the nominal and adjoint solvers are plotted against each other in Figure 6.3 for the finest grid case with $M = 500$. Once again, it can be seen that the adjoint solver has the same convergence rate as the nominal solver.

Finally, the CPU times for the nominal and adjoint solvers are measured. Moreover, the memory required to store the recorded tape for each case using our novel approach is presented. As shown in Table 6.6, the proposed technique is quite efficient since the sensitivity of the objective function to 500 design variables can be computed at a cost that is only 3 times that of the CFD solver. In addition, the tape size is quite manageable requiring less than 16 MB of RAM storage in all cases. These results once again demonstrate the memory and computational efficiency of the developed technique. Similar to previous examples, the adjoint solver was developed by adding less than 10 lines of code to the CFD solver.

6.3.1 Adjoint Sensitivity Analysis for a Converging-Diverging Nozzle

To further analyze the accuracy of the new discrete adjoint approach, a series of standard converging-diverging nozzle flow test cases are presented next. For the purpose of validation

Table 6.6: CPU timings and memory footprints for the nominal and adjoint solvers using the proposed approaches, quasi-1D nozzle flow case.

Grid Size M	Solver Type	CPU Time (ms)	Normalized CPU Time	Memory (Tape only)
10	Nominal	79.02	1.0	-
	FDOT	81.26	**1.028**	379 KBytes
100	Nominal	471	1.0	-
	FDOT	945	**2.006**	3.2 MBytes
500	Nominal	3,944	1.0	-
	FDOT	11,956	**3.031**	15.7 MBytes

and verification, the sensitivity results from a continuous adjoint approach [120] and finite difference approximations are compared to our discrete adjoint results. For these cases, area variation of the converging-diverging nozzle is given by

$$A(x) = \begin{cases} 2 & 0 \leq x \leq 0.5 \\ 1 + \sin^2(\pi(x - 1.0)) & 0.5 < x < 1.5 \\ 2 & 1.5 \leq x \leq 2.0 \end{cases} \tag{6.15}$$

Characteristic-based boundary conditions with prescribed stagnation pressure and stagnation temperature at the inlet and prescribed static pressure at the exit are used here. Three cases that have been previously studied by Kaminsky et al. [120] are considered. These include fully subsonic flow, subsonic-to-supersonic flow and transonic flow with shock cases. The boundary condition settings for these cases are presented in Table 6.7.

Table 6.7: Boundary conditions for the three converging-diverging nozzle flow cases.

Case	p_0	T_0	p_{exit}
1	1.0	1.0	0.9899
2	1.0	1.0	0.5200
3	1.0	1.0	0.8432

It is worth mentioning that the same cases were also studied by Giles and Pierce [77], and Lozano and Ponsin [139] using a continuous adjoint approach. To be able to compare our discrete adjoint sensitivity results to those from the continuous adjoint approach, the

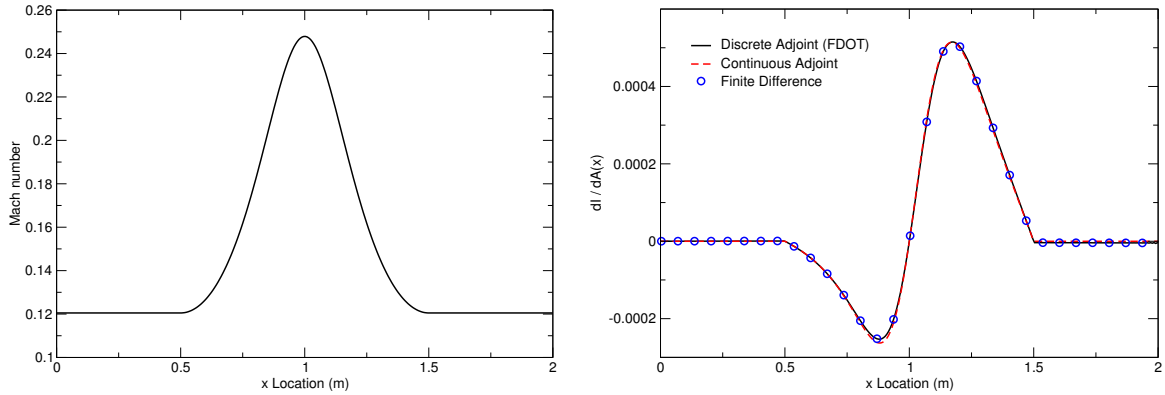
process of evaluating derivative information using the continuous adjoint solution needs to be discussed first. As explained earlier, in the continuous adjoint approach, the flow equations are linearized and the resulting analytic adjoint equations are discretized and solved numerically to obtain the adjoint solutions. Compared to the discrete adjoint method, the continuous approach requires a more complicated process that includes development of somewhat complex boundary conditions. Moreover, the numerical adjoint solutions, $\boldsymbol{\psi}$, must be post-processed to compute the desired sensitivities. Having the continuous adjoint solution, $\boldsymbol{\psi}$, the derivatives of the cost function with respect to the design variables can be found via [139]

$$\frac{\partial I}{\partial \vec{\alpha}_N} = \sum_{i=1}^{M} \vec{\psi}_i^T \left(\frac{\vec{R}_i(\vec{U}(x), \vec{\alpha}_N + \epsilon) - \vec{R}_i(\vec{U}(x), \vec{\alpha}_N)}{\epsilon} \right) \tag{6.16}$$

where $\vec{\alpha}_N$ are the N design variables.

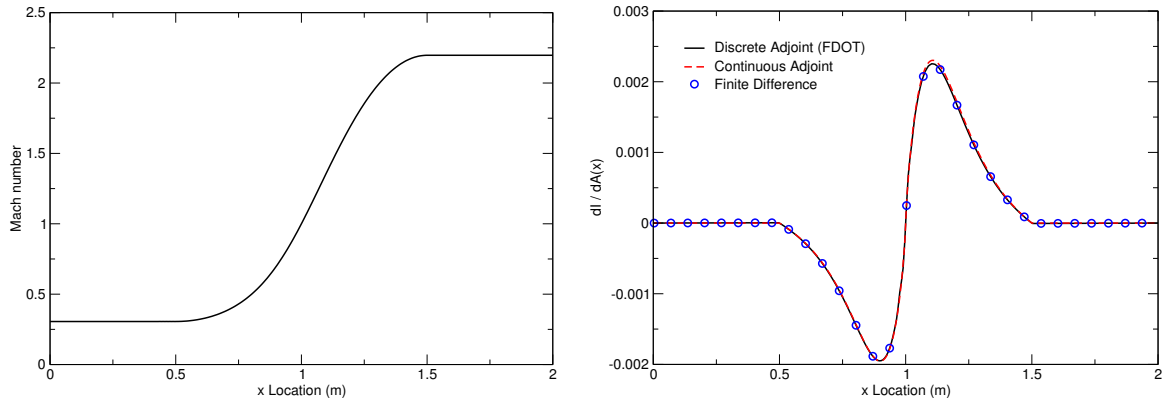
Here, the design variables are taken to be the cross-sectional area at each discrete x location, i.e., $A(x_M)$. For simplicity, the number of design variables is taken to be equal to the number of cells, i.e., $N = M$. However, in a practical application, shape parameterization techniques such as Hicks-Henne bump functions, Bezier or B-Spline functions are normally used to reduce the number of design variables. Similar to finite difference approximations, the perturbation parameter ϵ is tuned over a large range of values to ensure the independence of the obtained results. Additionally, the residual vectors $\vec{R}(\vec{U}(x), \vec{\alpha}_N)$ and $\vec{R}(\vec{U}(x), \vec{\alpha}_N + \epsilon)$ refer to the residuals of the flow solver for the original geometry and the geometry perturbed at the N-th design variable, respectively. As explained by Lozano and Ponsin [139] both residual vectors should be obtained using the unperturbed fully converged flow solution.

Finally, the calculated sensitivities using the FDOT toolbox for the cost function with respect to all design variables along the nozzle are compared against those evaluated using the continuous adjoint solutions and the finite difference approximations. These results as well as the flow solution (Mach number variation along the nozzle) are shown in Figures 6.4 through 6.6 for all three cases studied. As can be seen, there is close agreement between the discrete adjoint sensitivities obtained from our proposed technique and the finite difference approximations, which are determined at every 10 grid nodes. Moreover, a very good



(a) **(b)**

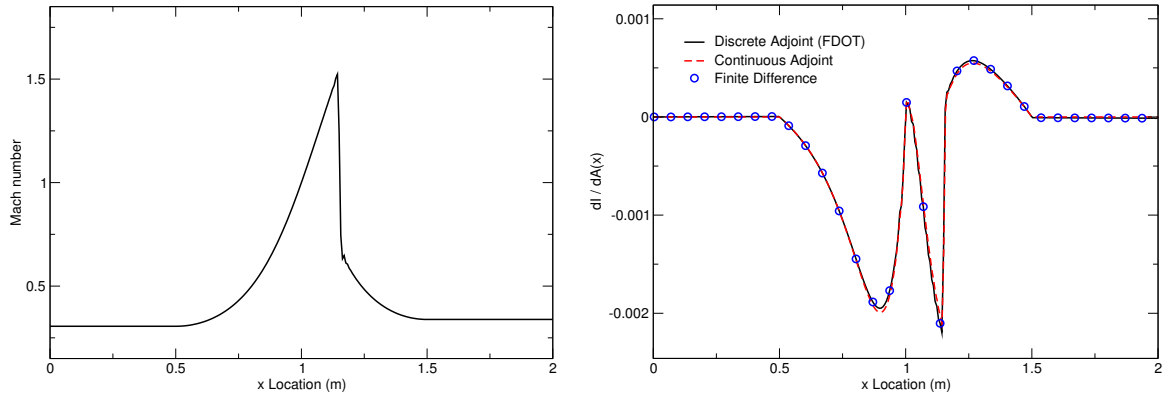
Figure 6.4: (a) Mach number distribution for the fully subsonic nozzle flow (case 1) with $p_{\text{exit}} = 0.9899$; (b) comparison of sensitivity results using the present discrete adjoint approach (FDOT toolbox), continuous adjoint [120], and finite difference approximations.



(a) **(b)**

Figure 6.5: (a) Mach number distribution for the transonic nozzle flow (case 2) with $p_{\text{exit}} = 0.52$; (b) comparison of sensitivity results using the present discrete adjoint approach (FDOT toolbox), continuous adjoint [120], and finite difference approximations.

agreement between the discrete and continuous adjoint results is observed. It must be noted that the sensitivity results from the discrete and continuous approaches might vary due to the fact that the discretization and linearization steps are generally non-commutative [139, 120]. Since the discrete adjoint method calculates the exact derivatives of the discrete governing equations, the discrete adjoint results are suspected to be slightly more accurate compared to the continuous adjoint results. For brevity, the CPU time comparisons are not included for these cases because they are nearly identical to those reported in Table 6.6.



(a) **(b)**

Figure 6.6: (a) Mach number distribution for the nozzle with shocked flow (case 3) with $p_{\text{exit}} = 0.84317$; (b) comparison of sensitivity results using the present discrete adjoint approach (FDOT toolbox), continuous adjoint [120], and finite difference approximations.

6.4 2D Euler Solver

Finally, and as the last test case for adjoint sensitivity calculations, the inviscid transonic flow past the NACA0012 airfoil is considered. For this case, the FDOT toolbox is coupled with a 2D structured compressible Navier-Stokes solver. This solver is based on the finite-volume discretization of the Euler and Navier-Stokes equations on cell-vertex-based overlapping control volumes and the details of the numerical procedure are presented by Djeddi et al. [53, 54].

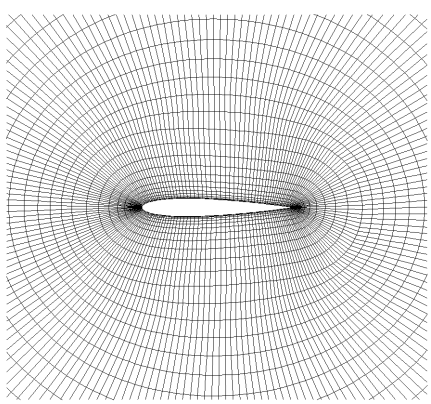


Figure 6.7: Near-field view of the structured grid used for the sensitivity analysis of flow past NACA0012 airfoil with 193×49 nodes.

Here, the FDOT module is added to the data structure of this CFD solver for the purpose of adjoint-based sensitivity analysis. An O-type grid with 193×49 nodes in the circumferential and radial directions is used (see Figure 6.7).

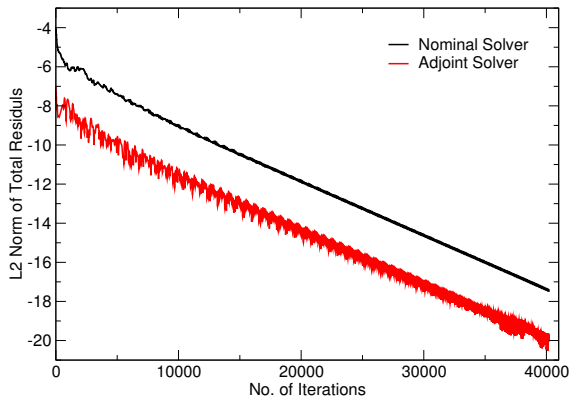


Figure 6.8: Convergence histories of the nominal and adjoint solvers for inviscid transonic flow past NACA0012 airfoil.

The drag coefficient, C_D, which is calculated in the post-processing stage is chosen as the cost function and is marked and passed to the FDOT toolbox. It takes about 40,000 iterations for the nominal CFD solver to converge to machine accuracy. With the fully-converged solution obtained, it is then passed to the FDOT toolbox for adjoint sensitivity calculations.

As discussed in the previous test cases, the first step in the FDOT module is to run one iteration of the overloaded converged CFD solution so as to record the expression tree into a tape. This tape is then rewound for the same number of iterations (40,000) and the sensitivity (gradient) information is calculated for all intermediate and design variables (all the **AReal** variables in the data structure). The convergence of the nominal and adjoint solvers are shown in Figure 6.8.

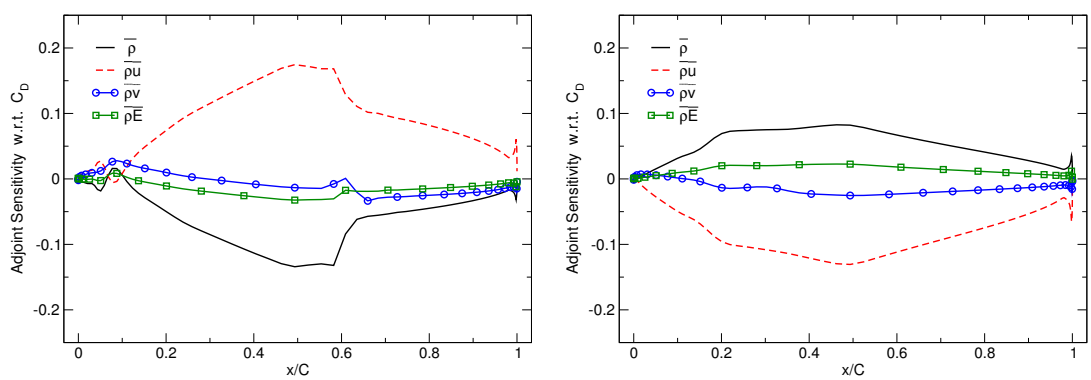
It must be noted that for this case with 37,828 degrees-of-freedom (DOF), the size of the recorded tape is only about 800 MBytes which is quite manageable by any standard workstation. In order to verify the accuracy of the computed sensitivities, a finite-difference approximation is performed using the nominal solver for the Mach number as the design variable. Therefore, using a first-order forward difference approximation with a perturbation of $\epsilon = 10^{-8}$, the sensitivity of the drag coefficient to the Mach number is calculated. The

results from the FDOT solver and the finite-difference approximation are compared against each other and are shown in Table 6.8.

Table 6.8: Sensitivity results using FDOT compared to finite-difference approximation for the inviscid transonic flow past NACA0012 airfoil.

Sensitivity	FD (1st order)	FDOT
$\dfrac{dC_D}{dM_\infty}$	**0.52089**<u>9829</u>	**0.52089**<u>8019</u>

As can be seen, there is a good agreement between the two results and the sensitivity values match up to five decimal places. It must be added that, the sensitivity results of FDOT are non-approximative, and therefore, supposedly much more accurate than the finite-difference result which is only a first-order accurate approximation.



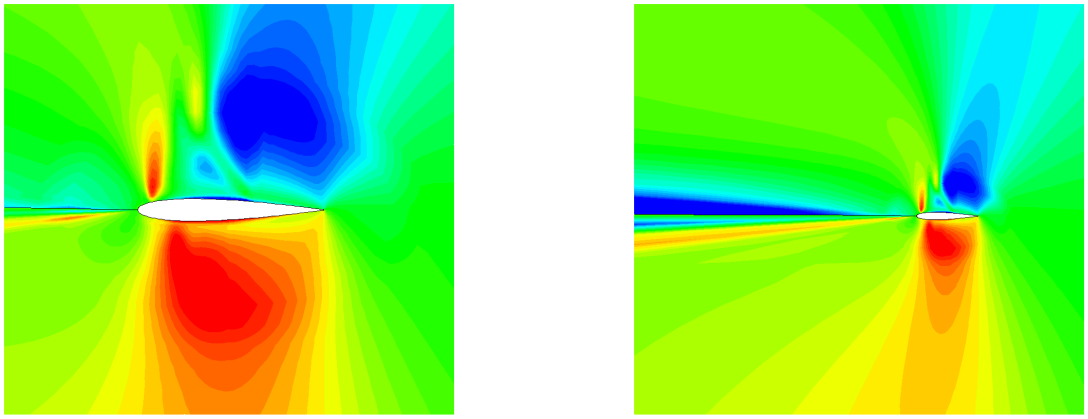
(a) suction side **(b)** pressure side

Figure 6.9: Sensitivities of the cost function (C_D) with respect to the flow variables on (a) the top surface and (b) bottom surface of the NACA0012 airfoil (inviscid transonic case at $M = 0.8$ and AOA $= 1.25$ deg.

Finally, the sensitivities of the conservation variables on the surface of the airfoil are presented. The distribution of these sensitivities on the suction and pressure sides of the airfoil are presented in Figure 6.9. As discussed in Section 5.1, the inviscid transonic flow past the NACA0012 at M = 0.8 and AOA = 1.25 degree leads to the formation of two shocks where a weak shock is at about 15% chord length from the leading edge on the pressure side and a strong shock is located at about 60% chord length on the suction side.

It can be clearly seen in Figure 6.9 that the sensitivities of the conservation variables on the surface of the airfoil also show discontinuities at the location of the two shocks. This

is even more pronounced for the continuity and momentum equations and agrees well with flow solution.



(a) near-field view **(b)** upstream view

Figure 6.10: Contour field of $\overline{\rho} = \dfrac{\partial C_D}{\partial \rho}$ for the inviscid transonic flow past NACA0012 airfoil at $M = 0.8$ and AOA $= 1.25$ deg.

Finally, the contour plots of the $\overline{\rho}$, i.e., $\dfrac{\partial C_D}{\partial \rho}$, are shown in Figure 6.10. A very interesting feature that is visible in these contour fields is that the flow sensitivities are stronger in the upstream rather than downstream of the airfoil. Obviously the drag coefficient, which is defined based on the integration of the pressure distribution on the surface of the airfoil, is mostly dependent on the free-stream velocity and the flow incidence angle. Therefore, as expected the magnitude of sensitivities are larger in the upstream. In other words, in the adjoint field, the wake is upstream of the airfoil while in the nominal flow field the wake is downstream. In fact, not only the location of the wake is reversed but the entire flow field also appears to be flipped [122]. This is consistent with the property of the adjoint characteristics, which have an opposite direction to the flow characteristics. Additionally, the upstream wake found in the adjoint sensitivity contour is typical of adjoint sensitivity contours as also seen in the literature [222, 122].

Chapter 7

Aerodynamic Shape Optimization

As discussed in the earlier chapters, the ultimate goal of this work is to develop an optimization framework for aerodynamic design applications. This framework will couple the UNPAC solver and the FDOT toolbox to offer a robust and efficient design tool based on discrete adjoint analysis. In what follows details of the UNPAC design optimization framework, **UNPAC-DOF**, are presented and several optimization test cases are considered.

7.1 Design Optimization Framework

Traditionally, aerodynamic design process has heavily relied on experimental wind tunnel tests and engineering judgment. With the advent of computational fluid dynamics, numerical shape optimization has been made possible without expensive and cost-prohibitive experiments and wind tunnel tests. Over the years, robust design methodologies have been proposed and used in aerodynamic shape optimization. Additionally, a whole field has been devoted to developing algorithms and black-box software packages used for numerical optimization [143].

In this work, the gradient-based optimization approach is considered. The UNPAC solver is used for obtaining nominal flow solutions. Furthermore, the FDOT toolbox and the CFD code are integrated into the UNPAC-AD framework to compute the gradient information. Finally, the UNPAC-OPT wrapper program is developed that couples the two solvers to perform design optimization.

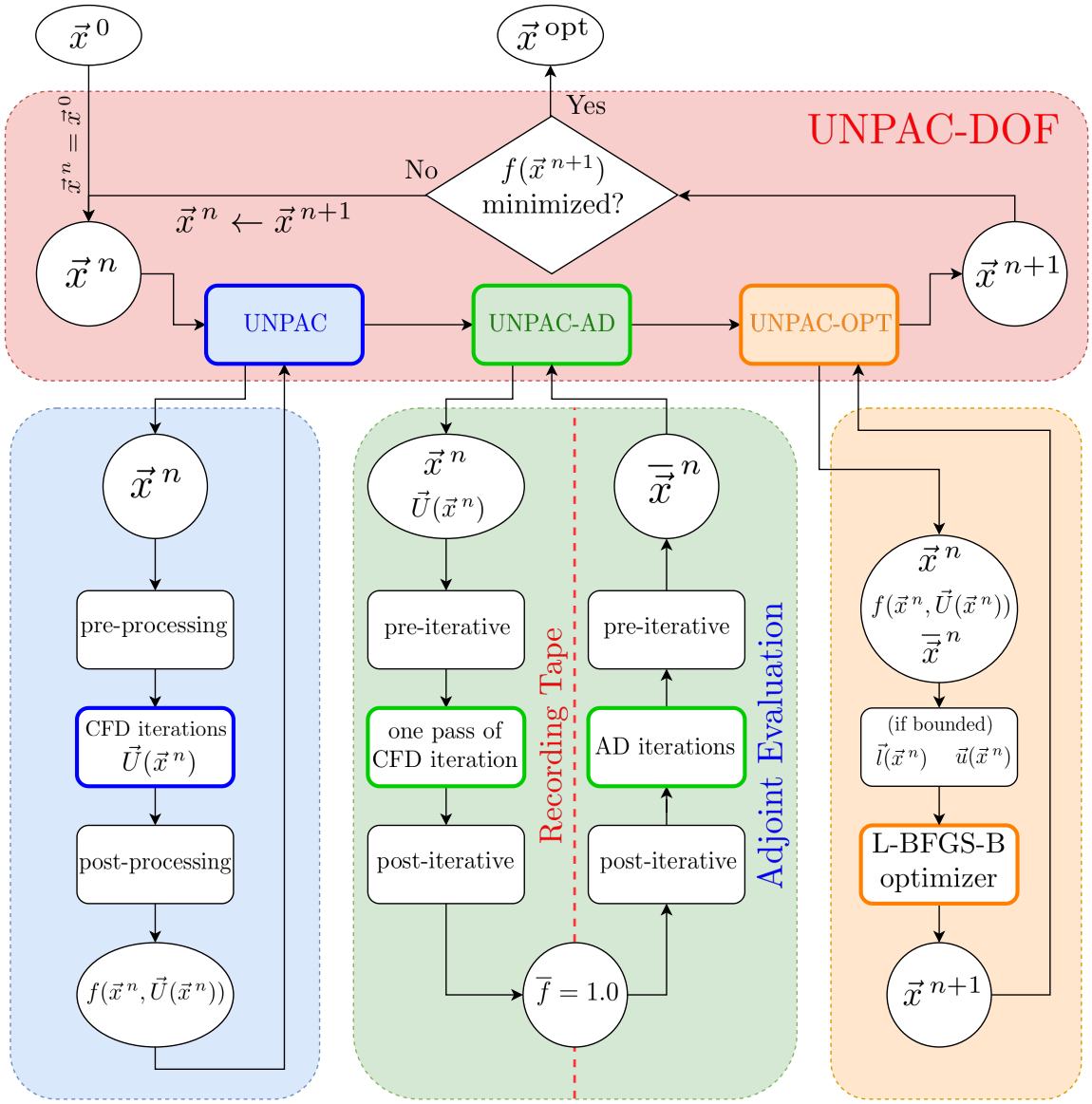


Figure 7.1: Flowchart of the **UNPAC D**esign **O**ptimization **F**ramework (UNPAC-DOF) and its three main components: UNPAC, UNPAC-AD, and UNPAC-OPT.

The UNPAC-OPT program uses a quasi-Newton method for optimization in both unbounded and bound constrained modes subject to upper and/or lower bounds for the design variables. The schematic of the present **UNPAC D**esign **O**ptimization **F**ramework (UNPAC-DOF) is provided in Figure 7.1.

The optimization framework seeks for optimal designs via an iterative process in which the following steps are considered for each design cycle:

1. The nominal CFD solver (UNPAC) is run to obtain the flow solution. In the first cycle, the initial design variables are used while in the subsequent cycles, the new values for the design variables (solution of the optimizer) are utilized. In the case of shape optimization, the design variables define the new geometry which will then be used to deform the computational mesh. This process is described in Section 7.1.1.

2. Using the flow solution, the UNPAC-AD solver is initiated which runs one pass of the CFD solver to record the expression tree as a tape. This tape is then rewound in the reverse mode with a novel iterative approach to evaluate the adjoints of all derived-type variables (including intermediate and ultimately design variables).

3. The adjoint solutions obtained from the UNPAC-AD solver are then returned back to the UNPAC-OPT program. Here, the gradient information is passed on to a quasi-Newton optimizer, details of which are presented in Section 7.1.2, to achieve the new set of design variables. The new solution is then returned back to the UNPAC solver to perform the next design cycle. The optimization process is repeated until either the desired number of design cycles is reached or the desired tolerance for the optimal solution has been achieved.

In general, any set of design variables can be used for the optimization process. For certain optimization problems, these variables can be the angle of attack, free-stream Mach number, etc. For the case of shape optimization, the design variables are taken to be the surface points defining the geometry. Compared to the cases where shape parameterization is utilized, the surface points can, theoretically, offer a complete design space. However, the surface mesh points as design variables can potentially lead to unsmooth surface profiles due

to high frequency modes. This problem is circumvented using a smoothing approach similar to that proposed by Huang and Ekici [108].

As for the objective function, although any desired definition can be easily implemented, three possible options are considered in this work. These are namely the:

1. Drag coefficient, C_D (minimized by default)
2. Lift coefficient, C_L (maximized by default)
3. Lift-to-drag ratio or efficiency, C_L/C_D (maximized by default)

The wrapper program for the design optimization framework uses bash scripts to automate the process of running nominal and adjoint solvers. Additionally, it runs scripts to organize solution data into different folders for each design cycle. It must be noted that the nominal solver can be run in serial or parallel mode while the adjoint solver is currently run only in serial mode. The parallelization of the adjoint solver is the subject of ongoing research and will be addressed in future works.

7.1.1 Shape Deformation

In aerodynamic shape optimization, it is common to use shape parameterization techniques where the focus is shifted from the actual grid points defining the geometry to a certain number of variables controlling the parameterized geometry. In this approach, the number design variables that can be in the order of hundreds to tens of thousands of variables will be reduced down to a fraction of that at the expense of limiting the design space.

In practice, Hicks-Henne bump functions [97], B-Spline (NURBS) [134], and Free Form Deformation (FFD) [40] techniques are commonly used for the purpose of shape parameterization. However, if not tuned correctly, these geometrical representation tools can lead to cases where the design variables would not form a complete design space, causing the optimizer getting trapped at local optima. As an alternative, the mesh points can be directly used as the design variables given the fact that the cost of the adjoint solver is independent of the number of design variables.

As discussed earlier, the use of mesh points can lead to unsmooth geometrical profiles which, at extreme conditions, can even cause convergence issues for the nominal solver. In the framework of steepest descent optimizers, Jameson [114] and Castonguay and Nadarajah [39] have utilized a smoother technique based on the Sobolev inner product to smooth the gradient information. However, for the Newton and quasi-Newton optimizers, smoothing the gradient information can cause gradient inaccuracies that can negatively affect the performance of the optimization algorithm. In this regard, Huang and Ekici [108] have proposed smoothing the surface perturbations using an implicit smoother before applying them to the design variables from the previous design cycle. In this work, a similar approach is utilized which follows the same procedure as described in Section 3.7.3 for the implicit residual smoothing. First, the perturbation of the design variable i is described as

$$\Delta x_i = x_i^{n+1} - x_i^n \tag{7.1}$$

where x^n and x^{n+1} are the design variables at two subsequent design cycles n and $n+1$, respectively. Here, the smoothed perturbation at node i is defined based on a pseudo-Laplacian of the perturbations at neighboring nodes via

$$\Delta x_i^* + \epsilon \sum_{j=1}^{\mathrm{Ngb}_i} [\Delta x_i^* - \Delta x_j] = \Delta x_i \tag{7.2}$$

where Δx_i and Δx_i^* are the original (unsmoothed) and smoothed perturbations of the design variable i, respectively. It is worth mentioning that the pseudo-Laplacian only includes neighbors of node i that lies on the surface and hence included in the set of design variables. Finally, the design variables at design cycle $n+1$ are updated via

$$\vec{x}^{n+1} = \vec{x}^n + \vec{\Delta x}^* \tag{7.3}$$

It must be noted that the smoothing parameter, ϵ, is taken to be $0.3 \leq \epsilon \leq 0.8$ with larger values of ϵ leading to more smoothing of the perturbations. In this work, a smoothing parameter of 0.5 is used. As will be shown later in this chapter, the proposed approach

can efficiently smooth the design variables that ultimately result in a smooth geometry deformation during design updates.

In addition to the smoothing process described here, an under-relaxation approach is used to limit the node movements around the trailing edge. Numerical tests performed in this work have shown that it is necessary to limit the movement of the mesh points in the vicinity of the sharp trailing edge due to the fact that strong singularities can be observed in this region. These modes around the sharp trailing edge have significantly higher frequencies than those in other parts of the airfoil and can lead to rapid node movements. In some severe cases, these movements can cause the top and bottom surfaces to cross each other. To avoid such issues an under-relaxation parameter is applied to the surface perturbations before they are used to update the design variables. As a rule of thumb, a variable under-relaxation parameter, ω, is applied to the last 10%-chord length with a value of $\omega = 1.0$ (no relaxation) at 90%-chord length up to a value of $\omega = 0.0$ (full relaxation) at the trailing edge to fix this node. Additionally, the leading edge is also fixed so that the flow incidence angle remains constant throughout the shape optimization process.

The smoothed (and under-relaxed) design variables are then passed on to the nominal solver to obtain the flow solution in the next design cycle. A radial basis function (RBF) approach is used to perform volume mesh deformation considering the displacements of the design variables as the control points of the RBF system of equations (see Section 3.8.1).

7.1.2 Optimization Algorithm

In general, most numerical optimization techniques involve an iterative process by considering a sequence of intermediate solutions for design variables, $\vec{x}^{\,n}$, that will theoretically converge to the minimizer of function $f(\vec{x})$ at an optimal solution, $\vec{x}^{\,\mathrm{opt}}$. The goal here is to use the information at $\vec{x}^{\,n}$ to find the next estimate $\vec{x}^{\,n+1}$ such that $f(\vec{x}^{\,n+1}) < f(\vec{x}^{\,n})$.

Newton and Quasi-Newton Optimization Methods

One of the most widely used gradient-based optimization algorithms is the Newton's method which uses a quadratic approximation of the objective function (assuming that the function

f is twice-differentiable). Thus, the Taylor series expansion of the objective function around the fixed point, $\vec{x}$, can be written as

$$f(\vec{x}+\Delta\vec{x}) \approx f(\vec{x}) + \Delta\vec{x}^{\,T}\nabla f(\vec{x}) + \frac{1}{2}\Delta\vec{x}^{\,T}\left(\nabla^2 f(\vec{x})\right)\Delta\vec{x}\;;\;\; \mathcal{O}(\Delta\vec{x}^{\,3}) \tag{7.4}$$

where $\nabla f(\vec{x})$ and $\nabla^2 f(\vec{x})$ are the first and second derivatives (Jacobians and Hessians) of the objective function. The goal here is to find the perturbation $\Delta\vec{x}$ such that $f(\vec{x}^{\,n+1} = \vec{x}^{\,n} + \Delta\vec{x}) < f(\vec{x}^{\,n})$. Without loss of generality, Eq. 7.4 can be written for a quadratic approximation, h^n, as a function of the perturbation, $\Delta\vec{x}$, via

$$h^n(\Delta\vec{x}) = f(\vec{x}^{\,n}) + \Delta\vec{x}^{\,T}\mathbf{g}_n + \frac{1}{2}\Delta\vec{x}^{\,T}\mathbf{H}_n\Delta\vec{x} \tag{7.5}$$

where, $\mathbf{g}_n$ and $\mathbf{H}_n$ are the gradient and Hessian of the objective function at $\vec{x}^{\,n}$, respectively. In order to find $\Delta\vec{x}$ that would minimize the local quadratic approximation, the partial derivative of $h^{\,n}(\Delta\vec{x})$ with respect to $\Delta\vec{x}$ is set to zero, i.e.,

$$\frac{\partial h^{\,n}(\Delta\vec{x})}{\partial\Delta\vec{x}} = \mathbf{g}_n + \mathbf{H}_n\Delta\vec{x} = 0 \tag{7.6}$$

knowing that any $\Delta\vec{x}$ that yields $\dfrac{\partial h^{\,n}(\Delta\vec{x})}{\partial\Delta\vec{x}} = 0$ would be a local extrema of the quadratic approximation. Assuming that the Hessian matrix is positive definite, then the solution to Eq. (7.6) will be a global minimum which can be defined as

$$\Delta\vec{x} = -\mathbf{H}_n^{-1}\mathbf{g}_n \tag{7.7}$$

Equation (7.7) describes the search direction that can move the design toward the optimal solution. In practice, however, the new iterate is calculated based on

$$\vec{x}^{\,n+1} = \vec{x}^{\,n} - \alpha\mathbf{H}_n^{-1}\mathbf{g}_n \tag{7.8}$$

where α is a step-size defined using a "line search" algorithm. A robust line search technique tries to find the optimal step-size that can rapidly and efficiently lead to the optimal solution.

As can be seen, the main issue with the Newton's method is the computational cost associated with the calculation of the inverse Hessian matrix. Therefore, quasi-Newton methods have been introduced to approximate the inverse Hessian matrix, thus eliminating the need for providing the second-derivative information to the optimizer. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [230] is a well-known quasi-Newton optimizer that addresses this issue by approximating the inverse Hessian matrix using the gradient information from previous iterations.

The original BFGS algorithm involves storing a dense $N \times N$ matrix that is used for the approximation of the inverse Hessian matrix. This requirement can lead to a significant memory footprint in large scale optimization problems. Therefore, the limited-memory BFGS (L-BFGS) [138] has been introduced where the gradient history size is limited to only M vectors ($M \ll N$), thus reducing the memory requirement significantly.

Optimizer Toolbox in the UNPAC-OPT Program

The UNPAC-OPT wrapper program developed in this work utilizes a quasi-Newton optimization algorithm for the following bound constrained minimization problem

$$\begin{aligned} &\min f(\vec{x}) \\ &\text{subject to} \quad \vec{l} \leq \vec{x} \leq \vec{u} \end{aligned} \tag{7.9}$$

where $\vec{x}$ is the vector of N design variables bounded by the lower, $\vec{l}$, and upper, $\vec{u}$, bounds and f is a differentiable scalar objective or cost function. The L-BFGS-B [31] tool written in Fortran 77 language is used as a black-box optimizer which is based on the bound constrained version of the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm [138]. Using an efficient Hessian approximation technique, this optimizer is suitable for large-scale optimization problems with limited memory footprint.

Additionally, the toolbox can be used for both unbounded as well as bound constrained problems. For bound constrained optimization, the method uses a simple gradient technique to determine free and fixed variables (according to the per-variable constant lower and upper

bounds). Ultimately, the L-BFGS method is applied to the free variables via an iterative process. An integer array, nBD(:), of size N is used that can have values between 0 and 3 for each design variable. These values determine whether either (nBD = 1 or nBD = 3) or both (nBD = 2) bounds are applied and a value of (nBD = 0) assumes an unbounded design variable. At each optimization cycle, the L-BFGS-B optimizer receives the current design variables and their bounds (if any), the gradient vector, and the value of the cost function. Upon successful termination, the optimizer returns the new values of the design variables or solutions.

7.2 Shape Optimization Results

In this section, the UNPAC-DOF framework is used for aerodynamic shape optimization of two well-known airfoils. First, the lift-to-drag ratio or the efficiency of a NACA0012 airfoil operating at inviscid subsonic flow regime is maximized. For this purpose, the unbounded as well as bound constrained optimization techniques are considered. Next, the drag of a NACA0012 airfoil operating at inviscid transonic flow regime is minimized. Finally, the turbulent flow past the National Renewable Energy Laboratory (NREL) S809 wind turbine cross-section is considered and the shape optimization is carried out with the goal of increasing the efficiency of the blade section by maximizing the lift-to-drag ratio at a certain operating condition.

7.2.1 Subsonic NACA0012 Airfoil

As the first optimization test case, the lift-to-drag maximization of the NACA0012 airfoil is sought. Here, the inviscid subsonic flow at $M = 0.5$ with an angle of attack of $\alpha = 2.0$ degrees is considered. The computational grid used for this case is the same unstructured mesh shown previously in Section 5.1. For the purpose of this optimization test case, the first-order Roe scheme is used for the discretization of the convective fluxes.

Initially, the unbounded or unconstrained efficiency maximization is considered. For this reason, the objective function is taken to be the ratio of lift coefficient, C_L, to drag coefficient, C_D, which is also known as the aerodynamic efficiency. Since the optimizer, in general, solves

a standard minimization problem, here the cost function is taken to be negative of the ratio in order to maximize its value, i.e.,

$$f(\vec{x}, \vec{U}(\vec{x})) = -\frac{C_L}{C_D} \tag{7.10}$$

where $\vec{x}$ and $\vec{U}(\vec{x})$ are the design variables and the flow solutions at the corresponding design cycle, respectively. Design variables are taken to be the y-coordinates of the grid nodes on the surface of the airfoil which result in vertical movement of these node throughout the design optimization process.

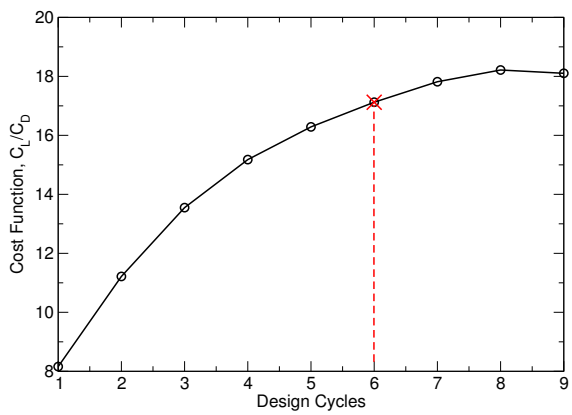
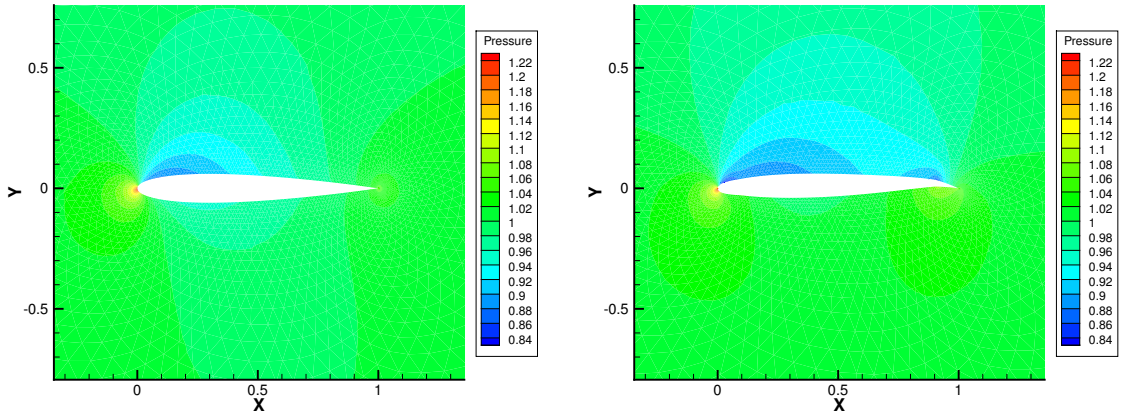


Figure 7.2: Convergence history of the lift-to-drag ratio for the unbounded efficiency optimization of NACA0012 airfoil.

As will be shown later, unbounded lift-to-drag ratio maximization can lead to extreme deformations in the airfoil geometry that can eventually cause a zero thickness at some point along the airfoil chord. As a result, for the first unbounded optimization case, the optimizer is intentionally setup in a way that will slow down the line search toward optimal solution. Convergence history of the unbounded optimization test case for the NACA0012 airfoil operating at inviscid subsonic flow regime is shown in Figure 7.2.

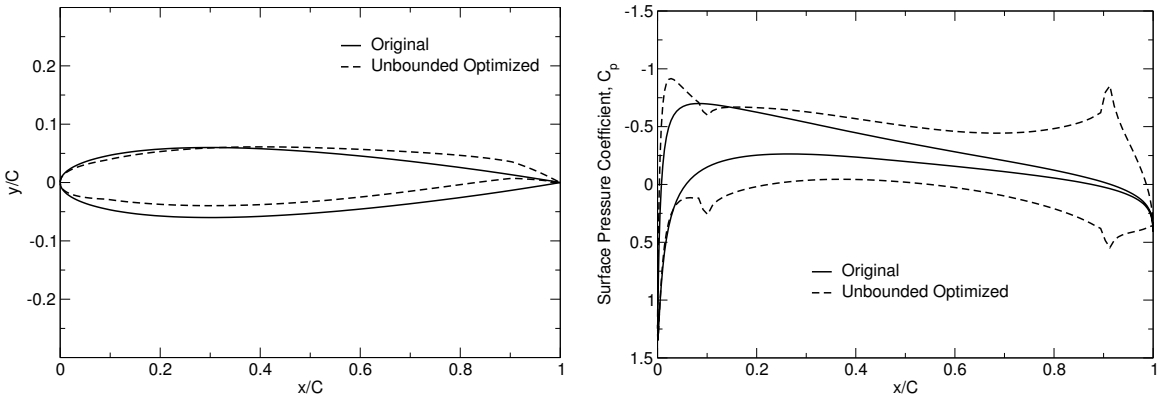
For the unbounded optimization case and after six design cycles, the airfoil becomes significantly thinner and has an apparent camber. The lift-to-drag ratio increases for almost 110% after 6 design cycles. The contour plots of pressure for the original and optimized (design cycle 6) airfoils are shown in Figure 7.3.

Additionally, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils are shown in Figure 7.4. As can be



(a) Original Design **(b)** Unbounded optimized (design cycle 6)

Figure 7.3: Contour field of pressure for the inviscid subsonic flow past NACA0012 airfoil at $M = 0.5$ and AOA $= 2.0$ deg.

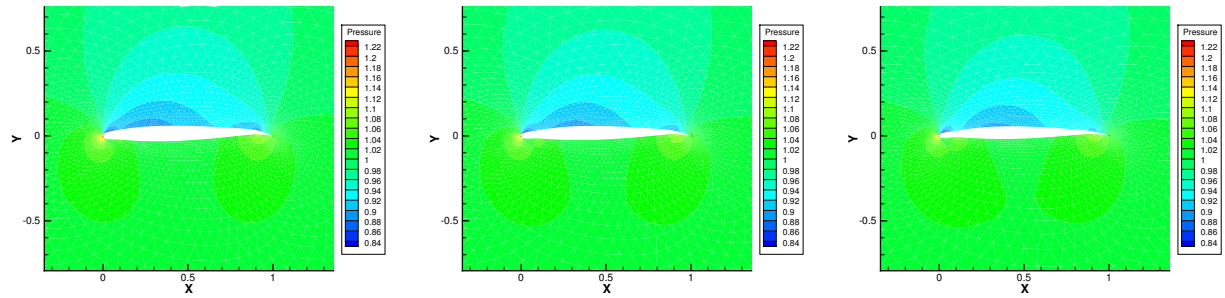


(a) Airfoil shape **(b)** Surface pressure coefficient, C_p

Figure 7.4: Comparison of airfoil shape and the surface pressure coefficients for the original and unbounded optimized geometries.

seen, there is an extreme movement close to the trailing edge region which proves the fact that under-relaxing surface perturbations in this region is necessary. Moreover, the airfoil thickness is reduced for about 18% from its original value after 6 design cycles.

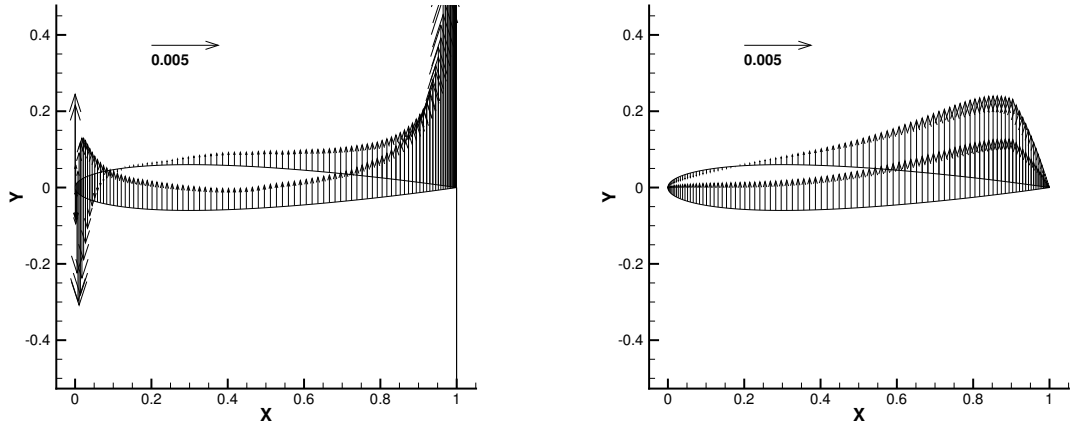
As shown earlier, for the unbounded optimization case, the sixth design cycle was chosen as the final optimized design although the value of the cost function is still increasing in the subsequent design cycles before starting to decrease at design cycle 9. The contour plots



(a) Design cycle 7 **(b)** Design cycle 8 **(c)** Design cycle 9

Figure 7.5: Contour field of pressure for the inviscid subsonic flow past NACA0012 airfoil at $M = 0.5$ and AOA $= 2.0$ deg.

of pressure for design cycles seven, eight, and nine are also shown in Figure 7.5. It can be clearly seen that the airfoil thickness decreases significantly around 10% chord length.



(a) Unsmoothed perturbations **(b)** Smoothed perturbations

Figure 7.6: Comparison of surface perturbations for the original (unsmoothed) and the smoothed cases in vector notation at the first design cycle for the NACA0012 airfoil geometry (lift-to-drag maximization case).

As discussed earlier, a necessary step to preserve a smooth geometry throughout the optimization process is to smooth the surface perturbations obtained at the end of each design cycle. The original unsmoothed perturbations are compared against the ones after smoothing and the results are presented as perturbation vectors in Figure 7.6. It can be clearly seen that the magnitude of perturbations are very large in the sharp trailing edge

region. Also, both leading and trailing edges have very sharp node movements initially. After the smoothing, these perturbations become much smoother and their magnitudes get smaller, leading to a more gradual surface deformation. Additionally, the effect of the under-relaxation approach applied to the 10% chord length region close to the trailing edge can be seen in Figure 7.6. This under-relaxation approach has proven to be necessary for bounding the node movements close to the trailing edge in order to avoid infeasible or non-physical shape deformations.

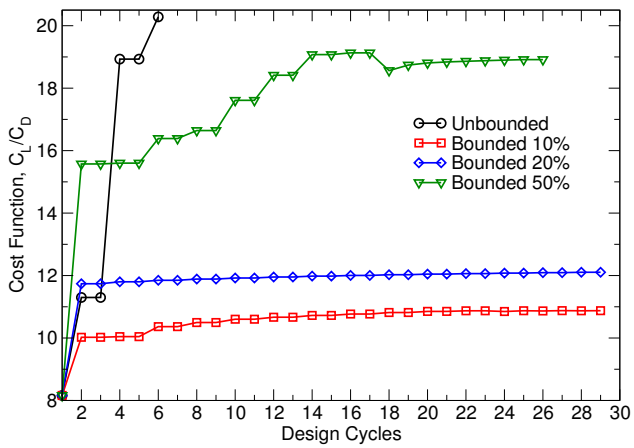
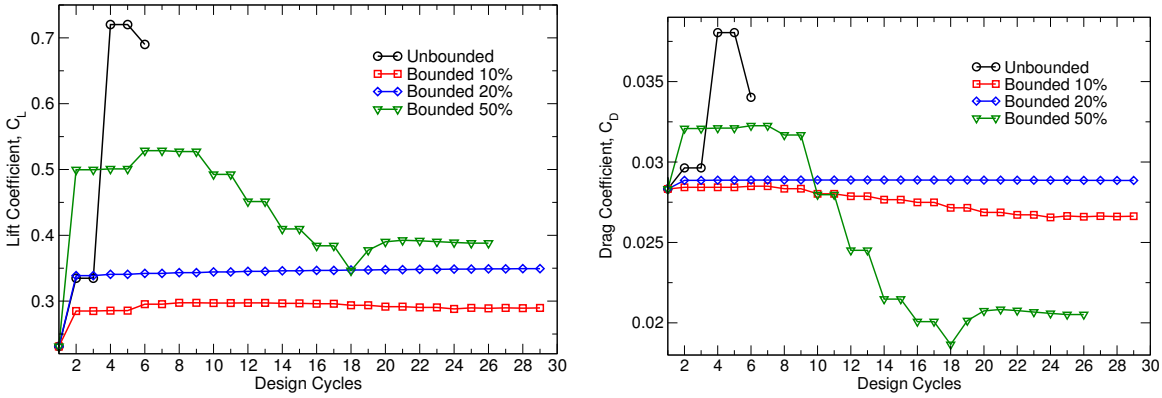


Figure 7.7: Convergence history of the lift-to-drag ratio for the bounded and unbounded efficiency optimization of NACA0012 airfoil.



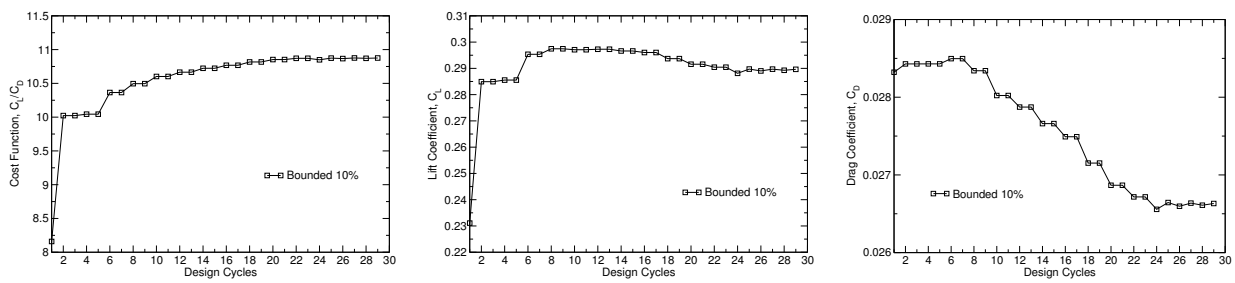
(a) Lift coefficient, C_L **(b)** Drag coefficient, C_D

Figure 7.8: Convergence histories of the lift and drag coefficients for the bounded and unbounded efficiency optimization of NACA0012 airfoil.

Next, the bound constrained optimization test cases are considered. Here, the original unbounded test case is once again included for the sake of comparison. However, unlike the

previous unbounded test case, optimal line search settings are utilized here. Three bound limits are used for the constrained optimization test cases where the y-coordinates of the grid nodes are bounded by 10%, 20%, and 50% of their original value. The convergence history of the objective function for the unbounded and the three bound constrained tests are shown in Figure 7.7.

It is apparent that the increase in the bound limit results in a more optimal geometry. However, the significant reduction in the airfoil thickness can make the unconventional design unfit for manufacturing. Additionally, while the unbounded case results in a highly optimal solution after 6 design cycles, the value of the cost function drops dramatically right after this point and the solution becomes infeasible. The convergence histories for the lift and drag coefficients are also presented in Figure 7.8.

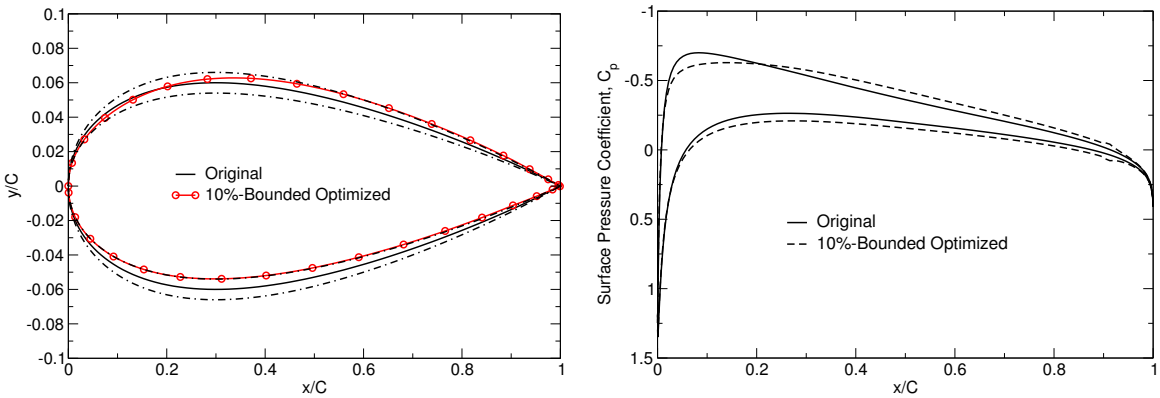


(a) Objective, C_L/C_D **(b)** Lift coefficient, C_L **(c)** Drag coefficient, C_D

Figure 7.9: Convergence histories of the objective function, lift, and drag coefficients for the 10% bound constrained efficiency optimization of NACA0012 airfoil.

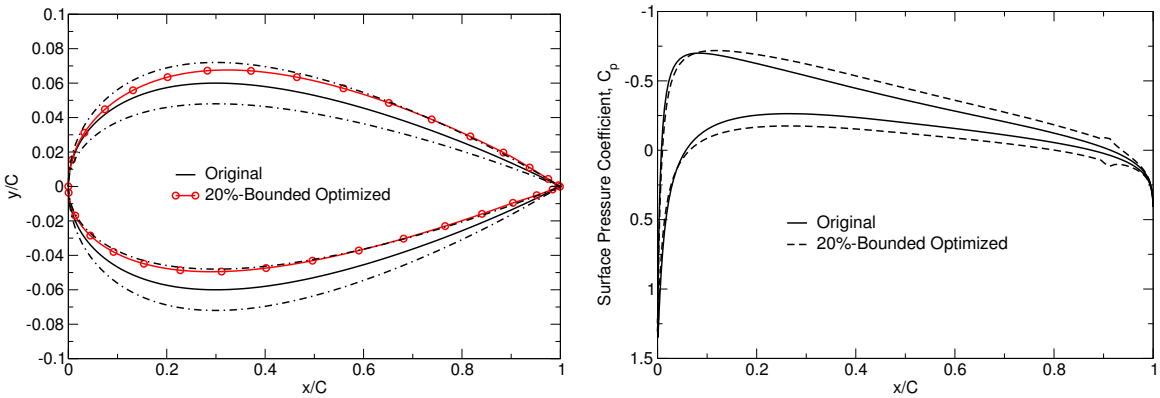
While the objective for these optimization test cases was to maximize the lift-to-drag ratio, it can be seen in Figure 7.8 that the 10% bounded case results in an increase in the lift coefficient and a decrease in the drag coefficient which is exactly what is desired in aerodynamic shape optimization. The convergence histories for the lift-to-drag ratio, lift coefficient, and drag coefficient for this bounded case are plotted separately from the other optimization test cases and are shown in Figure 7.9.

Next, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils are shown in Figures 7.10 through 7.12 for the three bound constrained test cases studied here. As intended, in all three cases the final airfoil shape is bounded by the upper and lower bounds specified for the optimizer. Additionally,



(a) Airfoil shape **(b)** Surface pressure coefficient, C_p

Figure 7.10: Comparison of airfoil shape and the surface pressure coefficients for the original and 10%-bounded optimized geometries (lower and upper bounds are shown in dashed lines).

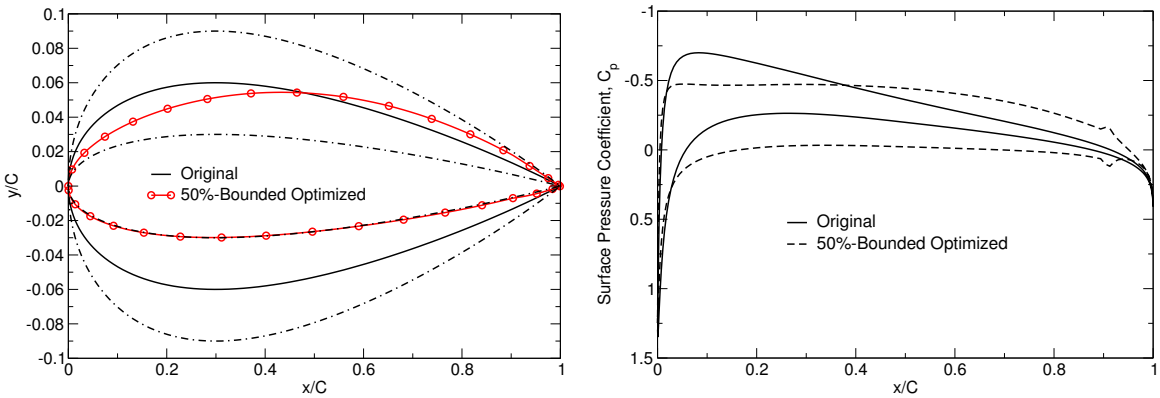


(a) Airfoil shape **(b)** Surface pressure coefficient, C_p

Figure 7.11: Comparison of airfoil shape and the surface pressure coefficients for the original and 20%-bounded optimized geometries (lower and upper bounds are shown in dashed lines).

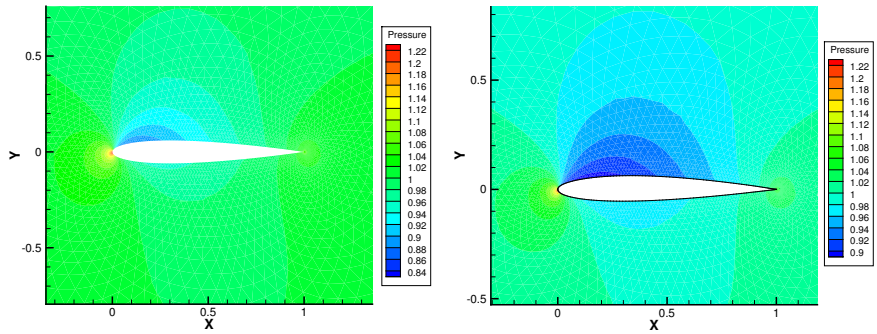
the movements in the trailing edge region are much larger in the case of 50%-bound since the node movements are much less limited for this case compared to the other two bounded test cases.

Additionally, the contour plots of pressure for the original, 10% bounded, 20% bounded, and 50% bounded optimized airfoils are shown in Figure 7.13. Once again, it is noted that after each design update, the existing computational mesh is deformed based on the topological changes on the surface. This approach eliminates the need to generate a new mesh after each design cycle. Similar to what is used in the r-adaptive mesh relocation technique

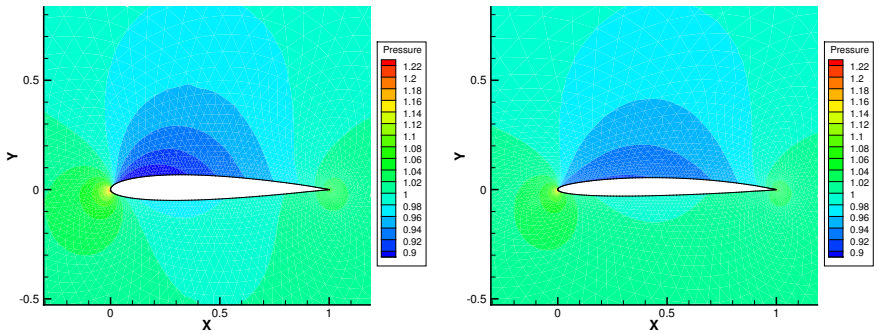


(a) Airfoil shape **(b)** Surface pressure coefficient, C_p

Figure 7.12: Comparison of airfoil shape and the surface pressure coefficients for the original and 50%-bounded optimized geometries (lower and upper bounds are shown in dashed lines).



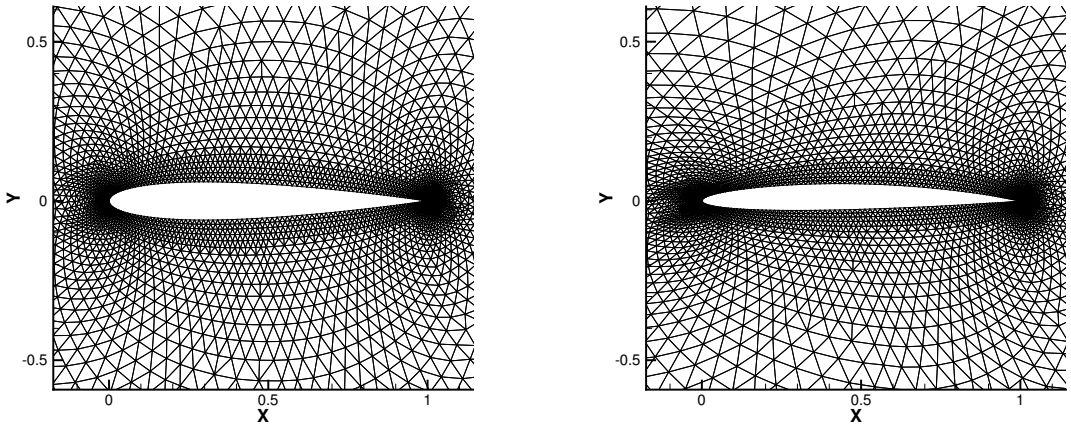
(a) Original design **(b)** 10%-bounded optimized



(c) 20%-bounded optimized **(d)** 50%-bounded optimized

Figure 7.13: Contour field of pressure for the inviscid subsonic flow past NACA0012 airfoil at $M = 0.5$ and AOA $= 2.0$ deg.

developed in this work, the mesh deformation for design optimization is performed using the RBF approach with a support radius that can efficiently relocate grid nodes to conform the deformed geometry of the airfoil. As an example, the original and deformed grids for the 50%-bounded case are shown in Figure 7.14. Since no re-meshing is required in this approach, the quality of the original mesh is preserved. Additionally, for the turbulent test cases, the initial $y+$ value is retained throughout the design cycles which is necessary for accurate turbulent solutions. As shown earlier in this work, the present RBF-based mesh deformation approach is capable of efficiently relocating grid points without leading to any inverted cells.



(a) Original geometry grid

(b) Deformed geometry grid

Figure 7.14: Original and deformed grids obtained using the RBF technique for the NACA0012 airfoil in the 50%-bounded optimization test case.

Finally, it must be mentioned that the memory footprint of the adjoint solver for this test case is about 200 MBytes and the computational cost of running the adjoint solver in each design cycle is about 1.5 times that of the nominal flow solver for the same number of iterations.

7.2.2 Optimization of the NACA0012 Airfoil in the Transonic Regime

The next optimization test case considered here is the drag minimization of a NACA0012 airfoil subject to an inviscid transonic flow. This case, which was extensively studied in earlier parts of this dissertation, has a free-stream Mach number of 0.8 and an angel of attack of 1.25 degrees. The same unstructured grid and solver settings used earlier are adopted here. The cost or the objective function is taken to be the drag coefficient, C_D, which is sought to be minimized and the design variables are also taken to be the y-coordinates of the grid points on the surface of the airfoil. Additionally, only the unconstrained optimization is considered in this section.

First, the history of the objective function (C_D) is plotted for different design cycles in Figure 7.15. As can be seen, there is a steady drop in the drag coefficients with a 95% drop in the drag force in the first 5 design cycles. Here, the optimization process is continued for 20 design cycles although most of the reduction in the drag coefficient has been achieved within the first 5 cycles.

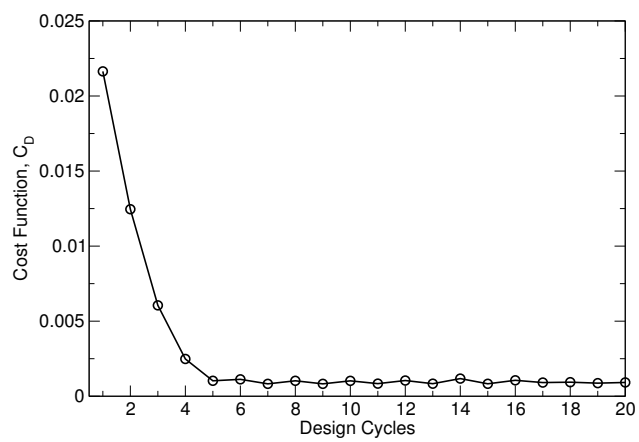
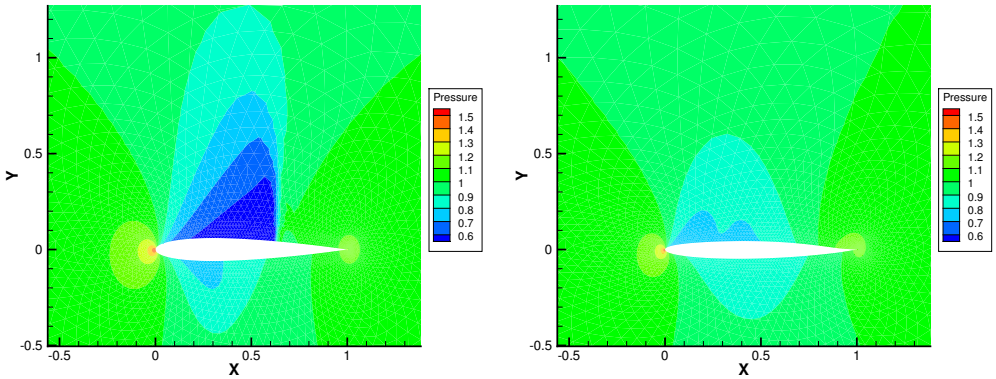


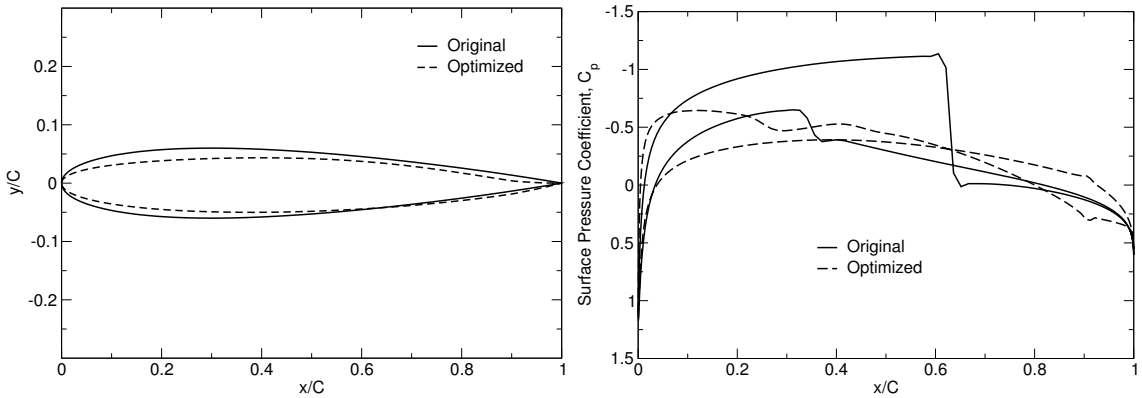
Figure 7.15: Convergence history of the objective function for the unbounded drag minimization of NACA0012 airfoil.

The contour plots of pressure for the original and the optimized airfoils are shown in Figure 7.16. Also, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils are shown in Figure 7.17.



(a) Original Design **(b)** Unbounded optimized

Figure 7.16: Contour field of pressure for the inviscid transonic flow past NACA0012 airfoil at $M = 0.8$ and AOA $= 1.25$ deg.



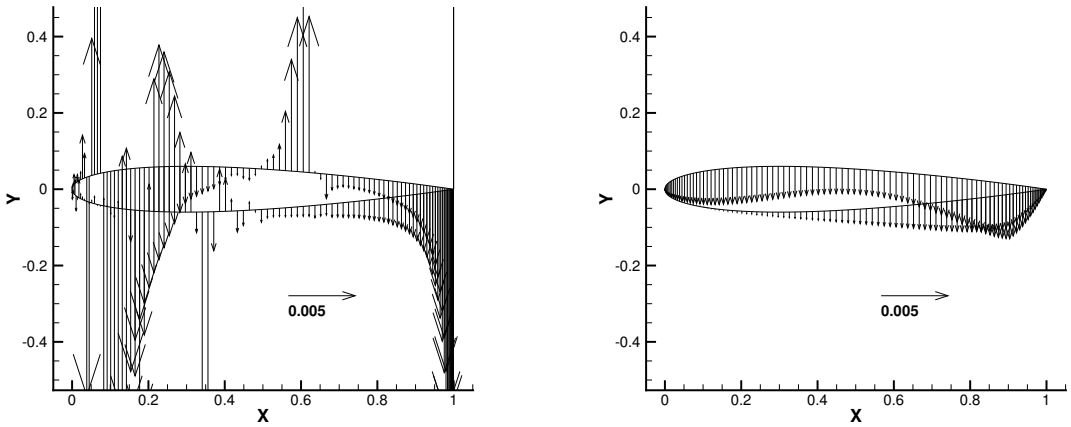
(a) Airfoil shape **(b)** Surface pressure coefficient, C_p

Figure 7.17: Comparison of airfoil shape and the surface pressure coefficients for the original and optimized geometries.

As shown earlier in this work, the inviscid transonic flow past the NACA0012 airfoil leads to the formation of a strong shock on the suction side and a weaker shock on the pressure side. As can be seen in Figures 7.16 and 7.17, the drag minimization leads to the elimination of these shocks on both sides of the airfoil.

Due to the fact that the mesh nodes are used as the design variables, it is again necessary to perform a smoothing process before applying the surface deformations. Therefore, the original unsmoothed perturbations are compared against the ones after smoothing iterations and the results are shown in terms of perturbation vectors in Figure 7.18. As can be seen, the

unsmoothed perturbations have high frequency modes that can lead to unsmooth or jagged surface deformation. With the use of the smoothing procedure, these surface perturbations are smoothed significantly while the overall geometry deformation features are retained. Additionally, an under-relaxation is applied to the 10% chord length region close to the trailing edge, and its effects can be clearly seen in Figure 7.18. Again, this under-relaxation approach has proven to be necessary for bounding the node movements close to the sharp trailing edge in order to avoid non-physical shape deformations.



(a) Unsmoothed perturbations **(b)** Smoothed perturbations

Figure 7.18: Comparison of surface perturbations for the original (unsmoothed) and the smoothed cases in vector notation at the first design cycle for the NACA0012 airfoil geometry (drag minimization case).

Once again, it must be noted that the memory footprint of the adjoint solver for this case is about 800 MBytes and the computational cost of running the adjoint solver in each design cycle is about 2.5x that of the nominal flow solver for the same number of iterations.

7.2.3 NREL S809 Wind Turbine Blade Section

Finally, the efficiency maximization of a wind turbine blade section is considered. The National Renewable Energy Laboratory (NREL) Phase VI wind turbine has been widely used as a test bed for CFD analysis as well as design optimization [186]. This particular horizontal-axis wind turbine (HAWT) is made of two tapered-twisted blades with an S809

airfoil cross-section. This airfoil has been developed by Airfoils, Inc. [190] and has been optimized with the goal of maximizing wind energy power production. As a result of these optimization studies, the NREL Phase VI wind turbine blade has been designed with a linear taper and a nonlinear twist distribution along the span while using the S809 airfoil exclusively from the root all the way to the tip of the blade.

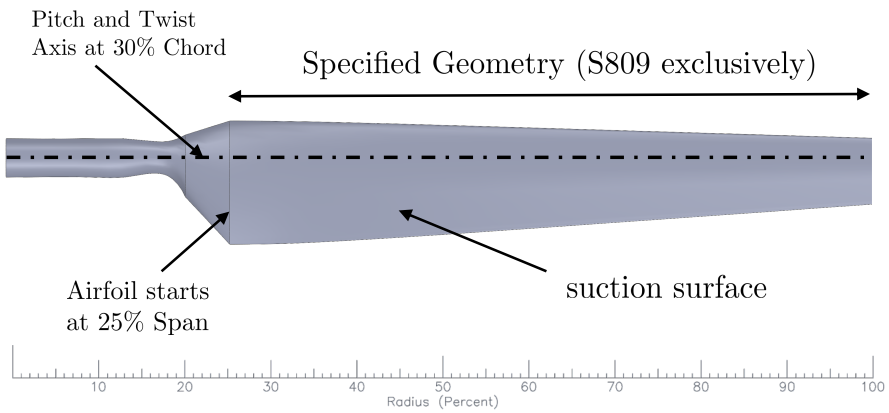


Figure 7.19: Geometry of the NREL Phase VI horizontal-axis wind turbine (HAWT) blade with S809 airfoil as blade cross-section.

The geometry of the NREL Phase VI wind turbine blade is shown in Figures 7.19 and 7.20. As can be seen, 75% of the span length is covered with the standard S809 airfoils while 15% of the span length has a semi-S809 cross-section in an area that is formed by lofting an S809 airfoil into a circular section.

The standard S809 airfoil has a 21% maximum thickness at 39.5% chord length and is designed as a laminar flow airfoil. Previously and in Section 5.3, UNPAC solver was used to study the turbulent and transitional flows past the S809 airfoil at two angles of attack of 4.1 and 12.2 degrees. In this section, the goal is to maximize the efficiency or the lift-to-drag ratio of the S809 airfoil operating at the same conditions while only considering the 4.1 degree angle of attack case. Here, the Mach number is 0.1 and the Reynolds number is 1.0 million. The computational grid used for this with a $y+$ value of less than 0.5 is shown in Figure 7.21.

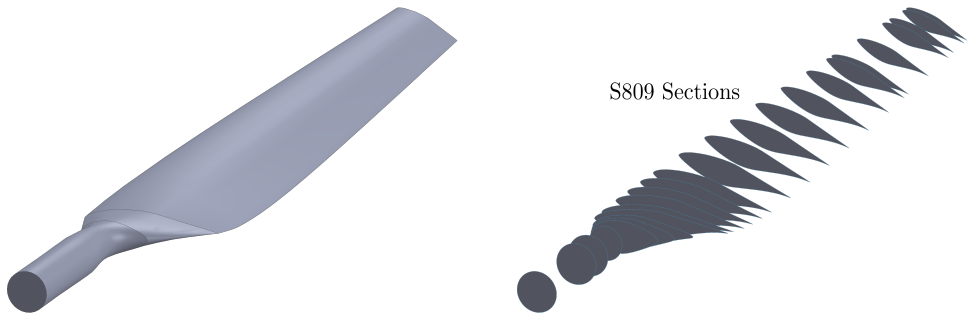
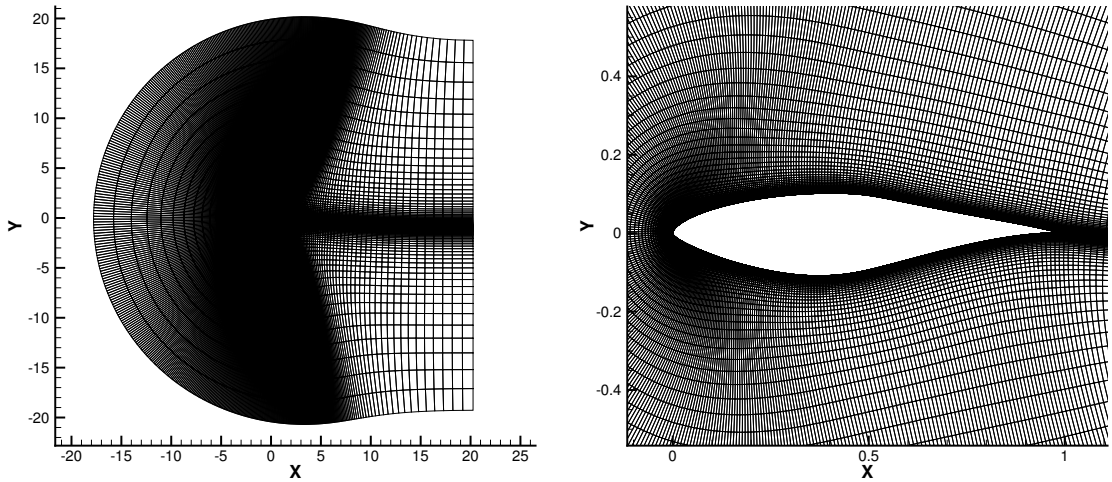


Figure 7.20: S809 cross-sections of the NREL Phase VI horizontal-axis wind turbine (HAWT) blade.



(a) far-field grid **(b)** near-field grid

Figure 7.21: Computational grid used for the lift-to-drag maximization test case for the S809 airfoil.

Here, an unconstrained optimization is considered and the objective function is once again taken to be $f = -C_L/C_D$ in order to maximize the lift-to-drag ratio. The optimization process is continued for 10 design cycles and the convergence histories of the objective function as well as the lift and drag coefficients are shown in Figures 7.22 and 7.23.

As can be seen in Figure 7.22, there is a steady increase in the lift-to-drag ratio with a maximum of 6.2% increase the efficiency of the S809 airfoil. Similarly, the lift coefficient is increased 2.7% whereas the drag coefficient is decreased about 4.6% compared to the original values.

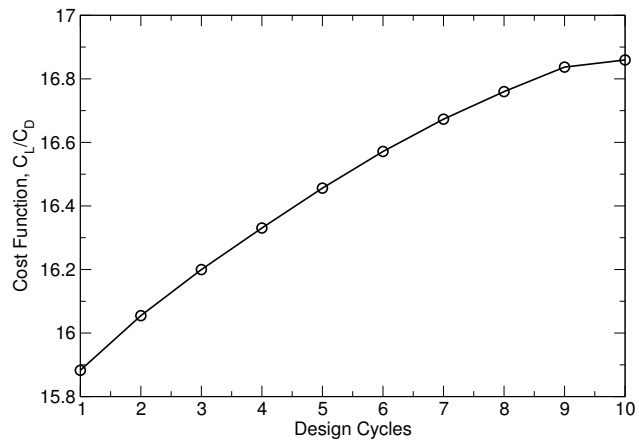
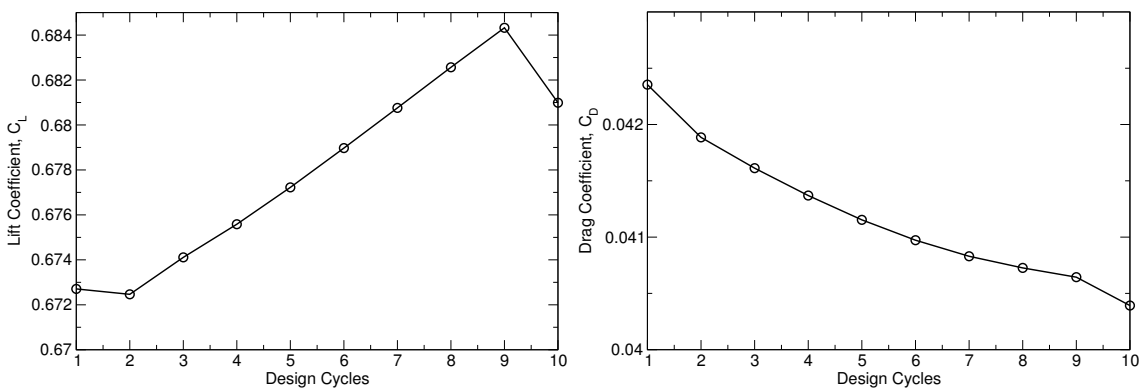


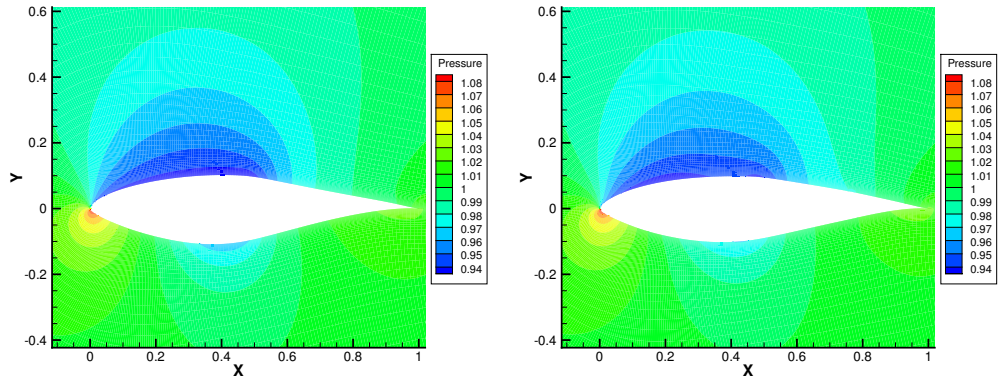
Figure 7.22: Convergence history of the lift-to-drag ratio for the unbounded efficiency maximization of the S809 airfoil.



(a) Lift coefficient, C_L **(b)** Drag coefficient, C_D

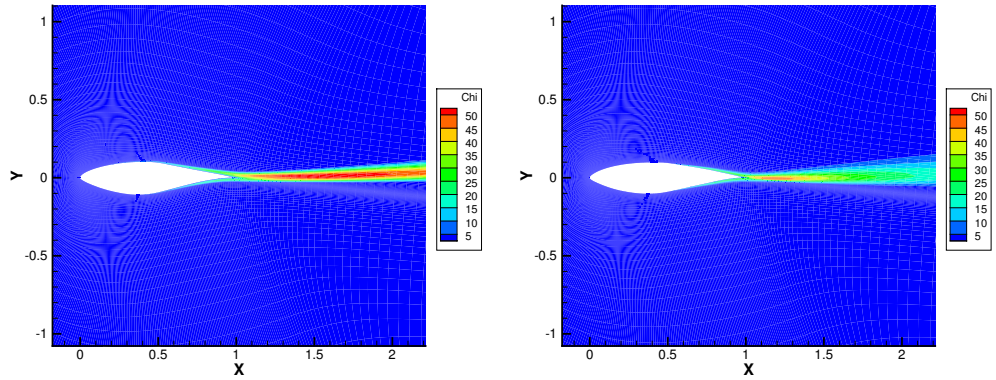
Figure 7.23: Convergence histories of the lift and drag coefficients for the unbounded efficiency maximization of the S809 airfoil.

The contour plots of pressure and non-dimensional eddy viscosity for the original and optimized airfoils are presented next in Figures 7.24 and 7.25. It is seen that the aerodynamic shape optimization leads to a reduced turbulent intensity. As shown earlier in Section 5.3, the flow past the S809 airfoil at an angle of attack of 4.1 degree remains mostly laminar and the fully-turbulent RANS solutions lead to an over-prediction of the turbulent boundary layer. For this case and with the same fully-turbulent flow assumption, the decrease in the turbulence intensity can be associated with the delay in laminar-turbulent transition. This is mainly due to the fact that the optimized airfoil has a slightly thicker profile aft of the maximum thickness location (39.5% chord-length) which results in a reduced adverse pressure gradient around this region.



(a) Original Design **(b)** Unbounded optimized

Figure 7.24: Contour field of pressure for the turbulent flow past S809 airfoil at $M = 0.1$, $\mathrm{Re} = 10^6$, and AOA = 4.1 deg.

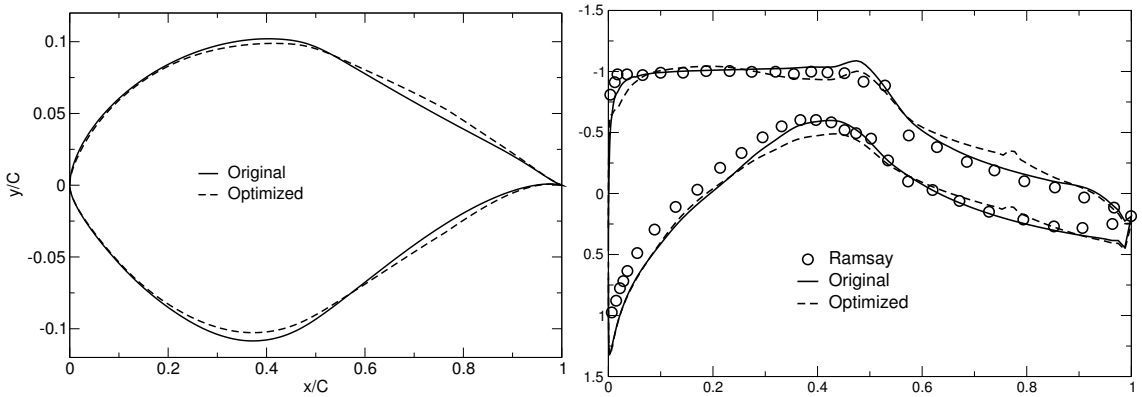


(a) Original Design **(b)** Unbounded optimized

Figure 7.25: Contour field of non-dimensional eddy viscosity for the turbulent flow past S809 airfoil at $M = 0.1$, $\mathrm{Re} = 10^6$, and AOA = 4.1 deg.

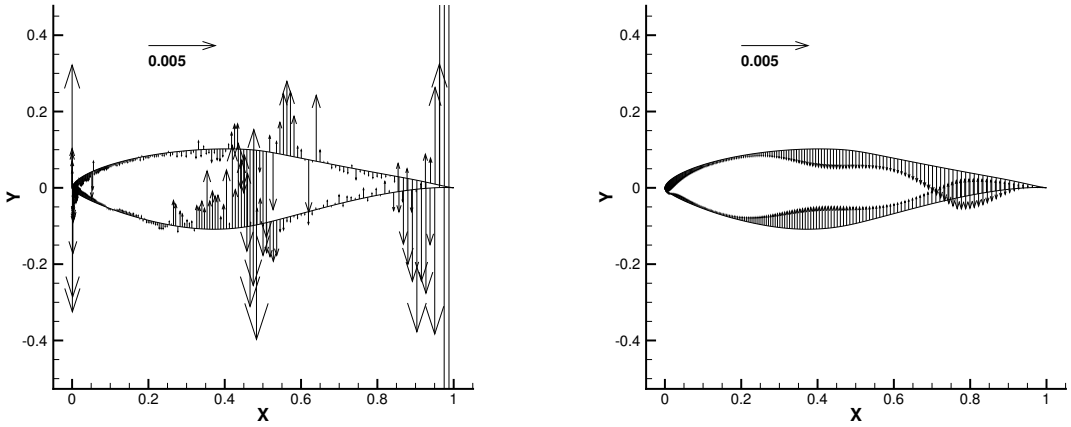
Also, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized S809 airfoils are shown in Figure 7.26. Here, the surface pressure distributions are also compared to the experimental data of Ramsay [171]. For this case, there is a 4.4% reduction in the maximum thickness after 10 design cycles. Although an unconstrained optimization was used here, the reduction in the airfoil thickness is well within the desired bound limits.

Similar to the previous optimization test cases, the effects of surface perturbation smoothing has been analyzed next. As discussed earlier, it is necessary to perform a



(a) Airfoil shape **(b)** Surface pressure coefficient, C_p

Figure 7.26: Comparison of S809 airfoil shape and the surface pressure coefficients for the original and optimized geometries.



(a) Unsmoothed perturbations **(b)** Smoothed perturbations

Figure 7.27: Comparison of surface perturbations for the original (unsmoothed) and the smoothed cases in vector notation at the first design cycle for the S809 airfoil geometry (lift-to-drag maximization case).

smoothing process before applying the surface deformations in order to preserve a smooth and non-jagged geometry. Here, the original unsmoothed perturbations are once again compared to the ones after smoothing iterations and the results are shown in terms of perturbation vectors in Figure 7.27. Although the magnitudes of the surface perturbations are not large for this case (compared to the previous test cases), there are high frequency modes involved which can lead to an unsmooth geometry. With the addition of the smoothing procedure, these surface perturbations are smoothed significantly while the overall geometry deformation

features are retained. Additionally, the under-relaxation approach applied to the 10% chord length region close to the trailing edge has been at play and its effects can be clearly seen in Figure 7.27. The under-relaxation approach used in this work has proven to be necessary for bounding the node movements close to the sharp trailing edge while maintaining a smooth shape deformation near the trailing edge.

Finally, it must be noted that the memory footprint of the adjoint solver for this case is about 1.4 GBytes and the computational cost of running the adjoint solver in each design cycle is about 2.8 times that of the nominal flow solver for the same number of iterations.

Chapter 8

Conclusions

8.1 Summary

A grid-transparent unstructured two- and three-dimensional compressible RANS solver (named UNPAC) was developed in this dissertation. This solver is also enhanced with an algebraic transition model that has proven to offer accurate flow separation and reattachment predictions for the transitional flows. The UNPAC solver uses an explicit time-marching scheme to obtain steady-state solutions. For the unsteady time-periodic flows, a harmonic balance method was incorporated that couples the sub-time level solutions over a single period via a pseudo-spectral operator. The convergence to steady-state solution was accelerated using a novel reduced-order-model (ROM) approach that has shown to offer significant reductions in the number of iterations for the explicit solver. An unstructured grid approach is adapted for both steady and HB problems using an r-adaptive mesh redistribution (AMR) that can efficiently cluster nodes around regions of large flow gradients. A novel toolbox for sensitivity analysis based on discrete adjoint method was also developed as part of this research effort. Unlike currently available operator-overloading-based differentiation tools, the **F**ast automatic **D**ifferentiation using **O**perator-overloading **T**echnique (FDOT) uses a memory-efficient iterative approach to evaluate the sensitivities of the cost function with respect to the entire design space and requires only minimal modifications to the available solver. Ultimately, the UNPAC solver and the FDOT toolbox are coupled together, and with the addition of a wrapper program, a design optimization

framework, called UNPAC-DOF, has been developed. This framework is used for performing aerodynamic shape optimization for several inviscid and turbulent flow cases past NACA0012 and the S809 wind turbine blade cross-section. Additionally, the unbounded and bound constrained optimization algorithms have been implemented and employed for efficiency maximization of a standard NACA0012 airfoil.

8.2 Future Work

The present work has utilized the UNPAC-DOF program to perform aerodynamic shape optimization. While numerical results have proven the computational and memory-wise efficiency of the FDOT toolbox, there is still room for further improvement of this adjoint sensitivity analysis tool. Additionally, the design optimization framework can be extended to more complex and three-dimensional test cases in the future. The following are some suggestions for the extension of the present work to more computationally demanding aerodynamic shape optimization problems:

1. Application of the design optimization framework to maximize efficiency of a full-body three-dimensional wind turbine blade.

2. Improving the computational efficiency of the FDOT toolbox in handling passive variables and semi-passive binary operations.

3. Further improving the memory efficiency of the FDOT toolbox by eliminating unnecessary adjoint entries in order to reduce the size of the recorded tape.

4. Parallelization the FDOT toolbox by employing a special domain decomposition technique and MPI tools to further improve the efficiency of the adjoint solver.

Bibliography

[1] Acikgoz, N. and Bottasso, C. L. (2007). A unified approach to the deformation of simplicial and non-simplicial meshes in two and three dimensions with guaranteed validity. *Computers & Structures*, 85(11):944–954. 14

[2] Aftosmis, M., Gaitonde, D., and Tavares, T. S. (1995). Behavior of linear reconstruction techniques on unstructured meshes. *AIAA Journal*, 33(11):2038–2049. 68

[3] Agarwal, R. and Deese, J. (1987). Euler calculations for flowfield of a helicopter rotor in hover. *Journal of Aircraft*, 24(4):231–238. 6

[4] Albring, T., Sagebaum, M., and Gauger, N. R. (2015). Development of a consistent discrete adjoint solver in an evolving aerodynamic design framework. AIAA Paper 2015-3240. 11

[5] Allmaras, S. R., Johnson, F. T., and Spalart, P. R. (2012). Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, pages 1–11. 42

[6] Anderson, W. K. and Venkatakrishnan, V. (1999). Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(4):443–480. 9

[7] Arieli, R. and Tauber, M. (1979). Computation of subsonic and transonic flow about lifting rotor blades. AIAA Paper 1979-1667. 6

[8] Avila, L. S., Barre, S., Blue, R., Geveci, B., Henderson, A., Hoffman, W. A., King, B., Law, C. C., Martin, K. M., and Schroeder, W. J. (2010). *The VTK User's Guide*. Kitware New York. 57

[9] Baldwin, B. S. and Barth, T. (1991). A one-equation turbulence transport model for high Reynolds number wall-bounded flows. AIAA Paper 91-0610. 167

[10] Baldwin, B. S. and Lomax, H. (1978). Thin-layer approximation and algebraic model for separated turbulent flows. AIAA Paper 1978-257. 40

[11] Barth, T. (1993). Recent developments in high order K-exact reconstruction on unstructured meshes. AIAA Paper 1993-668. 68

[12] Barth, T. and Frederickson, P. (1990). Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. AIAA Paper 1990-13. 68

[13] Barth, T. J. and Jespersen, D. C. (1989). The design and application of upwind schemes on unstructured meshes. AIAA Paper 1989-0366. 67

[14] Bas, O., Cakmakcioglu, S. C., and Kaynak, U. (2013). A novel intermittency distribution based transition model for low-Re number airfoils. AIAA Paper 2013-2531. 44, 45

[15] Batina, J. (1990). Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388. 13

[16] Baur, W. and Strassen, V. (1983). The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330. 120

[17] Bell, B. M. (2012). CppAD: A package for C++ algorithmic differentiation. *Computational Infrastructure for Operations Research*, 57. 11

[18] Benson, R. A. and McRae, D. (1991). A solution-adaptive mesh algorithm for dynamic/static refinement of two and three dimensional grids. International Conference on Numerical Grid Generation in CFD A92-47035. 14

[19] Berkman, M. E., Sankar, L. N., Berezin, C. R., and Torok, M. S. (1997). Navier–Stokes/full potential/free-wake method for rotor flows. *Journal of Aircraft*, 34(5):635–640. 6

[20] Bischof, C., Carle, A., Corliss, G., Griewank, A., and Hovland, P. (1992). ADIFOR–generating derivative codes from Fortran programs. *Scientific Programming*, 1(1):11–29. 11

[21] Blazek, J. (2015). *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, Oxford, UK. 21, 25, 26, 30, 32, 36, 37, 38, 39, 41, 55, 63, 64, 66, 69, 70, 71, 72, 77, 79, 80, 84, 86, 90, 94, 107, 109

[22] Blom, F. J. (2000). Considerations on the spring analogy. *International Journal for Numerical Methods in Fluids*, 32(6):647–668. 13, 14

[23] Bono, G. and Awruch, A. (2005). A mesh adaption method by node re-allocation using an edge-based error measure. *Revista de Engenharia Térmica*, 4(2). 14

[24] Borland, C., Rizzetta, D., and Yoshihar, H. (1982). Numerical solution of three-dimensional unsteady transonic flow over swept wings. *AIAA Journal*, 20(3):340–347. 6

[25] Bottasso, C. L., Detomi, D., and Serra, R. (2005). The ball-vertex method: A new simple spring analogy method for unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4244–4264. 14

[26] Brackbill, J. U. and Saltzman, J. S. (1982). Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics*, 46(3):342–368. 14

[27] Browne, P., Budd, C. J., Piccolo, C., and Cullen, M. (2014). Fast three dimensional r-adaptive mesh redistribution. *Journal of Computational Physics*, 275:174–196. 12

[28] Budd, C. J., Huang, W., and Russell, R. D. (2009). Adaptivity with moving grids. *Acta Numerica*, 18:111–241. 13

[29] Burg, C. O. (2004). A robust unstructured grid movement strategy using three-dimensional torsional springs. AIAA Paper 2004-2529. 13

[30] Butterfield, C., Musial, W., and Simms, D. (1992). Combined Experiment Phase I: Final report. Technical report, National Renewable Energy Lab., Golden, CO (United States). 8

[31] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208. 210

[32] Cabay, S. and Jackson, L. (1976). A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 13(5):734–752. 16

[33] Cakmakcioglu, S. C., Bas, O., and Kaynak, U. (2017a). A correlation-based algebraic transition model. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*. 44, 46

[34] Cakmakcioglu, S. C., Kaynak, U., and Bas, O. (2017b). A zero-equation transition model depending on local flow variables. In *9th Ankara International Aerospace Conference (AIAC-2017-205)*. xii, 44, 45, 46

[35] Campobasso, M. S. and Baba-Ahmadi, M. H. (2012). Analysis of unsteady flows past horizontal axis wind turbine airfoils based on harmonic balance compressible Navier–Stokes equations with low-speed preconditioning. *Journal of Turbomachinery*, 134(6):061020. 8

[36] Campobasso, M. S. and Giles, M. B. (2004). Stabilization of a linear flow solver for turbomachinery aeroelasticity using recursive projection method. *AIAA Journal*, 42(9):1765–1774. 17

[37] Caradonna, F., Tung, C., and Desopper, A. (1984). Finite difference modeling of rotor flows including wake effects. *Journal of the American Helicopter Society*, 29(2):26–33. 6

[38] Caradonna, F. X. and Tung, C. (1981). Experimental and analytical studies of a model helicopter rotor in hover. *NASA/TM 81232, NASA Ames Research Center, Moffett Field, CA*. 178, 182

[39] Castonguay, P. and Nadarajah, S. (2007). Effect of shape parameterization on aerodynamic shape optimization. AIAA Paper 2007-59. 207

[40] Chauhan, D., Chandrashekarappa, P., and Duvigneau, R. (2010). Wing shape optimization using FFD and twist parameterization. In *12th Aerospace Society of India CFD Symposium*. 206

[41] Chen, X. (2014). *Optimization of Wind Turbine Airfoils/Blades and Wind Farm Layouts*. PhD thesis, Washington University in St. Louis. 9

[42] Clark, W. S. and Hall, K. C. (2000). A time-linearized Navier–Stokes analysis of stall flutter. *Journal of Turbomachinery*, 122(3):467–476. 6

[43] Cook, P., McDonald, M., and Firmin, M. (1977). Aerofoil RAE 2822: Pressure distribution and boundary layer and wake measurements. AGARD AR 138. Technical report, A6-1-A6-77. xvi, 144, 145, 146

[44] Da Ronch, A., McCracken, A. J., Badcock, K. J., Widhalm, M., and Campobasso, M. (2013). Linear frequency domain and harmonic balance predictions of dynamic derivatives. *Journal of Aircraft*, 50(3):694–707. 154

[45] Dacles-Mariani, J., Kwak, D., and Zilliac, G. (1999). On numerical errors and turbulence modeling in tip vortex flow prediction. *International Journal for Numerical Methods in Fluids*, 30(1):65–82. 40

[46] Dacles-Mariani, J., Zilliac, G. G., Chow, J. S., and Bradshaw, P. (1995). Numerical/experimental study of a wingtip vortex in the near field. *AIAA Journal*, 33(9):1561–1568. 40

[47] Dagan, A. (2001). A convergence accelerator of a linear system of equations based upon the power method. *International Journal for Numerical Methods in Fluids*, 35(6):721–741. 16

[48] De Boer, A., Van der Schoot, M., and Bijl, H. (2007). Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85(11):784–795. 81, 82

[49] de Boor, C. (1973). Good approximation by splines with variable knots. In *Spline Functions and Approximation Theory*, pages 57–72. Springer-Verlag New York Inc. 13

[50] Degand, C. and Farhat, C. (2002). A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers & Structures*, 80(3):305–316. 13, 104

[51] Djeddi, R. and Ekici, K. (2016). Resolution of Gibbs phenomenon using a modified pseudo-spectral operator in harmonic balance CFD solvers. *International Journal of Computational Fluid Dynamics*, 30(7-10):495–515. 15

[52] Djeddi, R. and Ekici, K. (2018). An adaptive mesh redistribution approach for time-spectral/harmonic-balance flow solvers. AIAA Paper 2018-3245. 104

[53] Djeddi, R., Howison, J., and Ekici, K. (2016). A fully coupled turbulent low-speed preconditioner for harmonic balance applications. *Aerospace Science and Technology*, 53:22–37. 15, 199

[54] Djeddi, R., Kaminsky, A., and Ekici, K. (2017). Convergence acceleration of fluid dynamics solvers using a reduced-order model. *AIAA Journal*, pages 1–13. 96, 103, 144, 199

[55] Donea, J., Huerta, A., Ponthot, J.-P., and Rodríguez-Ferran, A. (2017). *Arbitrary Lagrangian–Eulerian Methods*. Wiley Online Library. 34

[56] Dowell, E. H., Curtiss, H. C., Scanlan, R. H., and Sisto, F. (1989). Unsteady transonic aerodynamics and aeroelasticity. In *A Modern Course in Aeroelasticity*, pages 443–501. Springer-Verlag New York Inc. 162

[57] Duque, E. P., Van Dam, C., and Hughes, S. C. (1999). Navier–Stokes simulations of the NREL combined experiment Phase II rotor. AIAA Paper 1999-0037. 6

[58] Economon, T., Palacios, F., and Alonso, J. (2012). Optimal shape design for open rotor blades. AIAA Paper 2012-3018. 182

[59] Ekici, K. and Hall, K. C. (2006). Fast estimation of unsteady flows in turbomachinery at multiple interblade phase angles. *AIAA Journal*, 44(9):2136–2142. 17, 99

[60] Ekici, K. and Hall, K. C. (2008). Nonlinear frequency-domain analysis of unsteady flows in turbomachinery with multiple excitation frequencies. *AIAA Journal*, 46(8):1912–1920. 7

[61] Ekici, K., Hall, K. C., and Dowell, E. H. (2008). Computationally fast harmonic balance methods for unsteady aerodynamic predictions of helicopter rotors. *Journal of Computational Physics*, 227(12):6206–6225. 175, 177, 182

[62] Ekici, K., Hall, K. C., Huang, H., and Thomas, J. P. (2013). Stabilization of explicit flow solvers using a Proper-Orthogonal-Decomposition technique. *AIAA Journal*, 51(5):1095–1104. 16, 17, 101

[63] Ekici, K. and Huang, H. (2012). An assessment of frequency-domain and time-domain techniques for turbomachinery aeromechanics. AIAA Paper 2012-3126. 15

[64] Elliott, J. and Peraire, J. (1997). Practical three-dimensional aerodynamic design and optimization using unstructured meshes. *AIAA Journal*, 35(9):1479–1485. 9

[65] Eppler, R. (2012). *Airfoil design and data*. Springer-Verlag New York Inc. 8

[66] Eyi, S. (2015). Convergence acceleration using convergence error estimation. AIAA Paper 2015-2751. 16

[67] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M. (1998). Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4):231–245. 13

[68] Fried, L., Shukla, S., and Sawyer, S. (2012). Global wind report: Annual market update 2011. Global Wind Energy Council, (GWEC). xiv, 2

[69] Fuglsang, P. and Madsen, H. A. (1995a). A design study of a 1 MW stall regulated rotor. Risø National Laboratory, Denmark. 8

[70] Fuglsang, P. L. and Madsen, H. A. (1995b). Optimization of stall regulated rotors. Technical report, American Society of Mechanical Engineers, New York, NY (United States). 8

[71] Giering, R. and Kaminski, T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474. 11

[72] Giguère, P. and Selig, M. (1997). Aerodynamic blade design methods for horizontal axis wind turbines. In *13th Annual Canadian Wind Energy Association Conference and Exhibition, Quebec City, Quebec, Canada, October*, pages 19–22. 9

[73] Giguère, P. and Selig, M. S. (1999). Design of a tapered and twisted blade for the NREL combined experiment rotor. Technical report, National Renewable Energy Lab., Golden, CO (US). 8

[74] Giguère, P., Selig, M. S., and Tangler, J. L. (1999). Blade design trade-offs using low-lift airfoils for stall-regulated HAWTs. *Journal of Solar Energy Engineering*, 121(4):217–223. 8

[75] Giles, M. B., Duta, M. C., Müller, J.-D., and Pierce, N. A. (2003). Algorithm developments for discrete adjoint methods. *AIAA Journal*, 41(2):198–205. 9

[76] Giles, M. B. and Pierce, N. A. (2000). An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3-4):393–415. 10

[77] Giles, M. B. and Pierce, N. A. (2001). Analytic adjoint solutions for the quasi-one-dimensional Euler equations. *Journal of Fluid Mechanics*, 426:327–345. 196

[78] Glauert, H. (1935). Airplane propellers. In *Aerodynamic Theory*, pages 169–360. Springer-Verlag New York Inc. 3, 6, 8

[79] Godunov, S. K. (1959). A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations. *Math. Sbornik*, 47:271–306. 63

[80] Griewank, A., Juedes, D., and Utke, J. (1996). Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software (TOMS)*, 22(2):131–167. 11

[81] Grinberg, L. and Karniadakis, G. E. (2011). Extrapolation-based acceleration of iterative solvers: Application to simulation of 3D flows. *Communications in Computational Physics*, 9(03):607–626. 17

[82] Gualdoni, F. (2014). *Design Optimization of Wind Turbines*. PhD thesis, Politecnico di Milano. 8

[83] Hafez, M., Parlette, E., and Salas, M. (1986). Convergence acceleration of iterative solutions of Euler equations for transonic flow computations. *Computational Mechanics*, 1(3):165–176. 16

[84] Hall, K. C. and Crawley, E. F. (1989). Calculation of unsteady flows in turbomachinery using the linearized Euler equations. *AIAA Journal*, 27(6):777–787. 6

[85] Hall, K. C., Thomas, J. P., and Clark, W. S. (2002). Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA Journal*, 40(5):879–886. 3, 7, 15

[86] Hall, K. C., Thomas, J. P., and Dowell, E. H. (2000). Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows. *AIAA Journal*, 38(10):1853–1862. 17

[87] Hansen, A. and Butterfield, C. (1993). Aerodynamics of horizontal-axis wind turbines. *Annual Review of Fluid Mechanics*, 25(1):115–149. 8

[88] Hansen, M. O. L., Sørensen, J. N., Michelsen, J. A., and Sørensen, N. N. (1997). A global Navier–Stokes rotor prediction model. AIAA Paper 1997-0970. 6

[89] Hansen, M. O. L., Sørensen, J. N., Voutsinas, S., Sørensen, N., and Madsen, H. A. (2006). State of the art in wind turbine aerodynamics and aeroelasticity. *Progress in Aerospace Sciences*, 42(4):285–330. 6

[90] Harten, A., Lax, P. D., and Leer, B. v. (1983). On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61. 66

[91] Hascoet, L. and Pascual, V. (2013). The Tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions on Mathematical Software (TOMS)*, 39(3):20. 11

[92] Haselbacher, A. and Blazek, J. (2000). Accurate and efficient discretization of Navier–Stokes equations on mixed grids. *AIAA Journal*, 38(11):2094–2102. 71, 72

[93] Haselbacher, A. C. (1999). *A Grid-Transparent Numerical Method for Compressible Viscous Flows on Mixed Unstructured Grids*. PhD thesis, Loughborough University. 22, 38, 56, 60, 73

[94] He, Y. and Agarwal, R. K. (2014). Shape optimization of NREL S809 airfoil for wind turbine blades using a multi-objective genetic algorithm. AIAA Paper 2014-2845. 9

[95] Hendriks, H., Schepers, G., van Engelen, T. G., Stern, A. J., and Boerstra, G. K. (1996). Aeroelastically optimised cost efficient wind turbine: A case study. *Netherlands Energy Research Foundation ECN*. 8

[96] Hibbs, B. and Radkey, R. (1983). Calculating rotor performance with the revised PROP computer code. Technical Report RFP-3508, UC-60, Wind Energy Research Center, Rockwell International, Rocky Flats Plant, Golden, CO. 8

[97] Hicks, R. M. and Henne, P. A. (1978). Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412. 206

[98] Hirsch, C. (2007). *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Butterworth-Heinemann, Oxford, UK. 23, 25, 26, 28, 29, 38, 63

[99] Hoadley, D., Madsen, H., and Bouras, B. (1993). Aerofoil section design and assessment. *Final Rep. Contract JOUR*, 79. 8

[100] Hogan, R. J. (2014). Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Transactions on Mathematical Software (TOMS)*, 40(4):26. 11

[101] Holmes, P., Lumley, J. L., and Berkooz, G. (1998). *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge University Press. 16

[102] Howison, J. and Ekici, K. (2013a). Unsteady analysis of wind turbine flows using the harmonic balance method. AIAA Paper 2013-1107. 8, 15

[103] Howison, J. and Ekici, K. (2013b). Wind turbine dynamic stall analysis using harmonic balance and correlation-based transition models. AIAA Paper 2013-2951. 15

[104] Howison, J. and Ekici, K. (2015). Dynamic stall analysis using harmonic balance and correlation-based γ–$\mathrm{Re}_{\theta t}$ transition models for wind turbine applications. *Wind Energy*, 18(12):2047–2063. 15, 44, 148, 151

[105] Howison, J. C. (2015). *Aeroelastic Analysis of a Wind Turbine Blade Using the Harmonic Balance Method*. PhD thesis, University of Tennessee. 8, 51, 148, 150, 175, 177

[106] Huang, H. (2013). *Shape Optimization of Turbomachinery Blades Using an Adjoint Harmonic Balance Method*. PhD thesis, University of Tennessee - Knoxville. 7

[107] Huang, H. and Ekici, K. (2013). An efficient harmonic balance method for unsteady flows in cascades. *Aerospace Science and Technology*, 29(1):144–154. 15

[108] Huang, H. and Ekici, K. (2014a). A discrete adjoint harmonic balance method for turbomachinery shape optimization. *Aerospace Science and Technology*, 39:481–490. 206, 207

[109] Huang, H. and Ekici, K. (2014b). Stabilization of high-dimensional harmonic balance solvers using time spectral viscosity. *AIAA Journal*, 52(8):1784–1794. 15

[110] Jameson, A. (1986). A vertex based multigrid algorithm for three dimensional compressible flow calculations. In *ASME Symposium on Numerical Methods for Compressible Flow, Anaheim*. 15

[111] Jameson, A. (1988). Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260. 9

[112] Jameson, A. (1991). Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. AIAA Paper 1991-43675. 75

[113] Jameson, A. (1993). Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence in transonic and hypersonic flows. AIAA Paper 1993-3359. 63

[114] Jameson, A. (2003). Aerodynamic shape optimization using the adjoint method. *Lectures at the Von Karman Institute, Brussels*. 207

[115] Jameson, A. and Baker, T. J. (1983). Solution of the Euler equations for complex configurations. AIAA Paper 1983-83. 6, 15, 79

[116] Jameson, A., Baker, T. J., and Weatherill, N. P. (1986). Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 1986-0103. 79, 80

[117] Jameson, A., Schmidt, W., and Turkel, E. (1981). Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. AIAA Paper 1981-1259. 63, 193

[118] Jespersen, D. C. and Buning, P. G. (1985). Accelerating an iterative process by explicit annihilation. *SIAM Journal on Scientific and Statistical Computing*, 6(3):639–651. 16

[119] Jones, B. W. (2015). Mesh Adaptation Through r-refinement Using a Truss Network Analogy. Master's thesis, University of Cape Town. 14, 106, 107, 137

[120] Kaminsky, A. L., Djeddi, R., and Ekici, K. (2017a). Convergence acceleration of continuous adjoint solvers using a reduced-order model. *International Journal for Numerical Methods in Fluids*, 86(9):582–606. xix, xx, 96, 196, 198, 199

[121] Kaminsky, A. L., Djeddi, R., and Ekici, K. (2017b). An efficient reduced-order-model for accurate projection of adjoint sensitivities. AIAA Paper 2017-0037. 96

[122] Kaminsky, A. L. and Ekici, K. (2016). Sensitivity and stability derivative analysis using an efficient adjoint harmonic balance technique. AIAA Paper 2016-0808. 202

[123] Karypis, G. and Kumar, V. (1998). A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.

University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN. 50, 113

[124] Kirn, S., Alonso, J. J., and Jameson, A. (2002). Design optimization of high-lift configurations using a viscous continuous adjoint method. AIAA Paper 2002-0844. 9

[125] Knoll, D. A. and Keyes, D. E. (2004). Jacobian-free Newton–Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397. 100

[126] Laflin, K. R. (1997). *Solver-Independent r-refinement Adaptation for Dynamic Numerical Simulations*. PhD thesis, North Carolina State University. 14

[127] Landon, R. (2000). NACA 0012 oscillatory and transient pitching. Technical report, Aircraft Research Association Ltd, Bedford (United Kingdom). xii, xiii, 152, 162, 163, 164, 165, 166, 167

[128] Langtry, R. B. (2006). *A Correlation-Based Transition Model Using Local Variables for Unstructured Parallelized CFD Codes*. PhD thesis, University of Stuttgart. 44, 150

[129] Langtry, R. B. and Menter, F. R. (2009). Correlation-based transition modeling for unstructured parallelized computational fluid dynamics codes. *AIAA Journal*, 47(12):2894–2906. 44, 148

[130] Lantz, E., Wiser, R., and Hand, M. (2012). The past and future cost of wind energy. National Renewable Energy Laboratory, Golden, CO, Report No. NREL/TP-6A20-53510. 2

[131] Lanzafame, R. and Messina, M. (2009). Optimal wind turbine design to maximize energy production. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 223(2):93–101. 8

[132] Legensky, S., Edwards, D., Bush, R., Poirier, D., Rumsey, C., Cosner, R., and Towne, C. (2002). CFD general notation system (CGNS)-status and future directions. AIAA Paper 2002-752. 57

[133] Leonard, B. (1994). Comparison of truncation error of finite-difference and finite-volume formulations of convection terms. *Applied Mathematical Modelling*, 18(1):46–50. 56

[134] Lépine, J., Guibault, F., Trépanier, J.-Y., and Pépin, F. (2001). Optimized nonuniform rational B-Spline geometrical representation for aerodynamic design of wings. *AIAA Journal*, 39(11):2033–2041. 206

[135] Lesoinne, M. and Farhat, C. (1996). Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134(1-2):71–90. 35, 83

[136] Li, Z., Chen, H., Zhang, Y., and Fu, S. (2015). Grid convergence study on a finite volume code NSAWET. AIAA Paper 2015-0812. 136

[137] Lin, T., Guan, Z., Chang, J., and Lo, S. (2014). Vertex-ball spring smoothing: An efficient method for unstructured dynamic hybrid meshes. *Computers & Structures*, 136:24–33. 14

[138] Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528. 210

[139] Lozano, C. and Ponsin, J. (2012). Remarks on the numerical solution of the adjoint quasi-one-dimensional Euler equations. *International Journal for Numerical Methods in Fluids*, 69(5):966–982. 118, 196, 197, 198

[140] Lucia, D. J., Beran, P. S., and Silva, W. A. (2004). Reduced-order modeling: New approaches for computational physics. *Progress in Aerospace Sciences*, 40(1):51–117. 17

[141] Luo, H., Baum, J., and Lohner, R. (1995). An improved finite volume scheme for compressible flows on unstructured grids. AIAA Paper 1995-348. 86

[142] Ma, R., Chang, X., Zhang, L., He, X., and Li, M. (2015). On the geometric conservation law for unsteady flow simulations on moving mesh. *Procedia Engineering*, 126:639–644. 36, 84

[143] Mani, K. (2009). *Application of the Discrete Adjoint Method to Coupled Multidisciplinary Unsteady Flow Problems for Error Estimation and Optimization*. PhD thesis, University of Wyoming. 203

[144] Markovinović, R. and Jansen, J. (2006). Accelerating iterative solution methods using reduced-order models as solution predictors. *International Journal for Numerical Methods in Engineering*, 68(5):525–541. 17, 101

[145] Martinelli, L. (1987). *Calculations of Viscous Flows with a Multigrid Method*. PhD thesis, Princeton Univ., NJ. 76, 78

[146] Mavriplis, D. J. (1995). Three-dimensional multigrid Reynolds-averaged Navier–Stokes solver for unstructured meshes. *AIAA Journal*, 33(3):445–453. 72, 73

[147] Mavriplis, D. J. (1998). Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics*, 145(1):141–165. 96

[148] Mavriplis, D. J. and Jameson, A. (1990). Multigrid solution of the Navier–Stokes equations on triangular meshes. *AIAA Journal*, 28(8):1415–1425. 76, 78

[149] McCroskey, W. J. (1982). Unsteady airfoils. *Annual Review of Fluid Mechanics*, 14(1):285–311. 162

[150] McMullen, M. S. (2003). *The Application of Non-linear Frequency Domain Methods to the Euler and Navier–Stokes Equations*. PhD thesis, Stanford University. 162, 164, 165, 166, 167

[151] McRae, D. S. (2000). r-Refinement grid adaptation algorithms and issues. *Computer Methods in Applied Mechanics and Engineering*, 189(4):1161–1182. 14

[152] Medida, S. (2014). *Correlation-Based Transition Modeling for External Aerodynamic Flows*. PhD thesis, University of Maryland College Park. 44

[153] Medida, S. and Baeder, J. (2011). Application of the correlation-based γ–$\mathrm{Re}_{\theta t}$ transition model to the Spalart-Allmaras turbulence model. AIAA Paper 2011-3979. 44

[154] Menter, F., Langtry, R., Likki, S., Suzen, Y., Huang, P., and Völker, S. (2006). A correlation-based transition model using local variables - Part I: model formulation. *Journal of Turbomachinery*, 128(3):413–422. 45, 46

[155] Menter, F. R. (1993). Zonal two equation k-ω turbulence models for aerodynamic flows. AIAA Paper 1993-2906. 44

[156] Menter, F. R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32(8):1598–1605. 44

[157] Mešina, M. (1977). Convergence acceleration for the iterative solution of the equations X = AX + f. *Computer Methods in Applied Mechanics and Engineering*, 10(2):165–173. 16

[158] Miller, M., Shipley, D., Young, T., Robinson, M., Luttges, M., and Simms, D. (1995). Combined Experiment Phase 2. Data Characterization. Technical report, National Renewable Energy Lab., Golden, CO (United States); Colorado Univ., Boulder, CO (United States). 8

[159] Moran, M. J., Shapiro, H. N., Boettner, D. D., and Bailey, M. B. (2010). *Fundamentals of engineering thermodynamics*. John Wiley & Sons, Hoboken, New Jersey. 29

[160] Morkovin, M. V. (1962). Effects of compressibility on turbulent flows. *Mécanique de la Turbulence*, 367:380. 37

[161] Murray, G. (2013). Rotation about an arbitrary axis in 3 dimensions. http://twist-and-shout.appspot.com/. Accessed 2018. 95

[162] Muzaferija, S. and Gosman, D. (1997). Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology. *Journal of Computational Physics*, 138(2):766–787. 60

[163] Nadarajah, S. and Jameson, A. (2000). A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. AIAA Paper 2000-0667. 9

[164] Naumann, U. (2012). *The art of differentiating computer programs: An introduction to algorithmic differentiation*, volume 24. SIAM. 11

[165] Naumann, U., Utke, J., Heimbach, P., Hill, C., Ozyurt, D., Wunsch, C., Fagan, M., Tallent, N., and Strout, M. (2006). Adjoint code by source transformation with OpenAD/F. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS). 11

[166] Osher, S. and Solomon, F. (1982). Upwind difference schemes for hyperbolic systems of conservation laws. *Mathematics of Computation*, 38(158):339–374. 63

[167] Pironneau, O. (1974). On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(01):97–110. 9

[168] Pletcher, R. H., Tannehill, J. C., and Anderson, D. (2012). *Computational Fluid Mechanics and Heat Transfer*. CRC Press. 51

[169] Pulliam, T. H. (1984). Euler and Thin Layer Navier–Stokes codes: ARC2D, ARC3D. In *Notes for Computational Fluid Dynamics User's Workshop*, pages 15–1. 6

[170] Pulliam, T. H. and Steger, J. L. (1980). Implicit finite-difference simulations of three-dimensional compressible flow. *AIAA Journal*, 18(2):159–167. 6

[171] Ramsay, R., Hoffman, M., and Gregorek, G. (1995). Effects of grid roughness and pitch oscillations on the S809 airfoil. *Technical Paper 442-7817, NREL*. 149, 151, 226

[172] Rapún, M.-L. and Vega, J. M. (2010). Reduced order models based on local POD plus Galerkin projection. *Journal of Computational Physics*, 229(8):3046–3063. 17

[173] Rendall, T. C. and Allen, C. B. (2009). Efficient mesh motion using radial basis functions with data reduction algorithms. *Journal of Computational Physics*, 228(17):6231–6249. 82

[174] Reynolds, O. (1895). On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London. A*, 186:123–164. 37, 38

[175] Reynolds, O. (1901). On the extent and action of the heating surface of steam boilers. *Papers on Mechanical and Physical Subjects*, page 81. 38

[176] Ritlop, R. M. (2009). *Toward the Aerodynamic Shape Optimization of Wind Turbine Profiles*. PhD thesis, McGill University. 8

[177] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372. 63, 64

[178] Sankar, L., Malone, J., and Tassa, Y. (1981). An implicit conservative algorithm for steady and unsteady three-dimensional transonic potential flows. AIAA Paper 1981-1016. 6

[179] Sankar, N., Wake, B., and Lekoudis, S. (1986). Solution of the unsteady Euler equations for fixed and rotor wing configurations. *Journal of Aircraft*, 23(4):283–289. 6

[180] Schmitt, V. and Charpin, F. (1979). Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers. *Experimental Data Base for Computer Program Assessment*, 4. xix, 176, 177

[181] Selig, M. S. and Coverstone-Carroll, V. L. (1996). Application of a genetic algorithm to wind turbine design. *Journal of Energy Resources Technology*, 118(1):22–28. 8, 9

[182] Selig, M. S. and Tangler, J. L. (1995). Development and application of a multipoint inverse design method for horizontal axis wind turbines. *Wind Engineering*, pages 91–105. 9

[183] Shiriaev, D. and Griewank, A. (1996). ADOL-F: Automatic differentiation of Fortran codes. *Computational Differentiation: Techniques, Applications, and Tools*, pages 375–384. 11

[184] Shterev, K. (2015). Iterative process acceleration of calculation of unsteady, viscous, compressible, and heat-conductive gas flows. *International Journal for Numerical Methods in Fluids*, 77(2):108–122. 17

[185] Sidi, A. (2012). Review of two vector extrapolation methods of polynomial type with applications to large-scale problems. *Journal of Computational Science*, 3(3):92–101. 16

[186] Simms, D., Schreck, S., Hand, M., and Fingersh, L. J. (2001). NREL unsteady aerodynamics experiment in the NASA-Ames wind tunnel: A comparison of predictions to measurements. *National Renewable Energy Laboratory Colorado, USA*. 222

[187] Skelboe, S. (1980). Computation of the periodic steady-state response of nonlinear networks by extrapolation methods. *Circuits and Systems, IEEE Transactions on*, 27(3):161–175. 16

[188] Slater, J. W. (2008). NPARC alliance validation archive: ONERA M6 wing. http://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing.html. Accessed 2018. xix, 176, 177

[189] Smith, D. A., Ford, W. F., and Sidi, A. (1987). Extrapolation methods for vector sequences. *SIAM Review*, 29(2):199–233. 16

[190] Somers, D. M. (1997). Design and experimental results for the S809 airfoil. Technical report, National Renewable Energy Lab., Golden, CO (United States). 223

[191] Sørensen, N. N. and Hansen, M. O. (1998). Rotor performance predictions using a Navier–Stokes method. AIAA Paper 1998-0025. 6

[192] Sørensen, N. N. and Michelsen, J. (2000). Aerodynamic predictions for the unsteady aerodynamics experiment Phase-II rotor at the National Renewable Energy Laboratory. AIAA Paper 2000-0037. 6

[193] Soria Guerrero, M. (2000). *Parallel multigrid algorithms for computational fluid dynamics and heat transfer*. PhD thesis, Universitat Politècnica de Catalunya. 113

[194] Spalart, P. R. and Allmaras, S. R. (1992). A one-equation turbulence model for aerodynamic flows. AIAA Paper 1992-0439. 40, 41, 98

[195] Srinivasan, G. and McCroskey, W. (1988). Navier–Stokes calculations of hovering rotor flowfields. *Journal of Aircraft*, 25(10):865–874. 6

[196] Srinivasan, G., Raghavan, V., Duque, E., and McCroskey, W. (1993). Flowfield analysis of modern helicopter rotors in hover by Navier–Stokes method. *Journal of the American Helicopter Society*, 38(3):3–13. 6

[197] Stamatiadis, S., Prosmiti, R., and Farantos, S. (2000). AUTO_DERIV: Tool for automatic differentiation of a FORTRAN code. *Computer Physics Communications*, 127(2):343–355. 11

[198] Stokes, G. G. (1880). On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids. *Transactions of the Cambridge Philosophical Society*, 8. 28

[199] Straka, C. W. (2005). ADF95: Tool for automatic differentiation of a FORTRAN code designed for large numbers of independent variables. *Computer Physics Communications*, 168(2):123–139. 11

[200] Sutherland, W. (1893). LII. The viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531. 29

[201] Swanson, R. C. and Turkel, E. (1992). On central-difference and upwind schemes. *Journal of Computational Physics*, 101(2):292–306. 135, 145

[202] Swanson, R. C., Turkel, E., and White, J. A. (1992). An effective multigrid method for high-speed flows. *International Journal for Numerical Methods in Biomedical Engineering*, 8(9):671–681. 78

[203] Tai, C. H., Zhao, Y., and Liew, K. (2005). Parallel-multigrid computation of unsteady incompressible viscous flows using a matrix-free implicit method and high-resolution

characteristics-based scheme. *Computer Methods in Applied Mechanics and Engineering*, 194(36-38):3949–3983. 113

[204] Tang, T. (2005). Moving mesh methods for computational fluid dynamics. *Journal of Contemporary Mathematics*, 383:141–174. 13

[205] Tardif, P.-O. and Nadarajah, S. (2017). Three-dimensional aeroelastic solutions via the nonlinear frequency-domain method. *AIAA Journal*, pages 1–17. 36, 83, 84

[206] Tatum, K. and Slater, J. (1999). The validation archive of the NPARC Alliance. AIAA Paper 1999-747. 177

[207] Theofilis, V. (2011). Global linear instability. *Annual Review of Fluid Mechanics*, 43:319–352. 16

[208] Thomas, J. P., Dowell, E. H., and Hall, K. C. (2003). Three-dimensional transonic aeroelasticity using proper orthogonal decomposition-based reduced-order models. *Journal of Aircraft*, 40(3):544–551. 17

[209] Thomas, P. and Lombard, C. (1979). Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037. 35, 83

[210] Tromeur-Dervout, D. and Vassilevski, Y. (2007). POD acceleration of fully implicit solver for unsteady nonlinear flows and its application on grid architecture. *Advances in Engineering Software*, 38(5):301–311. 17

[211] Tyson, W. C., Derlaga, J. M., Choudhary, A., and Roy, C. J. (2015). Comparison of r-adaptation techniques for 2-D CFD applications. AIAA Paper 2015-2611. 14

[212] US Energy Information Administration (2011). Annual Energy Outlook 2011: With Projections to 2035. Government printing office. xiv, 1

[213] Usab, Jr., W. J. and Murman, E. M. (1983). Embedded mesh solutions of the Euler equation using a multiple-grid method. AIAA Paper 83-1946. 90

[214] Van Leer, B. (1982). Flux-vector splitting for the Euler equations. In *8th International Conference on Numerical Methods in Fluid Dynamics*, pages 507–512. Springer-Verlag New York Inc. 63

[215] Van Leer, B., Lee, W.-T., Roe, P. L., Powell, K. G., and Tai, C.-H. (1992). Design of optimally smoothing multistage schemes for the Euler equations. *International Journal for Numerical Methods in Biomedical Engineering*, 8(10):761–769. 76

[216] Vassberg, C. and Jameson, A. (2009). In pursuit of grid convergence, Part I: Two-dimensional Euler solutions. AIAA Paper 2009-4114. 134, 137

[217] Venkatakrishnan, V. (1989). Newton solution of inviscid and viscous problems. *AIAA Journal*, 27(7):885–891. 137

[218] Venkatakrishnan, V. (1993). On the accuracy of limiters and convergence to steady state solutions. AIAA Paper 1993-880. 69

[219] Venkatakrishnan, V. (1995). Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, 118(1):120–130. 69

[220] Venkatakrishnan, V. and Mavriplis, D. (1996). Implicit method for the computation of unsteady flows on unstructured grids. *Journal of Computational Physics*, 127(2):380–397. 56

[221] Vijayan, P. and Kallinderis, Y. (1994). A 3D finite-volume scheme for the Euler equations on adaptive tetrahedral grids. *Journal of Computational Physics*, 113(2):249–267. 77, 78

[222] Wang, D. and He, L. (2010). Adjoint aerodynamic design optimization for blades in multistage turbomachines Part I: Methodology and verification. *Journal of Turbomachinery*, 132(2):021011. 202

[223] Watanabe, Y., Iwashita, H., and Ito, M. (2007). Shape optimum design of horizontal axis wind turbine in low Reynolds number range. In *European Wind Energy Conference, Milan, Italy*. 8

[224] Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396. 82

[225] Whitfield, D. (1984). Three-dimensional unsteady Euler equations solution using flux vector splitting. AIAA Paper 1984-1552. 90

[226] Wilcox, D. C. (1988). Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, 26(11):1299–1310. 38

[227] Wilcox, D. C. et al. (1993). *Turbulence modeling for CFD, Second Edition*. DCW industries La Canada, CA. 38, 39, 40, 41

[228] Wilson, R. (1980). Wind-turbine aerodynamics. *Journal of Wind Engineering and Industrial Aerodynamics*, 5(3):357–372. 5

[229] Wolfe, P. (1982). Checking the calculation of gradients. *ACM Transactions on Mathematical Software (TOMS)*, 8(4):337–343. 120

[230] Wright, S. and Nocedal, J. (1999). Numerical optimization. *Springer Science*, 35(67-68):7. 210

[231] Wynn, P. (1962). Acceleration techniques for iterated vector and matrix problems. *Mathematics of Computation*, 16(79):301–322. 15

[232] Xu, G. and Sankar, L. N. (2000). Computational study of horizontal axis wind turbines. *Journal of Solar Energy Engineering*, 122(1):35–39. 6

[233] Yano, M. and Darmofal, D. L. (2012). Case C1.3: Flow over the NACA 0012 airfoil: Subsonic inviscid, transonic inviscid, and subsonic laminar flows. In *First International Workshop on High-Order CFD Methods*. xii, 135, 137, 138

[234] Yu, W. and Blair, M. (2013). DNAD, a simple tool for automatic differentiation of Fortran codes using dual numbers. *Computer Physics Communications*, 184(5):1446–1452. 11

[235] Zeng, D. and Ethier, C. R. (2005). A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains. *Finite Elements in Analysis and Design*, 41(11):1118–1139. 14, 104

[236] Zhao, L. and Zhang, C. (2014). A parallel unstructured finite-volume method for all-speed flows. *Numerical Heat Transfer, Part B: Fundamentals*, 65(4):336–358. 50, 112

Appendices

A Navier-Stokes Equations in Rotating Frame of Reference (Special Cases)

Here the Navier-Stokes equations are defined in rotating frame of reference for special cases where the axis rotation is aligned with one of the main axes in the Cartesian coordinate system. For each case, equations for the calculation of the relative total energy, rothalpy, and static pressure as well as the source term vector including the Coriolis and centrifugal forces are provided.

In general, the Navier-Stokes equations in relative frame of reference for a steady rotation with constant angular velocity vector, $\vec{\omega}$, are given by

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \vec{U}_{\text{rel}}\, d\mathcal{V} + \oint_{\partial\mathcal{V}} \left[\vec{F}_{c_{\text{rel}}} - \vec{F}_{v_{\text{rel}}}\right]\, dS = \int_{\mathcal{V}} \vec{Q}_{\text{rel}}\, d\mathcal{V} \tag{A.1}$$

where $\vec{U}_{\text{rel}}$ is the vector of conservation variables in relative frame of reference defined as

$$\vec{U}_{\text{rel}} = \begin{bmatrix} \rho \\ \rho\vec{v}_{\text{rel}} \\ \rho E_{\text{rel}} \end{bmatrix} \tag{A.2}$$

Here, the convective and viscous fluxes are given by Eq. (2.33) and (2.13).

A.1 Rotation about x-Axis

Assuming that the rotation axis coincides with x-coordinate axis, the angular velocity vector, $\vec{\omega}$, for a steady rotation with magnitude Ω (positive according to the right-hand-rule), is given by

$$\vec{\omega} = [\Omega, 0, 0]^T \tag{A.3}$$

In such case, the position vector perpendicular to the rotation axis is given by

$$\vec{r}_n = [0, y, z]^T \tag{A.4}$$

where y and z are the y- and z-components of the position vector for an arbitrary point in space (see Figure 2.2). Therefore, the Coriolis and centrifugal forces can be simplified as

$$\vec{f}_{\text{Cor}} = -2\left(\vec{\omega} \times \vec{v}_{\text{rel}}\right) = -2 \begin{bmatrix} 0 \\ -\Omega w_{\text{rel}} \\ \Omega v_{\text{rel}} \end{bmatrix} \tag{A.5}$$

$$\vec{f}_{\text{cent}} = -\vec{\omega} \times \left(\vec{\omega} \times \vec{r}\right) = \begin{bmatrix} 0 \\ \Omega^2 y \\ \Omega^2 z \end{bmatrix} \tag{A.6}$$

The entrainment or rotational velocity vector is given by

$$\vec{v}_{\text{rot}} = \vec{\omega} \times \vec{r} = [0, -\Omega z, \Omega y]^T \tag{A.7}$$

with its magnitude (squared) being

$$|\vec{v}_{\text{rot}}|^2 = \Omega^2(y^2 + z^2) = \Omega^2|\vec{r}_n|^2. \tag{A.8}$$

The above relation is used for the calculation of the relative total energy, rothalpy, and static pressure which are now given by

$$E_{\text{rel}} = e + \frac{u_{\text{rel}}^2 + v_{\text{rel}}^2 + w_{\text{rel}}^2}{2} - \frac{\Omega^2(y^2 + z^2)}{2} \tag{A.9}$$

$$I = H_{\text{rel}} - \frac{\Omega^2(y^2 + z^2)}{2} \tag{A.10}$$

$$p = (\gamma - 1)\left[\rho E - \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho} + \rho\,\frac{\Omega^2(y^2 + z^2)}{2}\right] \tag{A.11}$$

Finally, the source terms vector can be rewritten in a simplified form using the special Coriolis and centrifugal forces for the case of rotation about x-axis as

$$\vec{Q}_{\text{rel}} = \begin{bmatrix} 0 \\ 0 \\ \rho\Omega\left(\Omega y + 2w_{\text{rel}}\right) \\ \rho\Omega\left(\Omega z - 2v_{\text{rel}}\right) \\ 0 \end{bmatrix} \tag{A.12}$$

A.2 Rotation about y-Axis

Similarly, for the cases when the rotation axis coincides with y-coordinate axis, the angular velocity vector, $\vec{\omega}$, for a steady rotation with magnitude Ω (positive according to the right-hand-rule), is given by

$$\vec{\omega} = [0, \Omega, 0]^T \tag{A.13}$$

and the position vector perpendicular to the rotation axis is now given by

$$\vec{r}_n = [x, 0, z]^T \tag{A.14}$$

Thus, the Coriolis and centrifugal forces can be simplified as

$$\vec{f}_{\text{Cor}} = -2\left(\vec{\omega} \times \vec{v}_{\text{rel}}\right) = -2 \begin{bmatrix} \Omega w_{\text{rel}} \\ 0 \\ -\Omega u_{\text{rel}} \end{bmatrix} \tag{A.15}$$

$$\vec{f}_{\text{cent}} = -\vec{\omega} \times \left(\vec{\omega} \times \vec{r}\right) = \begin{bmatrix} \Omega^2 x \\ 0 \\ \Omega^2 z \end{bmatrix} \tag{A.16}$$

The entrainment or rotational velocity vector is now given by

$$\vec{v}_{\text{rot}} = \vec{\omega} \times \vec{r} = [\Omega z, 0, -\Omega x]^T \tag{A.17}$$

with its magnitude (squared) being

$$|\vec{v}_{\text{rot}}|^2 = \Omega^2(x^2 + z^2) = \Omega^2|\vec{r}_n|^2. \tag{A.18}$$

Therefore, the relative total energy, rothalpy, and static pressure can be calculated using the following equations

$$E_{\text{rel}} = e + \frac{u_{\text{rel}}^2 + v_{\text{rel}}^2 + w_{\text{rel}}^2}{2} - \frac{\Omega^2(x^2 + z^2)}{2} \tag{A.19}$$

$$I = H_{\text{rel}} - \frac{\Omega^2(x^2 + z^2)}{2} \tag{A.20}$$

$$p = (\gamma - 1)\left[\rho E - \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho} + \rho\,\frac{\Omega^2(x^2 + z^2)}{2}\right] \tag{A.21}$$

Eventually, the source terms vector can be rewritten for the case of rotation about y-axis as

$$\vec{Q}_{\text{rel}} = \begin{bmatrix} 0 \\ \rho\Omega\left(\Omega x - 2w_{\text{rel}}\right) \\ 0 \\ \rho\Omega\left(\Omega z + 2u_{\text{rel}}\right) \\ 0 \end{bmatrix} \tag{A.22}$$

A.3 Rotation about z-Axis

Finally, for the cases when the rotation axis coincides with z-coordinate axis, the angular velocity vector, $\vec{\omega}$, is given by

$$\vec{\omega} = [0, 0, \Omega]^T \tag{A.23}$$

and the position vector perpendicular to the rotation axis is now given by

$$\vec{r}_n = [x, y, 0]^T \tag{A.24}$$

This time, the Coriolis and centrifugal forces can be simplified as

$$\vec{f}_{\text{Cor}} = -2\left(\vec{\omega} \times \vec{v}_{\text{rel}}\right) = -2 \begin{bmatrix} -\Omega v_{\text{rel}} \\ \Omega u_{\text{rel}} \\ 0 \end{bmatrix} \tag{A.25}$$

$$\vec{f}_{\text{cent}} = -\vec{\omega} \times (\vec{\omega} \times \vec{r}) = \begin{bmatrix} \Omega^2 x \\ \Omega^2 y \\ 0 \end{bmatrix} \tag{A.26}$$

with the entrainment or rotational velocity vector being

$$\vec{v}_{\text{rot}} = \vec{\omega} \times \vec{r} = [-\Omega y, \Omega x, 0]^T \tag{A.27}$$

and its magnitude (squared) being

$$|\vec{v}_{\text{rot}}|^2 = \Omega^2(x^2 + y^2) = \Omega^2 |\vec{r}_n|^2. \tag{A.28}$$

The relative total energy, rothalpy, and static pressure can be calculated using the following equations

$$E_{\text{rel}} = e + \frac{u_{\text{rel}}^2 + v_{\text{rel}}^2 + w_{\text{rel}}^2}{2} - \frac{\Omega^2(x^2 + y^2)}{2} \tag{A.29}$$

$$I = H_{\text{rel}} - \frac{\Omega^2(x^2 + y^2)}{2} \tag{A.30}$$

$$p = (\gamma - 1)\left[\rho E - \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho} + \rho\,\frac{\Omega^2(x^2 + y^2)}{2}\right] \tag{A.31}$$

while the source terms vector for the case of rotation about z-axis given by

$$\vec{Q}_{\mathrm{rel}} = \begin{bmatrix} 0 \\ \rho\Omega\left(\Omega x + 2v_{\mathrm{rel}}\right) \\ \rho\Omega\left(\Omega y - 2u_{\mathrm{rel}}\right) \\ 0 \\ 0 \end{bmatrix} \tag{A.32}$$

Vita

Reza Djeddi was born in 1984 in Tehran, Iran. After graduating from high school, he attended Amirkabir University of Technology (Tehran Polytechnic) where he received a BSc in Naval Architecture and Marine Engineering in 2008. Reza ranked second in the nationwide entrance exam for graduate studies and attended Sharif University of Technology where he received an MSc in Mechanical Engineering. In 2013, he moved to the United States to pursue his doctorate degree in Mechanical Engineering at the University of Tennessee, Knoxville where he also received an MSc in Aerospace Engineering in addition to a graduate minor in Computational Science. Reza is a member of the Sigma Xi scientific research society, Phi Kappa Phi honor society, American Institute of Aeronautics and Astronautics (AIAA) and American Society of Mechanical Engineers (ASME).