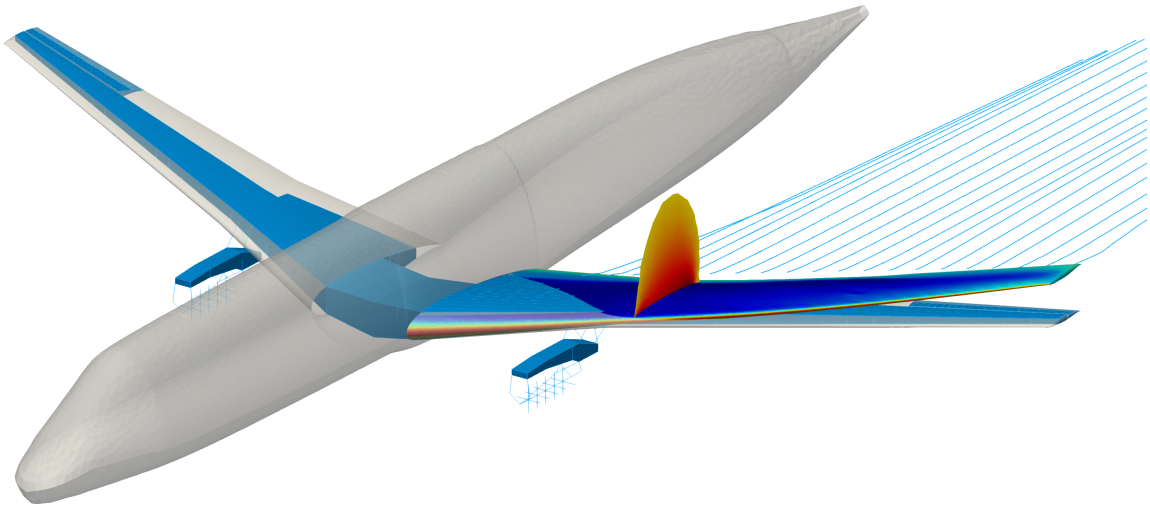
Steady Transonic Aerodynamic and Aeroelastic Modeling for Preliminary Aircraft Design

by
Adrien Crovato



Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy in **Aerospace Engineering**
at the **University of Liège**
in **October 2020**

LIÈGE université

Steady Transonic Aerodynamic and Aeroelastic Modeling for Preliminary Aircraft Design

A thesis submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy in **Aerospace Engineering**
at the
University of Liège
in
October 2020

by
Adrien Crovato

M.Sc.Eng, University of Liège, 2015
B.ASc., University of Liège, 2013

Jury:
Prof. Grigorios Dimitriadis, University of Liège (adviser)
Prof. Vincent E. Terrrapon, University of Liège (co-adviser)
Prof. Koen Hillewaert, University of Liège (president)
Dr. Romain Boman, University of Liège
Prof. Laurent Joly, ISAE-SUPAERO
Dr. Carlos Breviglieri, Embraer S.A.
Dr. Marco Carini, ONERA

Department of Aerospace and Mechanical Engineering, University of Liège
Academic Year 2020-2021

Per i miei nonni e le mie nonne

Abstract

Modern aircraft usually have light and flexible wings with large aspect ratio, which are subjected to significant deformations. The aeroelastic behavior of such wings is of paramount importance as it affects both the structural design and the aircraft performance. The present thesis aims at assessing the impact of aerodynamic modeling on transonic aerodynamic and aeroelastic computations performed in the preliminary stage of the aircraft design process, and at developing an aerodynamic modeling tool performing such computations with low computational cost.

First, the different levels of fidelity commonly used for aerodynamic modeling in aircraft design are investigated. The aerodynamic loads predicted by the different models and the computational cost of the solutions are compared using two rigid wings simulated in transonic flight conditions. The results show that linear modeling is not suitable for transonic flow calculations as it cannot capture shocks, and that higher-fidelity models are too computationally expensive to be used routinely in the preliminary design stage. On the other hand, nonlinear potential solutions offer an excellent trade-off between accuracy and computational cost. Accordingly, the main full potential formulations and their solution techniques are subsequently reviewed.

Two specific methods for solving the full potential equation are then developed and implemented. The first is the field panel method, an extension to the widely used and computationally effective panel method. Numerical calculations show that the iterative field panel approach implemented during this work yields satisfactory results for subcritical flows, but smears shock waves and thus cannot model transonic flows accurately. Moreover, the computational cost of the method is prohibitive for such computations. The second method is based on a continuous Galerkin, unstructured, finite element formulation. The code developed in the context of this work, `Flow`, is compared to a commercial state-of-the-art solver, `Tranair`, using various two and three-dimensional cases. In practice, `Flow`'s accuracy and computational cost are similar to `Tranair`, but the code is open-source.

The modular design of `Flow` allows the code to be easily coupled with `CUPyDO`, an in-house fluid-structure interaction code, which is then used to perform static transonic aeroelastic computations. In this context, the different aerodynamic levels of fidelity are compared using a flexible benchmark wing. For the wing shape considered in the present work, the results illustrate that linear potential methods yield sufficiently reliable static deformations, provided they model wing camber. On the other hand, nonlinear methods yield more accurate aerodynamic predictions but their computational cost is significantly higher. Consequently, static aeroelastic computations could be efficiently performed following a multi-fidelity approach, whereby a linear model would first be used to obtain the deformed wing shape, and a nonlinear model would then be used to compute the aerodynamic loads on that deformed wing shape.

Acknowledgments

A PhD thesis is usually considered as the accomplishment of one researcher. However, it is often, if not always, the result of many people's contributions, who sometimes even work in the shadows. I would like to take a few lines to express my deep gratitude to these people.

I would first like to thank my adviser, Grigorios Dimitriadis, and my co-adviser, Vincent Terrapon, for giving me the opportunity to carry out a thesis in their research groups. Greg and Vincent were always available to answer my questions and to share their insights on the various research topics that I am interested in. I could always count on them whenever I needed advice. I would also like to thank the Aeroelastic Tailoring group at Embraer for setting up such an ambitious project and giving us the opportunity to join them. I am also deeply grateful to the members of the Research and Development team, and specifically to Alex, Carlos, Eduardo, Gustavo, Hugo and Pedro, for their warm welcome during my research stay at São José dos Campos. Finally, I would like to thank the members of the jury, namely Koen Hillewaert, Laurent Joly and Marco Carini for the time they spent reading and evaluating my dissertation.

I would like to thank the colleagues that I directly worked with during the past five years. More specifically, I first want to thank Romain Boman. He played a major role in the development of the different codes that I implemented during my PhD. Romain was always there to answer the tons of questions I had about computer science, and to discuss various ideas and projects that could improve software development and teamwork. I am also grateful to Luc Papeleux, who was of precious help regarding various programming aspects. I also would like to thank Hüseyin Güner, with whom I took part in the Embraer's aeroelastic tailoring journey. I am also grateful to David Thomas and Marco-Lucio Cerquaglia, the lead developers of `CUPyDO`, for their help with the code. Finally, I am thankful to Ludovic Noels, for our joint work in the Aerospace Design Project, to Christophe Geuzaine, for his help and advice with the `gmsh` software, and to Maarten Arnst and Vincent Denoël, for our collaboration during the Stochastic class.

I am deeply grateful to my family who always stands besides me. I specifically think about my parents, grandparents and my uncle, who always supported me regardless of my choices. I am also thankful to my martial arts instructors, Arnaud Lejeune and Léo Tamaki, who taught me, through commitment and kindness, how to adapt to and overcome adversity. I would also like to thank my friends, and more specifically Sophie, Benoit, Thomas, Baptiste, Benoit, Sébastien, and the Coco&co group, for their support. I am also thankful to my fellow PhD colleagues: Amaury, Pantoufl, Mariano, Sébastien, Arnaud, Nayan, Dominik, Kim, Joffrey, Kévin and Juan.

I finally would like to gratefully acknowledge the *Fonds National pour la Recherche Scientifique* which partly funded the present doctoral thesis through a *Funds for Research training in Industry and Agriculture* grant.

Contents

Chapter 1

Introduction

1.1 Motivation

Air traffic is increasing dramatically fast. Over the past fifteen years, the number of passengers traveling by air worldwide has increased from 2.5 billion to 4.7 billion ($+150\%$) and is expected to reach 16 billion ($+200\%$) by 2050 [1, 2]. In Europe, the number of flights per year is expected to increase from 10 million in 2011 to 25 million in 2050. To compensate this rise, and for the air transport sector to remain economically competitive, aircraft fuel consumption must be reduced. Moreover, this will allow the sector to keep its rather low environmental impact[1]. For example, *Flightpath 2050 - Europe's Vision for Aviation* [2] prescribes a reduction of 75% in $\mathrm{CO_2}$ and of 90% in $\mathrm{NO_x}$ gas emissions, as well as a 65% noise reduction with respect to engines made in the year 2000. Various strategies are being investigated to reduce aircraft fuel consumption and environmental impact, such as increasing aircraft engine efficiency, using new types of fuel, improving aircraft operations and air services, reducing structural weight, improving aerodynamic efficiency, etc. The combination of the last two approaches usually leads to the design of light and very flexible, highly loaded composite wings with high aspect ratios and complex shapes. For such wings, aeroelastic deformations cannot be ignored as they will yield highly different wing shapes depending on the flight condition. These wing shapes are constrained by various requirements. For example, the jig shape, *i.e.* the shape of the wing on the ground, must respect a given clearance between the wing and the ground. The flight shape must be designed so that it has low drag in cruise conditions. The wing shape obtained during a maneuver must withstand higher loads while allowing the aircraft to remain controllable. Aeroelasticity must therefore be integrated early in the aircraft design process[2], typically in the preliminary stage, during which aero-structural design and optimization are performed. Moreover, taking fluid-structure interaction into account in this early design stage also allows other fuel reduction strategies to be used, such as actively deflecting the control surfaces during flight to minimize the aerodynamic loads. Introducing such computations in the preliminary stage might also prevent future failures resulting from complex aeroelastic behaviors, hence making the design process more robust. During the preliminary design stage, many design

[1] In 2018, the air transport sector was responsible for 12% of $\mathrm{CO_2}$ emissions of the transport sector, while road transport's share was of 74%. The transport sector represented $20 - 25\%$ of the total $\mathrm{CO_2}$ emissions [3].

[2] The aircraft design process is usually split into three stages: conceptual, preliminary and detail. The conceptual stage aims at defining an aircraft configuration which meets the mission requirements. The preliminary stage aims at refining the design and optimizing the aircraft performance. The detail stage aims at finalizing the design, and manufacturing, testing and certifying the aircraft.

parameters and configurations are considered. Moreover, aeroelastic modeling involves both aerodynamic and structural computations. As a consequence, low-fidelity modeling methodologies are usually favored so that results can be obtained quickly [4]. These models are linear and their range of validity is theoretically restricted to the subsonic regime, where the velocity of the entire flowfield is lower than the speed of sound. However, modern transport aircraft fly in the transonic regime, for which the oncoming freestream flow is subsonic and accelerates locally to velocities beyond the speed of sound, hence creating pockets of supersonic flow. Nonlinear compressible and viscous effects become important in transonic flight conditions and cannot be properly captured by low-fidelity aerodynamic models.

The present thesis has two main objectives. The first objective is to assess the effect of the aerodynamic level of fidelity on steady aerodynamic and static aeroelastic computations typically performed in preliminary aircraft design. More specifically, the model offering the best trade-off between accuracy and computational cost will be identified. To this end, different models based on the linear and nonlinear flow equations are compared on rigid and flexible benchmark wings. Only steady computations are considered in the present work. The second objective is to develop a fast but accurate aerodynamic modeling tool which can be used to perform aerodynamic and aeroelastic computations for preliminary aircraft design. The tool will be based on the model identified through the first objective.

1.2 Aeroelastic modeling in preliminary aircraft design

The main purpose of static aeroelastic modeling in preliminary aircraft design is to predict the deformations of the lifting surfaces, which will in turn affect the aerodynamic loads acting on these surfaces. Static aero-structural computations have various applications. For example, as the wing is first designed in cruise conditions, the deflections yielded by such a computation are used to recover the wing shape in other conditions, such as on the ground or during a maneuver. Another and more recent application is aeroelastic tailoring, which was initially defined by Shirk and Hertz in 1986 as "*the embodiment of directional stiffness into an aircraft structural design to control aeroelastic deformation, static or dynamic, in such a fashion as to affect the aerodynamic and structural performance of that aircraft in a beneficial way*" [5]. Aeroelastic tailoring is particularly relevant when the structure is designed using orthotropic materials, such as composite materials, which have low weight and high resistance, and are becoming more and more widely used in aerospace applications.

Static aeroelastic modeling usually follows a process known as fluid-structure interaction computations. For a given flight condition and an initial wing shape, the aerodynamic loads are first computed using an aerodynamic model. The loads are then passed to a structural model, which calculates the wing deflections. The two models are coupled using a coupling scheme and the process is iterated until convergence is achieved.

1.2.1 Aerodynamic modeling

Aerodynamic models can be categorized according to their level of fidelity, *i.e.* their ability to properly represent the physics of a given flow. The highest level of fidelity consists in solving the Navier-Stokes equations and is referred to as Direct Numerical Simulation (DNS). Such computations require very fine meshes, because all scales of turbulence need to be resolved, hence making the computation extremely expensive. To lower the computational cost, the small turbulence scales can be filtered out and coarser grids can be used. This kind of computation is called Large Eddy Simulation (LES). While DNS and LES capture flow physics accurately and are suitable for studying specific benchmark cases, they are not used in preliminary aircraft design because of their prohibitive computational cost. Moreover, DNS and LES both capture the chaotic nature of turbulence, which is not of practical interest in the early design stages. The highest level of fidelity considered in such early design stages is therefore the Reynolds-Averaged Navier-Stokes (RANS) equations, in which all the turbulence is modeled instead of being resolved numerically. With recent increases in computational power and resources, RANS computations are now performed routinely in aircraft design. However, they remain expensive and their predictions might not be totally relevant, as the aircraft shape is usually not sufficiently refined in the early design stages. Consequently, lower levels of fidelity are often considered. To simplify the RANS equations, the viscosity of the fluid can be neglected. This is valid for the aeronautical flows considered in preliminary aircraft design, because the boundary layer, in which the viscous effects are predominant, is very thin and develops close to the body's surface. As a consequence, the rest of the flow can be considered as inviscid. An alternative to solving the RANS equations, and to taking viscous effects into account, is to use an inviscid model coupled to the boundary layer equations. These equations can be readily derived from the Navier-Stokes equations. The inviscid form of the Navier-Stokes equations is known as the Euler equations. The full potential equation is obtained by further assuming that the flow is isentropic and therefore irrotational. Finally, the full potential equation can be linearized to obtain the linear potential equation.

Five levels of fidelity are considered in the present work: the Reynolds-Averaged Navier-Stokes equations, the Euler equations, the full potential equation, on its own or corrected by the boundary layer equations, and the linear potential equation.

High-fidelity modeling

The unsteady Reynolds-Averaged Navier-Stokes equations can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}^{\mathrm{c}} - \nabla \cdot \mathbf{F}^{\mathrm{d}} = 0, \tag{1.2.1}$$

where $\mathbf{U}$ is the vector of the conservative flow variables defined as

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho \mathbf{u} \\ \rho e \end{bmatrix}. \tag{1.2.2}$$

The convective fluxes $\mathbf{F}^{\mathrm{c}}$ and the diffusive fluxes $\mathbf{F}^{\mathrm{d}}$ are defined as

$$\mathbf{F}^{\mathrm{c}} = \begin{bmatrix} \rho\mathbf{u} \\ \rho\mathbf{u}\otimes\mathbf{u} + p\mathbf{I} \\ \rho e\mathbf{u} + p\mathbf{u} \end{bmatrix}, \qquad \mathbf{F}^{\mathrm{d}} = \begin{bmatrix} . \\ \boldsymbol{\tau} \\ \boldsymbol{\tau}\cdot\mathbf{u} + \mu^{\star}c_p\nabla T \end{bmatrix} \tag{1.2.3}$$

where ρ is the density, $\mathbf{u}$ the velocity vector, p the pressure, e the total specific energy, c_p the specific heat capacity at constant pressure, and T the temperature. The stress tensor for a Newtonian fluid is given by

$$\boldsymbol{\tau} = \mu\left(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}} - \frac{2}{3}\mathbf{I}\nabla\cdot\mathbf{u}\right). \tag{1.2.4}$$

The total viscosity μ and $\mu^{\star}$ in Equations 1.2.3 and 1.2.4 can be expressed as

$$\begin{aligned} \mu &= \mu_{\mathrm{d}} + \mu_{\mathrm{t}}, \\ \mu^{\star} &= \frac{\mu_{\mathrm{d}}}{\mathrm{Pr}_{\mathrm{d}}} + \frac{\mu_{\mathrm{t}}}{\mathrm{Pr}_{\mathrm{t}}}, \end{aligned} \tag{1.2.5}$$

where Pr is the Prandtl number. The subscript d refers to dynamic quantities, which are properties of the fluid, while the subscript t refers to turbulent quantities, which are given by a turbulence model. In the present work, the Spalart-Allmaras model [6], commonly used and specifically developed for aeronautical flows, has been used. Computations using the Menter's $k-\omega$ Shear Stress Transport model [7] have also been performed and are given in appendix A. However, the Spalart-Allmaras model has been favored over Menter's due to its better rate of convergence for the considered cases. The system of equations needs to be closed with the state equations

$$\begin{aligned} e &= c_v T + \frac{1}{2}\|\mathbf{u}\|^2, \\ p &= \rho R T, \end{aligned} \tag{1.2.6}$$

where c_v is the specific heat capacity at constant volume and R is the ideal gas constant.

Medium-fidelity modeling

The Euler equations are the inviscid counterpart of the Navier-Stokes equations and are obtained by neglecting the diffusive fluxes, $\mathbf{F}^{\mathrm{d}}$ in Equation 1.2.1, yielding

$$\frac{\partial\mathbf{U}}{\partial t} + \nabla\cdot\mathbf{F}^{\mathrm{c}} = 0, \tag{1.2.7}$$

where the unknown vector $\mathbf{U}$ is defined by Equation 1.2.2, and the vector of convective fluxes $\mathbf{F}^{\mathrm{c}}$ is given in Equation 1.2.3.

The steady full potential equation assumes that the fluid is inviscid, and that the flow is steady and isentropic. The flow is then irrotational and the velocity derives from a potential ϕ such that,

$$\mathbf{u} = \nabla\phi, \tag{1.2.8}$$

and the mass conservation can therefore be cast into

$$\nabla \cdot (\rho \nabla \phi) = 0. \tag{1.2.9}$$

The density ρ is given by the isentropic flow relationship

$$\rho = \rho_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - |\nabla \phi|^2 \right) \right]^{\frac{1}{\gamma - 1}}, \tag{1.2.10}$$

where ρ_∞ is the freestream density, γ is the heat capacity ratio and M_∞ is the freestream Mach number. Note that the term $|\nabla \phi|$ in Equation 1.2.10, which is the magnitude of the total velocity, has been normalized by the freestream velocity. The entropy produced through a shockwave is related to the normal Mach number just upstream of that shock, M_n, using

$$\Delta S_\mathrm{E} = \mathcal{O}(M_n^2 - 1)^3. \tag{1.2.11}$$

Therefore, the isentropicity assumption restricts the use of the nonlinear potential equation to transonic flows with embedded weak shocks only. A common upper limit for the upstream normal Mach number is $M_n < 1.3$ [8]. Additionally, the Kutta condition must be enforced in order to allow a potential (irrotational) flow to generate aerodynamic loads. Mathematically, the pressure on the upper and lower sides of the trailing edge of a wing are imposed to be equal. Numerically, the potential is discontinuous across a wake extending from the trailing edge of any lifting body to the downstream boundary, and the Kutta condition is enforced by prescribing the continuity of the mass flux and velocity magnitude on both sides of the wake such that

$$\begin{aligned} [\![\rho \nabla \phi]\!] &= 0, \\ [\![|\nabla \phi|^2]\!] &= 0, \end{aligned} \tag{1.2.12}$$

where the double squared bracket indicates a jump through the wake surface. The exact implementation of this boundary condition depends on the method used to discretize the potential equation and on the grid type. Finite volume implementations of the Kutta condition have been proposed by Neel [9], Liegl [10] and Lyu et al. [11], while finite element formulations can be found in Nishida [12], Galbraith et al. [13] and Crovato et al. [14].

Low-fidelity modeling

The full potential equation can be transformed into an integral equation by integrating Equation 1.2.9 and using Green's third identity. The resulting expression can then be linearized, such that

$$\phi = \phi_\infty - \int_S \left[\nabla \phi \cdot \mathbf{n} K - \phi \mathbf{n} \cdot \nabla K \right] dS, \tag{1.2.13}$$

where ϕ_∞ is the freestream potential, K is a kernel function depending on the geometry and the freestream conditions, and $\mathbf{n}$ is the outward unit vector normal to the surface S of the geometry. The terms appearing under the integral in Equation 1.2.13 represent the linear part of the flow, and are modeled by sources and doublets, which are fundamental solutions of the equation.

Viscous-inviscid coupling

The boundary layer equations are derived by simplifying the Navier-Stokes equations to the two-dimensional boundary layer region using order of magnitude analysis. For high Reynolds numbers, the continuity, momentum and energy equations read,

$$\begin{aligned}
\frac{\partial \rho u}{\partial \xi} + \frac{\partial \rho v}{\partial \eta} &= 0, \\
u\frac{\partial u}{\partial \xi} + v\frac{\partial u}{\partial \eta} &= -\frac{1}{\rho}\frac{\partial p}{\partial \xi} + \frac{1}{\rho}\frac{\partial}{\partial \eta}\left(\mu \frac{\partial u}{\partial \eta}\right), \\
\frac{\partial p}{\partial \eta} &= 0, \\
u\frac{\partial h}{\partial \xi} + v\frac{\partial h}{\partial \eta} &= \frac{u}{\rho}\frac{\partial p}{\partial \xi} + \frac{\mu}{\rho}\left(\frac{\partial u}{\partial \eta}\right)^2 + \frac{1}{\rho}\frac{\partial}{\partial \eta}\left(\frac{\mu}{\mathrm{Pr}}\frac{\partial h}{\partial \eta}\right),
\end{aligned} \tag{1.2.14}$$

where ξ and η are the coordinates in the tangent and normal directions to the solid surface over which the boundary layer exists, and u and v are the velocity components along these two respective directions. The enthalpy h, is related to the other variables through the equation of state

$$h = \frac{\gamma}{\gamma - 1}\frac{p}{\rho}. \tag{1.2.15}$$

In practice, the boundary layer equations are not solved directly, but an integral formulation is used instead. A frequent choice consists in solving the integral momentum and the kinetic energy shape parameter equations,

$$\begin{aligned}
\frac{d\theta}{d\xi} + (2 + H - M_e^2)\frac{\theta}{u_e}\frac{du_e}{d\xi} &= \frac{C_f}{2}, \\
\theta\frac{dH^\star}{d\xi} + \left[2H^{\star\star} + H^\star(1 - H)\right]\frac{\theta}{u_e}\frac{du_e}{d\xi} &= 2C_D - H^\star\frac{C_f}{2}.
\end{aligned} \tag{1.2.16}$$

The shape parameter is defined as

$$H = \frac{\delta^\star}{\theta}. \tag{1.2.17}$$

The displacement and density thicknesses are defined as

$$\begin{aligned}
\delta^\star &= \int \left(1 - \frac{\rho u}{\rho_e u_e}\right) d\eta, \\
\delta^{\star\star} &= \int \frac{u}{u_e}\left(1 - \frac{\rho}{\rho_e}\right) d\eta.
\end{aligned} \tag{1.2.18}$$

The momentum and kinetic energy thicknesses are defined as

$$\begin{aligned}
\theta &= \int \frac{\rho u}{\rho_e u_e}\left(1 - \frac{u}{u_e}\right) d\eta, \\
\theta^\star &= \int \frac{\rho u}{\rho_e u_e}\left(1 - \frac{u^2}{u_e^2}\right) d\eta.
\end{aligned} \tag{1.2.19}$$

The following dependencies for the skin friction and dissipation coefficients, as well as for the kinetic energy and density shape parameters, are assumed in order to close the set of equations,

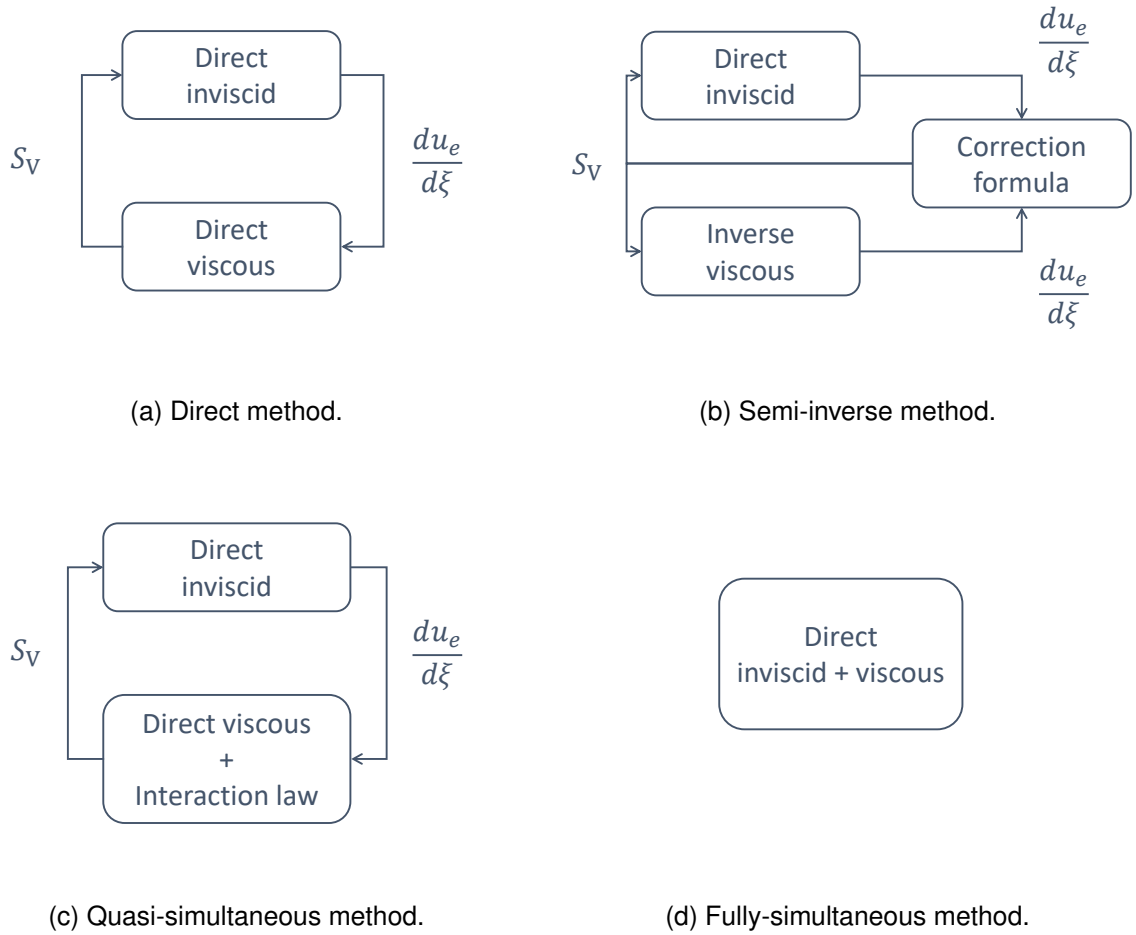
$$\begin{aligned} C_f &= C_f(H_k, M_e, Re_\theta), \\ C_D &= C_D(H_k, M_e, Re_\theta), \\ H^\star &= H^\star(H_k, M_e, Re_\theta), \\ H^{\star\star} &= H^{\star\star}(H_k, M_e), \end{aligned} \tag{1.2.20}$$

where the momentum thickness Reynolds number and the kinematic shape parameter are defined as

$$\begin{aligned} Re_\theta &= \frac{\rho_e u_e \theta}{\mu_e}, \\ H_k &= \frac{\int \left(1 - \frac{u}{u_e}\right) d\eta}{\int \frac{u}{u_e}\left(1 - \frac{u}{u_e}\right) d\eta}. \end{aligned} \tag{1.2.21}$$

In the above equations, note that the subscript e denotes inviscid variables at the edge of the boundary layer, and that all the integrals are performed over the boundary layer thickness. While the equations presented in this section are inherently two-dimensional, the viscous-inviscid interaction method can be extended to three dimensions. More details about the formulation can be found in several works by Drela and Mughal [15, 16, 17, 18].

Various techniques can be used to couple the viscous integral equations to an inviscid model. A direct coupling scheme cannot be used for aeronautical flows, since they can be partly detached. In such a case, the semi-inverse, quasi-simultaneous and fully-simultaneous coupling techniques are usually favored, since they are able to deal with both attached and detached flows, as demonstrated by Veldman [19]. In the semi-inverse method, the inviscid and viscous equations are solved iteratively by separate solvers and a correction is applied to couple the two solutions. In the quasi-simultaneous technique, the inviscid computation is first performed separately, and then the viscous equations are solved simultaneously with an interaction law, representing the inviscid equations. The process is iterated until convergence. Finally, the fully-simultaneous technique consists in coupling the inviscid and viscous equations in the same set, and solving them together. The different methods are schematically depicted in Figure 1.2.1. In Figures 1.2.1a to 1.2.1c, the term S_V denotes the effect of a viscous solution on an inviscid computation. Two general methods exist to compute this effect. The first is to thicken the body shape by the displacement thickness of the boundary layer, so that the momentum deficit generated by the presence of the boundary layer can be felt by the inviscid flow. The second is to compute a blowing velocity, based on the displacement thickness, and enforce it in place of the impermeability boundary condition on the body surface. Although the second method requires more computations, it is usually favored because it allows the geometry and the mesh to remain unchanged. More details about the coupling schemes can be found in Veldman [19] and Lock [20].



(a) Direct method.

(b) Semi-inverse method.

(c) Quasi-simultaneous method.

(d) Fully-simultaneous method.

Figure 1.2.1: Coupling schemes (adapted from Lock [20]).

1.2.2 Structural modeling

Two techniques for structural modeling are considered in the present work: a physical space-based approach and a modal approach. Neglecting internal damping, the equilibrium equations of a solid are obtained by balancing the inertial and internal forces in the solid with the external forces applied to it. The structural equations can be written as

$$\rho_{\mathrm{s}} \frac{d^2 \mathbf{u}_{\mathrm{s}}}{dt^2} - \nabla \cdot \boldsymbol{\sigma}_{\mathrm{s}} = \mathbf{f}_{\mathrm{s}}, \tag{1.2.22}$$

where ρ_{s} is the solid density, $\boldsymbol{\sigma}_{\mathrm{s}}$ is the stress tensor, $\mathbf{f}_{\mathrm{s}}$ are the external forces and $\mathbf{u}_{\mathrm{s}}$ are the displacements.

The displacements of the solid can also be expressed in the modal space by splitting them in a spatial and a time-dependent term,

$$\mathbf{u}_{\mathrm{s}} = \boldsymbol{\phi}_{\mathrm{s}}(x, y, z) \exp(\mathrm{i}\omega t), \tag{1.2.23}$$

where $\boldsymbol{\phi}_{\mathrm{s}}$ are the mode shapes of the solid depending solely on the spatial coordinates x, y,

z, and ω is a frequency of vibration. Injecting the modal decomposition into Equation 1.2.22 and solving the associated eigenvalue problem allows to recover the mode shapes and their associated modal frequencies. Then, noting that the energy related to the displacements is usually contained in the lowest frequency modes further allows to work with a reduced set of modal coordinates $\mathbf{q}_\mathrm{s}$, defined such that

$$\mathbf{q}_\mathrm{s} = \mathbf{\Phi}_\mathrm{s}^\mathrm{T} \mathbf{u}_\mathrm{s}, \tag{1.2.24}$$

where $\mathbf{\Phi}_\mathrm{s}$ is the modal matrix, containing the first mode shapes of the solid. By neglecting the time dependent terms, since only steady computations are considered, equation 1.2.22 can be further discretized into

$$\mathbf{K}_\mathrm{q} \mathbf{q}_\mathrm{s} = -\mathbf{f}_\mathrm{q}, \tag{1.2.25}$$

where $\mathbf{K}_\mathrm{q}$ is the modal stiffness matrix and $\mathbf{f}_\mathrm{q}$ is the vector of modal forces, obtained by multiplying the vector of forces by the mode shape matrix.

1.2.3 Aerostructural coupling

The aerodynamic and structural models can be coupled using a monolithic or a partitioned approach [21, 22]. In the monolithic approach, both the fluid and the structure are modeled within the same mathematical and numerical framework, and are solved by the same solver. While this approach allows to easily couple both physics, it lacks generality and modularity, since the solver has usually been implemented to deal with a given type of coupled physics. In the partitioned approach, the aerodynamic and the structural models are handled by different solvers, which are then coupled through an interface. This allows to take advantage of the efficient solution strategies available in the individual solvers and developed to target their specific physics, but requires an efficient communication procedure to be implemented in the interface. Since several different aerodynamic solvers are compared in the present work, the partitioned approach will be used.

The partitioned approach can be implemented using a Dirichlet-Neumann procedure, as described by Kuttler and Wall [23]. In this context, the loads at the fluid interface, $\mathbf{f}_\mathrm{f}^\mathrm{I}$, are obtained by applying a nonlinear Dirichlet operator $\mathcal{F}$ on a given fluid interface displacement $\mathbf{x}_\mathrm{f}^\mathrm{I}$,

$$\mathbf{f}_\mathrm{f}^\mathrm{I} = \mathcal{F}\left(\mathbf{x}_\mathrm{f}^\mathrm{I}\right). \tag{1.2.26}$$

Similarly, the displacement of the solid interface, $\mathbf{x}_\mathrm{s}^\mathrm{I}$, is related to the loads on this interface, $\mathbf{f}_\mathrm{s}^\mathrm{I}$, using a nonlinear Neumann operator $\mathcal{S}$, such that

$$\mathbf{x}_\mathrm{s}^\mathrm{I} = \mathcal{S}\left(\mathbf{f}_\mathrm{s}^\mathrm{I}\right). \tag{1.2.27}$$

The coupling of the fluid and the structure can be formulated as a fixed-point problem,

$$\mathbf{x}^\mathrm{I} = \mathcal{S}\left(\mathcal{F}^{-1}\left(\mathbf{x}^\mathrm{I}\right)\right) \Leftrightarrow \mathbf{x}^\mathrm{I} = \mathcal{T}\left(\mathbf{x}^\mathrm{I}\right), \tag{1.2.28}$$

where $\mathcal{T}$ is a nonlinear transfer operator. Several techniques exist to solve Equation 1.2.28, such as explicitly iterating over the fluid and solid problems, the Block Gauss-Seidel (BGS) algorithm [24] and the Interface quasi-Newton Inverse Least Squares algorithm originally proposed by Degroote et al. [25]. The BGS algorithm, used in the present work, is described in Figure 1.2.2. The interface displacement at a given iteration, $\mathbf{x}_n^{\mathrm{I}}$, is first used to update the fluid domain and its boundary conditions. The updated fluid variables, $\mathbf{U}_{n+1}^{\mathrm{f}}$, are subsequently computed. The loads on the interface, $\mathbf{f}_{n+1}^{\mathrm{I}}$, are then updated and used as new boundary conditions in the structural solver to compute the new displacement field, $\mathbf{U}_{n+1}^{\mathrm{s}}$. This process is repeated until convergence is reached. The convergence criterion is usually defined by using the difference either in the displacements or in the loads between two consecutive iterations.

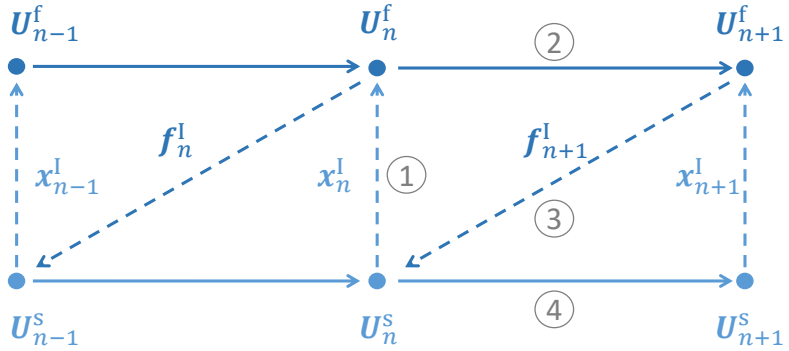


Figure 1.2.2: Block Gauss-Seidel algorithm (adapted from Wood et al. [24]).

1.3 Thesis overview

The present thesis is organized as follows. In chapter 2, the different levels of fidelity used for aerodynamic modeling in preliminary aircraft design are investigated using two benchmark wings. The aerodynamic pressure and loads distributions, as well as the computational cost of the different models, are analyzed with the aim of identifying the fastest method yielding consistent and reliable results. In chapter 3, the nonlinear potential equation and the methods for solving it are reviewed. Three of the main approaches for solving this equation are presented and discussed, namely the finite volume, finite element and field panel methods. The formulation and implementation of the field panel method are described in chapter 4. Computations are performed on various three-dimensional wings to validate the method. A finite element method is then developed and presented in chapter 5. The code is tested on several two and three-dimensional configurations. In chapter 6, the finite element code is first validated in the context of static fluid-structure interaction computations. Then, the different levels of fidelity used for aerodynamic modeling in aircraft design are compared in this context. The aerody-

namic loads distributions and the static deflections, as well as the computational cost of the different models, are analyzed. Chapter 7 summarizes and concludes the present dissertation, and suggests research directions for future work.

Chapter 2

Comparison of aerodynamic models

The effect of the different levels of fidelity on steady aerodynamic computations is assessed in the present chapter. Previous comparisons performed by various authors are first briefly reviewed, and the levels of fidelity presented in chapter 1 are then compared on two benchmark wings in terms of solution accuracy and computational cost.

2.1 State-of-the-art

Various comparison analyses have already been performed on rigid geometries. For example, Bhateley and Cox [26], Verhoff and O'Neil [27], and Rubbert and Saaris [28] used transonic small disturbance and linear potential theory to compute transonic flows over fighter configurations, and compared their results to nonlinear potential modeling or experimental data. In particular, Verhoff and O'Neil already suggested to resort to multi-fidelity modeling to extend transonic prediction capabilities by combining panel methods to nonlinear potential solvers. Flores et al. [29] compared full potential to Euler solvers using two-dimensional airfoils and showed that the nonlinear potential formulation was noticeably faster than the Euler formulation, for a similar accuracy in the integrated aerodynamic coefficients, as long as the shocks were weak. Klopfer and Nixon [30] further showed that adding a non-isentropic correction to the full potential formulation greatly improved the results for strong shocks. Several authors, such as Le Balleur [31], Melnik et al. [32], and Van Muijden et al. [33], also added an interactive boundary layer modeling capability to full potential solvers and were able to match experimental data. Validation of various full potential codes with respect to higher-fidelity data can be found in the survey work by Holst [34]. Drela et al. [35], Potsdam [36], and Aftosmis et al. [37] also extended various Euler solvers with viscous-inviscid calculations. The latter compared their results to both Reynolds-Averaged Navier-Stokes computations and experimental data.

Some authors also performed surveys to assess the capabilities and limitations of the different aforementioned aerodynamic levels of fidelity. For example, Jameson [38], and more recently Johnson et al. [39], regrouped and analyzed the comparison studies performed by various authors using different solvers. However, the computations are based on different geometries and are scattered across different years, which makes direct comparison difficult. Moreover, the emphasis is usually placed on model or methodology validation rather than on the tradeoff between accuracy and computational time. A systematic and extensive comparative study of all major aerodynamic modeling methods for transonic flow on the same benchmarks was

performed by Crovato et al. [40].

2.2 Methodology

In the present chapter, the RANS equations 1.2.1 and the Euler equations 1.2.7 are solved using SU2 [41, 42, 43], an open-source code for multiphysics simulations and design optimization. The equations are spatially discretized on an unstructured dual-grid using a finite volume method with a cell-vertex based approach and a second-order accurate Jameson-Schmidt-Turkel scheme [44]. The fluxes are reconstructed using a Green-Gauss procedure. The time dependent terms are discretized using an Euler implicit scheme, and steady state is reached through a time marching procedure. Different acceleration techniques can be used in SU2. For the Euler computations, the Courant-Friedrich-Levy number is set to 5 and a multigrid with a W pattern and 3 coarsening levels is used, while for the RANS computations, the CFL number is kept close to 1 and no multigrid is used. The solution is considered converged when the relative residual of the equations drops below 10^{-6}.

The full potential equation 1.2.9 is solved using Tranair [45, 46], a commercial software for aircraft design and optimization developed by NASA[1] and The Boeing Company[2] in the last two decades of the 20th century, and distributed by Calmar Research[3] since 2004. The equation is discretized using finite elements on a rectangular Octree Cartesian grid, which is refined automatically by the software using a solution adaptation procedure. The equation is solved with a quasi-Newton procedure combined with the Bank and Rose line search [47]. Tranair also offers the possibility to model the effect of the boundary layer on body surfaces by coupling the inviscid solution to an integral solution of the boundary layer equations. The coupling between the inviscid and viscous equations is performed using the fully-simultaneous and wall transpiration approaches, as described in chapter 1. Details about the formulation can be found in several works by Drela [15, 16, 18]. The solution is considered converged when the relative residual of the equations drops below 10^{-6}.

The integral form of the linear potential equation 1.2.13 is solved using Panair [48, 49] and NASTRAN [50, 51]. Panair is a high-order panel method developed at NASA during the eighties. The body is discretized using first-order source and second-order doublet panels, allowing Panair to account for both the thickness and camber of a body with relatively few panels. The impermeability boundary condition, applied at the center of each panel, allows to compute the singularity strengths from which the potential can be recovered. NASTRAN is distributed by MSC Software[4] and uses the doublet-lattice method to solve the integral potential equation. The mean plane surface of a lifting configuration is discretized into a flat sheet containing panels with constant doublet line segments at their quarter chords, and the impermeability condition is imposed at the three quarter-chord of each panel. As a result, the doublet-lattice

[1] https://www.nasa.gov/
[2] https://www.boeing.com/
[3] http://www.calmarresearch.com/NF/home.htm
[4] https://www.mscsoftware.com/

approach ignores the thickness and camber of the body but NASTRAN offers the possibility to apply corrections using geometric, numerical or experimental data [52, 53]. In the present work, the Euler solution obtained with SU2 is used to build the *FA2J* matrix required by NASTRAN to correct the doublet-lattice pressure loads. The computations are performed by means of the elastic trim analysis, also known as *SOL 144*.

The different aerodynamic models, their abbreviations and the corresponding software packages are summarized in Table 2.2.1.

Name	Solver	Equations
PAN	Panair	Linear potential
NAS	NASTRAN	Linear potential
NASC	NASTRAN	Linear potential corrected by Euler
TRN	Tranair	Full potential
SU2	SU2	Euler
TRNV	Tranair	Full potential and boundary layer
SU2V	SU2	Reynolds-Averaged Navier-Stokes

Table 2.2.1: Naming convention and equations solved in the present chapter.

The different models and methods are first compared on the Onera M6 wing, a standard transonic flow test case for which experimental data are available [54]. The solvers are then used to study the Embraer benchmark wing, which is provided by Embraer S.A.[5] and representative of a regional jet wing model used in preliminary aircraft design [40, 55]. Both wings are considered to be rigid.

2.3 Onera M6

The Onera M6 wing model is depicted in Figure 2.3.1 and its geometric parameters are given in Table 2.3.1. Note that a thin sharp trailing edge has been considered. Wind tunnel measurements are available and documented by Schmitt and Charpin [54] for several Mach numbers ranging from 0.7 to 0.92, and various angles of attack up to $6°$. The present simulations are set up according to test number 2308, *i.e.* with a Mach number $M = 0.839$ and an angle of attack $\alpha = 3.06°$, as this test is commonly used in the literature and that the flow conditions correspond to those typically encountered in transonic aircraft design.

[5] https://embraer.com/

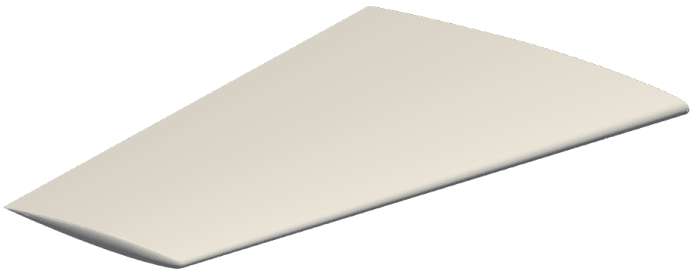


Figure 2.3.1: Onera M6 wing model.

Parameter	Value
Aspect ratio	3.8
Taper ratio	0.56
Sweep angle	$30°$
Root chord	805 mm
Semi-span	1196 mm

Table 2.3.1: Geometrical properties of the Onera M6 wing.

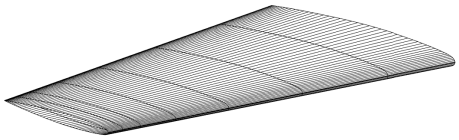
2.3.1 Mesh convergence

A surface grid made of rectangular surface panels is used in `Panair` and `NASTRAN`. In `Tranair`, the wing is enclosed in a box-shaped computational domain with boundaries placed 2 chord lengths away from the wing in the chordwise and normal directions, and a half-span length from the wingtip in the spanwise direction. The final grid, built automatically by `Tranair`, consists of hexahedral cells with a minimum size of $1/200$ of the chord at the shock and leading edge. The inviscid `SU2` grid is built using `gmsh` [56, 57]. It is based on an unstructured O-grid topology extending 50 root chords away from the wing. The mesh has a characteristic cell size of $1/200$ and $1/100$ of the local chord at the leading edge and at the trailing edge, respectively. The viscous `SU2` grid is built in `ANSYS ICEM` [58] using a multiblock structured C-grid topology extending 50 root chords away from the wing. The grid has 150, 75 and 25 hexahedra in the chordwise, normal and spanwise directions respectively. `SU2`'s implementation does not offer wall function. As a result, the grid must be resolved up to the wall, and the height of the first mesh cell y must be set so that $y^+ = \frac{\sqrt{\rho \tau_w} y}{\mu} \sim 1$, where ρ is the fluid density, τ_w are the wall shear stress and μ is the dynamic viscosity of the fluid. Experimental results consist of surface pressures measured at 271 locations distributed on 7 spanwise sections, and were gathered by Schmitt and Charpin [54].

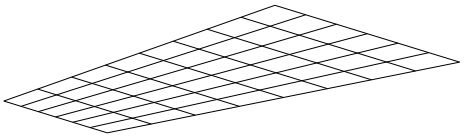
A convergence study was performed to find a suitable mesh size for each model. Table 2.3.2 shows the convergence for the lift and drag coefficients of the Onera M6 wing at $M = 0.839$ and $\alpha = 3.06^\circ$ for the different models. Note that NASC and TRNV use the same grid as NAS and TRN, respectively. Also, note that `NASTRAN` does not provide any value for the drag coefficient. In each case, the selected grid is the one for which the results did not change significantly when the number of cells was increased, that is the medium density grid indicated in Table 2.3.2. The final meshes are displayed in Figure 2.3.2.

Model	n. cells	C_L	C_D
PAN	360	0.246	0.0055
	1 000	0.247	0.0047
	1 440	0.247	0.0045
NAS	125	0.258	-
	500	0.248	-
	2 000	0.245	-
TRN	50 000	0.272	0.0137
	500 000	0.288	0.0111
	1 000 000	0.288	0.0111
SU2	140 000	0.281	0.0146
	510 000	0.286	0.0130
	1 200 000	0.287	0.0129
SU2V	300 000	0.257	0.0220
	1 500 000	0.272	0.0181
	3 000 000	0.270	0.0183

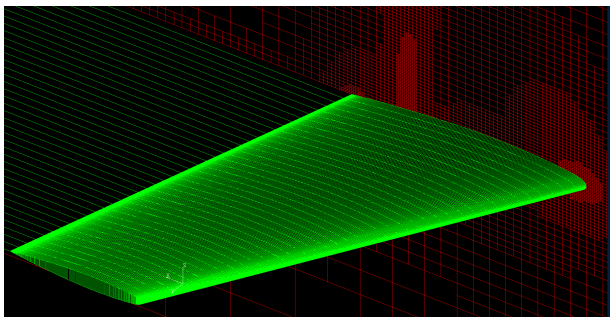
Table 2.3.2: Aerodynamic coefficients obtained on several meshes with the different numerical models for the Onera M6 wing at $M = 0.839$ and $\alpha = 3.06^\circ$.



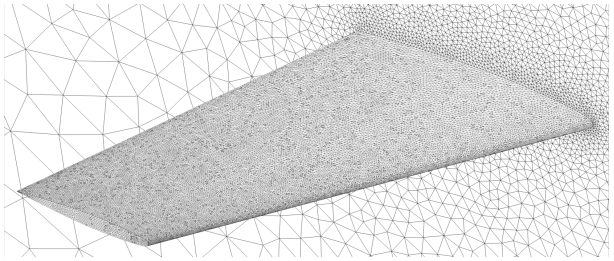
(a) Model PAN.



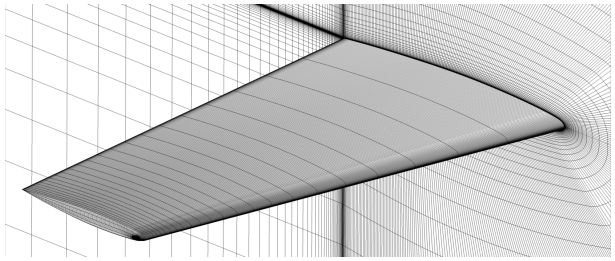
(b) Models NAS and NASC.



(c) Models TRN and TRNV.



(d) Model SU2.



(e) Model SU2V.

Figure 2.3.2: Computational grids used by the different models for the Onera M6 wing.

2.3.2 Aerodynamic loads

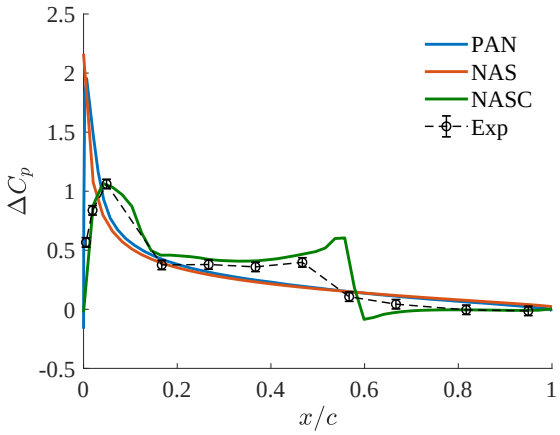
The aerodynamic load coefficients, obtained by integrating the forces on the surface of the wing, are given in Table 2.3.3. The reference point for the moment computation is taken at the leading edge of the root chord. The aerodynamic coefficients were not measured experimentally and are therefore not included in the table. As expected, the nonlinear inviscid models TRN and SU2 tend to predict higher lift and moment coefficients and a lower drag coefficient compared to the viscous models TRNV and SU2V because they ignore the boundary layer, which modifies the pressure distribution and produces shear forces. Compared to the nonlinear inviscid models, the linear models PAN and NAS slightly underestimate the lift and moment coefficients. PAN also strongly underestimates the drag coefficient, as it cannot compute the wave drag produced by shockwaves. When corrected by the Euler solution obtained using `SU2`, NASC predictions move closer to the results of the nonlinear models, except for the drag coefficient.

Model	C_L	C_D	C_M
PAN	0.247	0.0047	-0.181
NAS	0.248	-	-0.181
NASC	0.271	-	-0.201
TRN	0.288	0.0111	-0.212
SU2	0.286	0.0130	-0.212
TRNV	0.255	0.0161	-0.181
SU2V	0.272	0.0181	-0.196

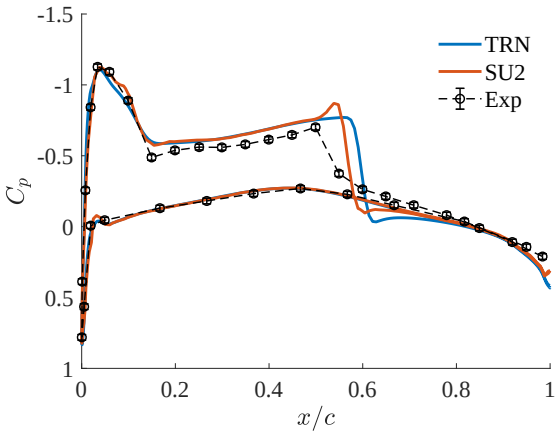
Table 2.3.3: Aerodynamic coefficients obtained by different levels of fidelity for the Onera M6 wing at $M = 0.839$ and $\alpha = 3.06°$.

Figure 2.3.3 shows the pressure distribution along the mean aerodynamic chord of the wing, located at 44% of the span, obtained using the different models and experimental data. Note that the difference in pressure distribution between the suction and pressure sides is used to compare linear models, since NAS and NASC are based on a lattice approach, hence not accounting for the wing thickness. Because of their underlying assumptions, PAN and NAS are unable to predict shocks and to represent the actual physics of transonic flows. Since NASC is corrected using an Euler calculation, it is the only linear approach that captures the shock. The nonlinear inviscid models TRN and SU2 are found to correctly represent the physics even though they predict a stronger shock when compared to the experimental results. Finally, the viscous models TRNV and SU2V give accurate pressure distribution predictions, although a small difference in the shock location and strength is still observed between them; the shock predicted by TRNV appears to be in slightly better agreement with the experimental measurements but this should not be taken as a general result.

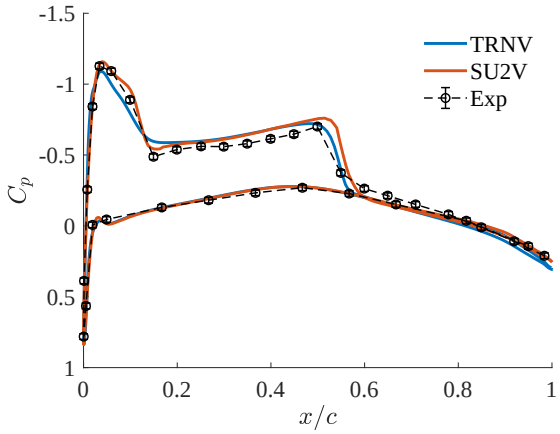
Figure 2.3.3b also shows that the different numerical solution procedures implemented in `SU2` and `Tranair` slightly affect the shock strength and location. Moreover, in the case of viscous models (Fig. 2.3.3c), the difference in boundary layer and turbulence modeling also affects the solution. This has a direct impact on the aerodynamic coefficients: SU2 tends to predict higher values of drag and moment compared to TRN while TRNV tends to underestimate the lift, moment and drag coefficients compared to SU2V, as illustrated in Table 2.3.3.



(a) Linear models



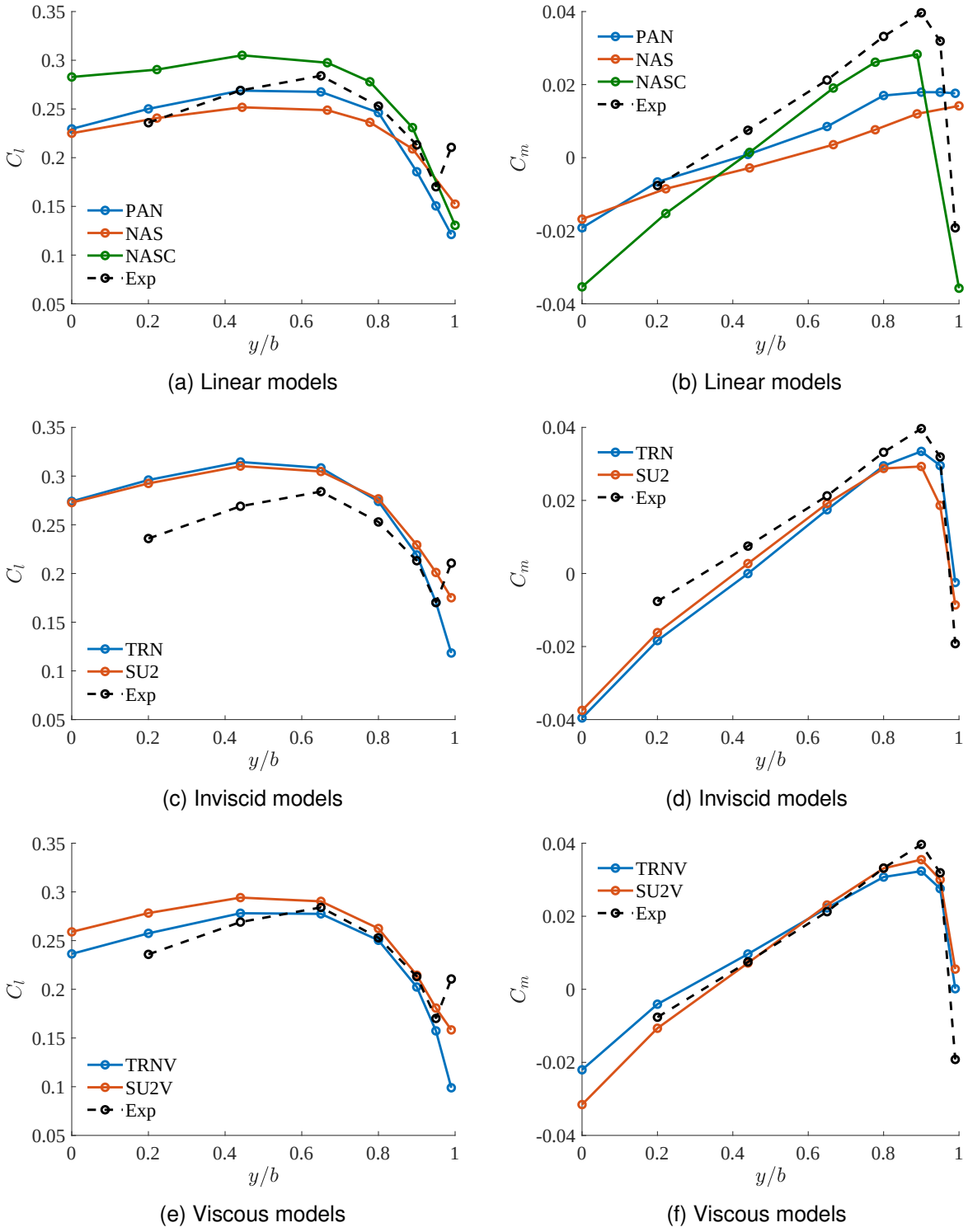
(b) Inviscid models



(c) Viscous models

Figure 2.3.3: Pressure distribution along the mean aerodynamic chord of the Onera M6 wing ($y/b = 0.44$) at $M = 0.839$ and $\alpha = 3.06°$ obtained from different levels of fidelity and compared to experimental data [54].

Figure 2.3.4 shows the distribution of the sectional lift and moment coefficients along the span of the Onera M6 wing, obtained by integrating numerically the pressure coefficient in the chord-wise direction. The sectional moment is computed around the local quarter-chord. The lift distribution predicted by the different solvers is similar to the experimental measurements, but there are differences in magnitude. In particular, as already noted in Table 2.3.3, the nonlinear inviscid models tend to predict higher lift coefficients for the same angle of attack. Both the inviscid and viscous nonlinear models predict sectional moment distributions that are similar to the experimental results. Since inviscid models ignore the boundary layer and predict stronger shocks, they tend to yield higher magnitudes for the moment coefficient. Finally, the moment distribution predicted by the linear models does not follow the same trend as the experimental data, except when corrected by a nonlinear solution.



(a) Linear models

(b) Linear models

(c) Inviscid models

(d) Inviscid models

(e) Viscous models

(f) Viscous models

Figure 2.3.4: Sectional aerodynamic loads along the span of the Onera M6 wing at $M = 0.839$ and $\alpha = 3.06°$ obtained from different levels of fidelity and compared to experimental data [54].

2.3.3 Computational performance

Models PAN, NAS, NASC, TRN and TRNV were run in serial on a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz), and models SU2 and SU2V were run on a cluster equipped

with Intel Xeon E5-2650 processors (2.0 GHz)[6]. The mesh sizes and computational times are given in Table 2.3.4. The linear models PAN, NAS and NASC are very fast since they require only one iteration to solve a scalar equation and need a small number of cells. Note that the computational time needed to compute the reference Euler solution required by NASC is not taken into account. On the other hand, the higher-fidelity models SU2 and SU2V need many iterations to solve five and six equations respectively, on a volume grid, which makes them slower. The medium-fidelity models TRN and TRNV require few Newton iterations to solve a scalar equation, for a typical runtime of five to ten minutes.

Table 2.3.4 shows that SU2V is quite slow. This is mainly due to the fact that using acceleration techniques such as Courant-Friedrich-Levy number adaptation or multigrid was not possible for this computation. Note that the goal of the present work is to identify the trends in the computational cost of the different models. Optimizing the numerical parameters of the different solvers could lead to a decrease in computational time.

Model	n. cells	n. threads	wall-clock time	cpu time
PAN	$1\,000$	1	10 s	10 s
NAS	500	1	20 s	20 s
NASC	500	1	20 s	20 s
TRN	$500\,000$	1	4 min	4 min
SU2	$510\,000$	12	14 min	3 h
TRNV	$500\,000$	1	8 min	8 min
SU2V	$1\,500\,000$	36	24 h	36 d

Table 2.3.4: Mesh size and computational time required by the different models for the Onera M6 benchmark case.

2.4 Embraer benchmark wing

In this section, the different models are compared on the Embraer benchmark wing [40, 55], which is more representative of a transport aircraft wing and has been designed and optimized for a transonic cruise flight at Mach 0.78. The wing model is depicted in Figure 2.4.1 and its geometrical parameters are provided in Table 2.4.1. Since no experimental data is available for this case, the numerical results will be compared to the highest aerodynamic level of fidelity considered in the present work: the Reynolds-Averaged Navier-Stokes equations. This wing is simulated in five different flight conditions: low-speed and high-speed maneuvers, and low-speed, nominal and high-speed cruise. For each calculation, the angle of attack is adjusted such that the resulting lift coefficient is equal to a prescribed lift coefficient, computed based on a typical transport aircraft weight. Note that this procedure is automatically handled by `Tranair`. Since the maneuver cases involve more complex flows, they are thoroughly analyzed in this section, while the results for cruise cases are given in appendix A and only briefly

[6]Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

presented here. For the low-speed maneuver, the Mach number is $M = 0.50$, the prescribed lift coefficient is $C_L = 0.80$ and the altitude is 6500 ft, while for the high-speed maneuver, the Mach number is $M = 0.78$, the prescribed lift coefficient is $C_L = 0.60$ and the altitude is 27000 ft. Note that the computations are carried out using the pressures and temperatures defined by International Standard Atmosphere at the given altitudes.

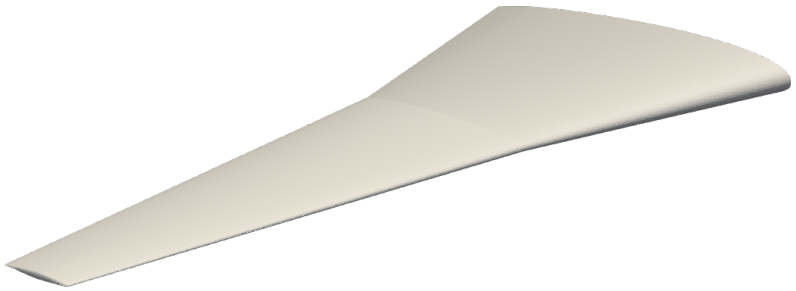
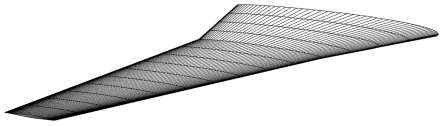


Figure 2.4.1: Embraer Benchmark wing model.

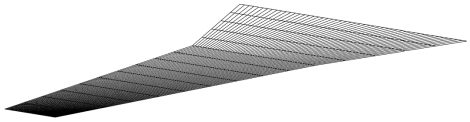
Parameter	Value
Aspect ratio	10
Taper ratio	0.28
Sweep angle	$26°$
Dihedral angle	$5°$

Table 2.4.1: Geometrical properties of the Embraer wing.

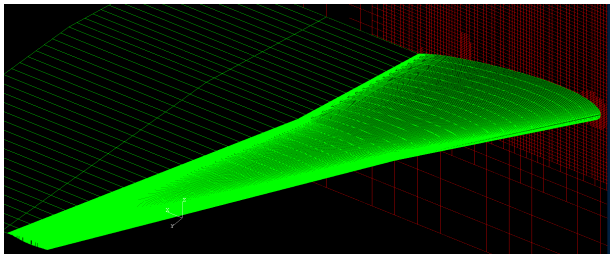
The computational domains for the different solvers are built in the same way as for the Onera M6 wing, but the grid sizes differ. Again, a convergence study was performed for each model. PAN and NAS/NASC have been discretized with $1\,400$ and 800 surface panels respectively. The final grids of TRN and TRNV consist of $500\,000$ hexahedral cells, with a minimum cell size of $1/200$ of the chord at the shock and leading edge. The unstructured grid used for SU2 contains 1.3 million tetrahedra with characteristic cell sizes of $1/200$ and $1/100$ of the local chord at the leading edge and at the trailing edge, respectively. Finally, the grid for SU2V contains 150, 75 and 25 cells in the chordwise, normal and spanwise direction respectively, for a total of 1.5 million hexahedra. The height of the first mesh cell is set so that $y^+ \sim 1$ for each flight condition. The final meshes are illustrated in Figure 2.4.2.



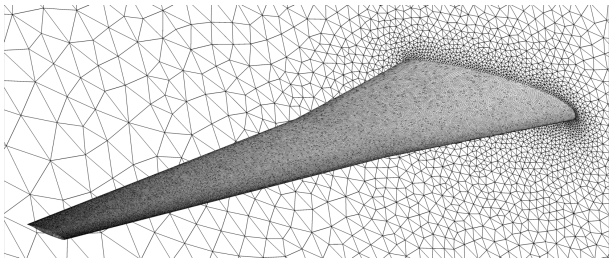
(a) Model PAN.



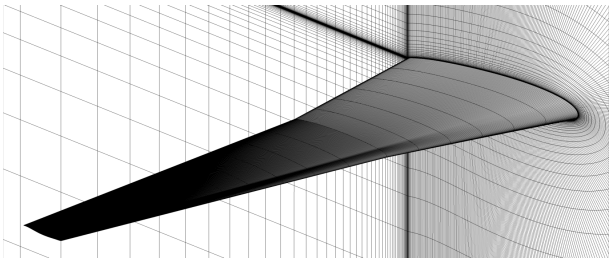
(b) Models NAS and NASC.



(c) Models TRN and TRNV.



(d) Model SU2.



(e) Model SU2V.

Figure 2.4.2: Computational grids used by the different models for the Embraer benchmark wing.

2.4.1 Aerodynamic loads

The computed angle of attack of the wing and the resulting aerodynamic load coefficients, obtained by integrating the forces on the surface of the wing, are given in Tables 2.4.2a and 2.4.2b for low-speed and high-speed maneuvers, respectively. TRNV and SU2V require a higher angle of attack to reach the same target lift than the inviscid models, and they also predict higher values for the drag coefficient. At high speed, the viscous models predict a lower moment coefficient than the inviscid models, while it is comparable at low speed. Linear models fall into two categories. On the one hand, PAN, which makes use of the full geometry of the wing, predicts a similar angle of attack and a lower value of the drag coefficient when compared to TRN and SU2. On the other hand, NAS is based on a flat lattice geometry and predicts a significantly higher angle of attack. Predictably, when the pressure correction calculated from the Euler solution is used, NASC predicts almost the same results as the nonlinear solvers.

Model	α	C_L	C_D	C_M
PAN	$+2.7$	0.80	0.0233	-1.069
NAS	$+8.6$	0.80	-	-0.973
NASC	$+3.2$	0.80	-	-1.041
TRN	$+3.1$	0.80	0.0243	-1.034
SU2	$+2.8$	0.80	0.0252	-1.055
TRNV	$+3.6$	0.80	0.0310	-1.025
SU2V	$+3.8$	0.80	0.0317	-1.018

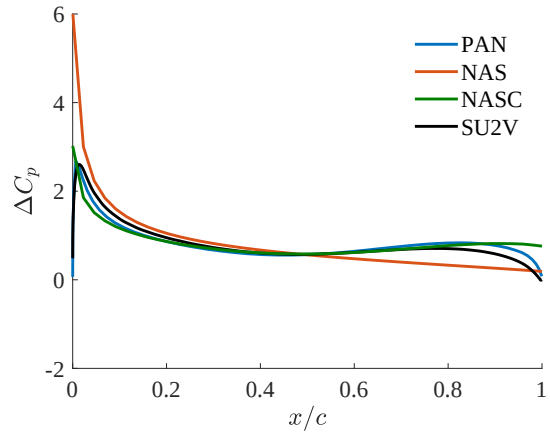
(a) $M = 0.50$ and $C_L = 0.80$.

Model	α	C_L	C_D	C_M
PAN	-0.5	0.60	0.0136	-0.866
NAS	$+5.3$	0.60	-	-0.739
NASC	-1.1	0.60	-	-0.872
TRN	-0.9	0.60	0.0159	-0.857
SU2	-0.9	0.60	0.0167	-0.866
TRNV	$+0.2$	0.60	0.0241	-0.815
SU2V	$+0.4$	0.60	0.0244	-0.819

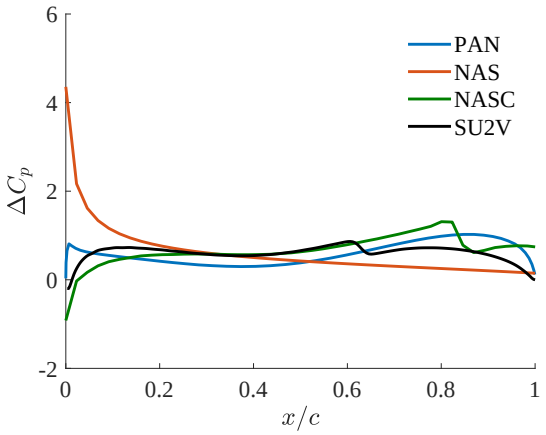
(b) $M = 0.78$ and $C_L = 0.60$.

Table 2.4.2: Aerodynamic coefficients obtained from the different levels of fidelity for the Embraer benchmark wing for low and high-speed maneuvers.

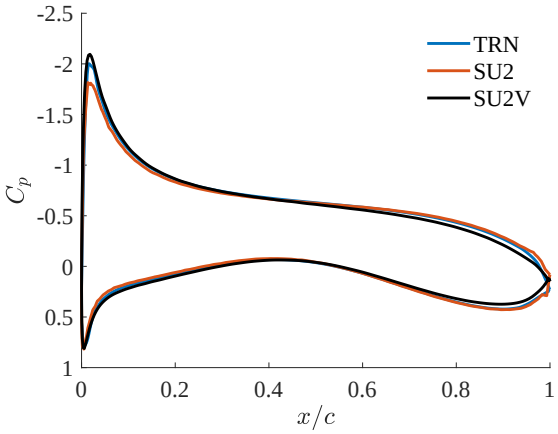
Figure 2.4.3 shows the pressure distribution along the mean aerodynamic chord of the Embraer benchmark wing, located at 42% of the span, at low and high speeds. At low speed, the various models predict a similar pressure difference to the one predicted by SU2V, except for NAS, which strongly overpredicts the pressure peak at the leading edge. This is mostly due to the high angle of attack needed to achieve the target lift coefficient. Since NASC is corrected by an Euler solution that allows to account for wing camber, the resulting difference in pressure is comparable to that calculated by SU2V. Similar conclusions can be drawn at higher speeds. Results obtained with PAN are similar to those obtained by SU2V, except at the shock and pressure peak locations. Since PAN and NAS solve linear equations, they cannot predict the shock. However, using the correction method implemented in `NASTRAN` allows NASC to take the shock into account, except that it is stronger and located further downstream. This is consistent, since the correction comes from an Euler calculation. The same is true for the results obtained from the inviscid nonlinear solvers, which also feature a reduced pressure peak compared to the SU2V prediction. TRNV, found to be as accurate as SU2V in the Onera M6 case, predicts a stronger shock and a slightly higher drag coefficient in the present case.



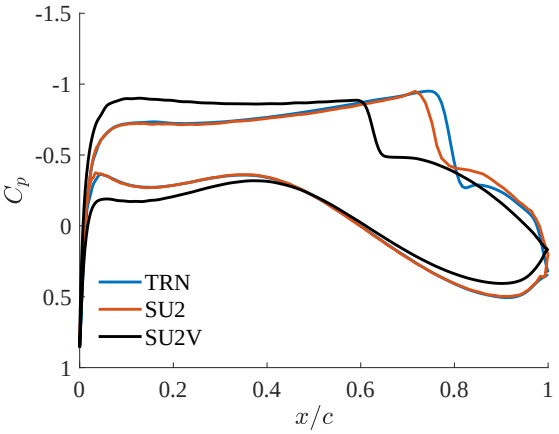
(a) Linear models ($M = 0.50$, $C_L = 0.80$).



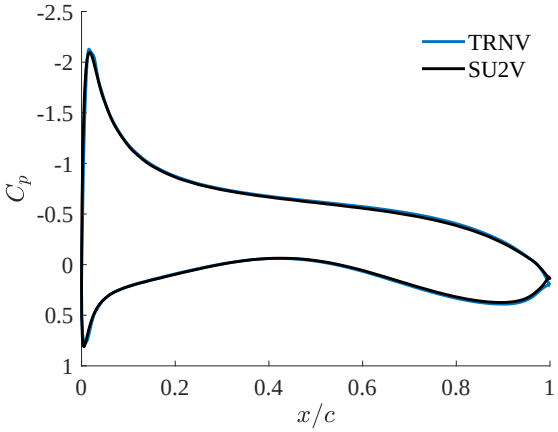
(b) Linear models ($M = 0.78$, $C_L = 0.60$).



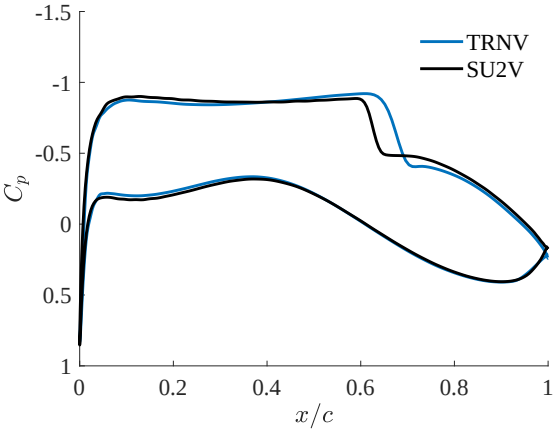
(c) Inviscid models ($M = 0.50$, $C_L = 0.80$).



(d) Inviscid models ($M = 0.78$, $C_L = 0.60$).



(e) Viscous models ($M = 0.50$, $C_L = 0.80$).

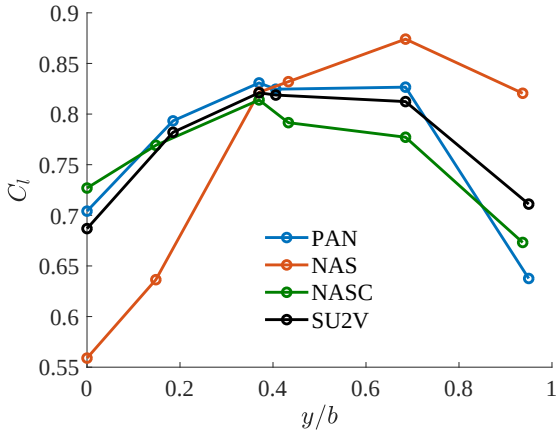


(f) Viscous models ($M = 0.78$, $C_L = 0.60$).

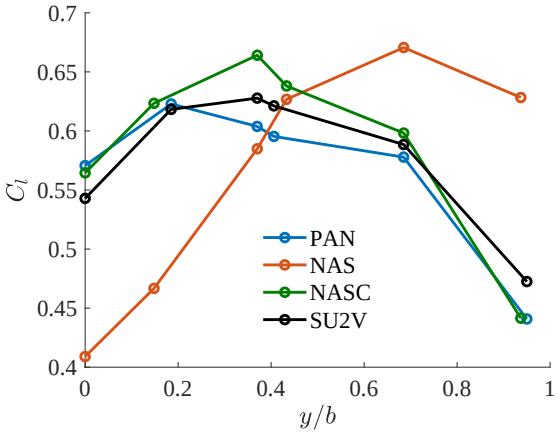
Figure 2.4.3: Pressure distribution along the mean aerodynamic chord of the Embraer benchmark wing ($y/b = 0.42$) for low and high-speed maneuvers obtained from the different levels of fidelity.

Figure 2.4.4 shows the sectional lift coefficient distribution along the span of the Embraer benchmark wing at low and high speeds. In this case, where the lift coefficient of the wing is prescribed, the lift distributions predicted by the different models are similar, except for NAS,

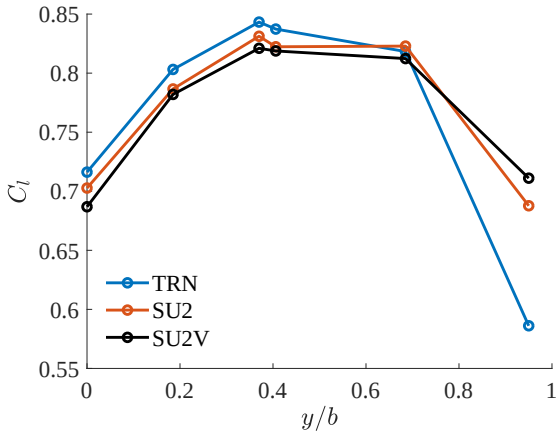
which yields highly inaccurate results. It should be recalled that the Embraer wing is cambered while the Onera M6 is not. NASTRAN does not include camber in its calculation by default; a camber or a pressure correction, or both, must be applied. The pressure correction used here improves the predictions significantly, as shown by the NASC results.



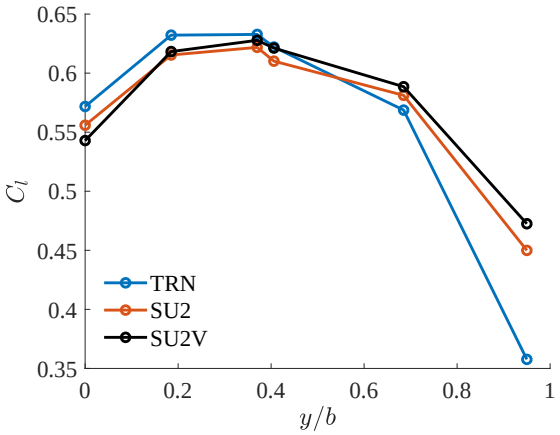
(a) Linear models ($M = 0.50$, $C_L = 0.80$).



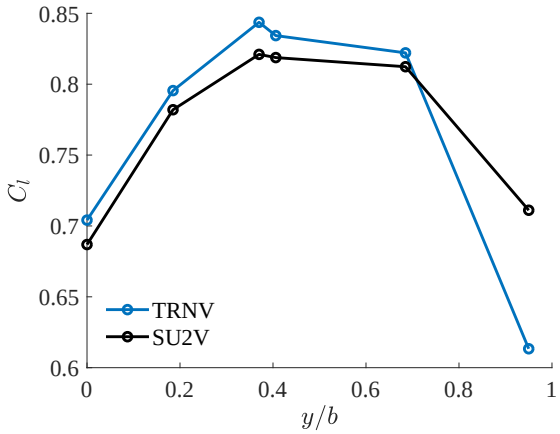
(b) Linear models ($M = 0.78$, $C_L = 0.60$).



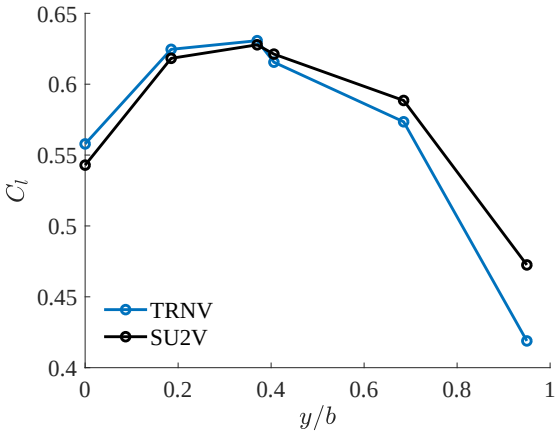
(c) Inviscid models ($M = 0.50$, $C_L = 0.80$).



(d) Inviscid models ($M = 0.78$, $C_L = 0.60$).



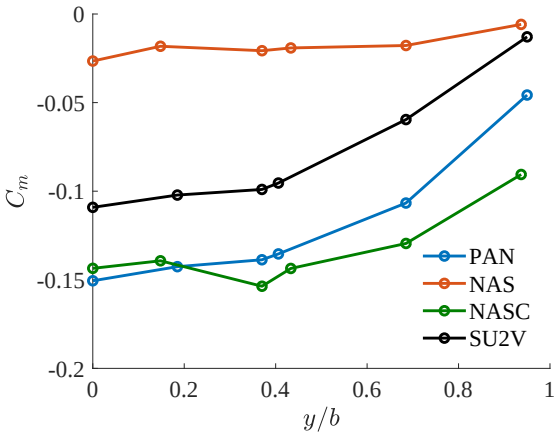
(e) Viscous models ($M = 0.50$, $C_L = 0.80$).



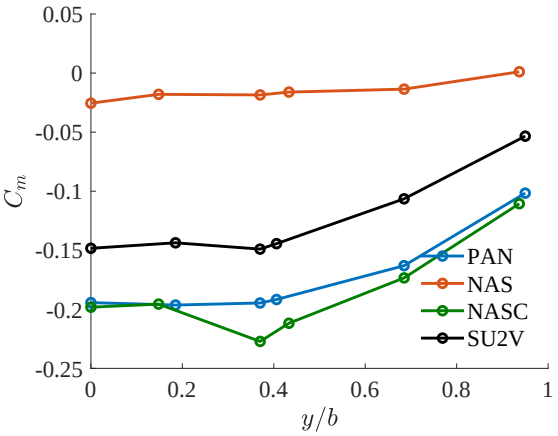
(f) Viscous models ($M = 0.78$, $C_L = 0.60$).

Figure 2.4.4: Sectional aerodynamic lift coefficient along the span of the Embraer benchmark wing for low and high-speed maneuvers obtained from the different levels of fidelity.

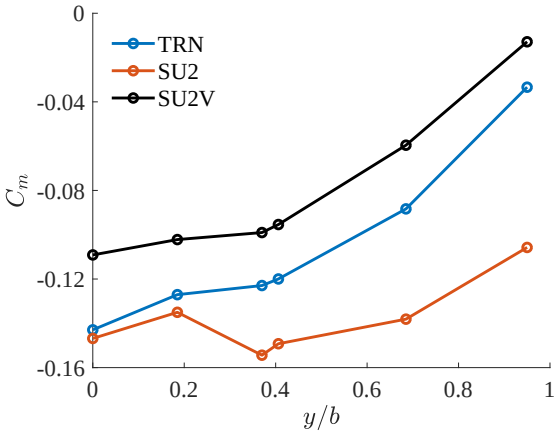
Figure 2.4.5 shows the sectional moment coefficient distribution along the span of the Embraer benchmark wing at low and high speeds. The moment distributions predicted by the various models are similar, with the following exceptions. Firstly, as in the Onera M6 case, inviscid models tend to predict moment coefficients with higher magnitude. Secondly, NAS yields highly inaccurate results. Again, the pressure correction used in NASC improves the predictions. Thirdly, SU2 overpredicts the dip in the moment distribution located at the kink of the wing ($y/b = 0.37$) at low speed, while PAN does not capture it at high speed.



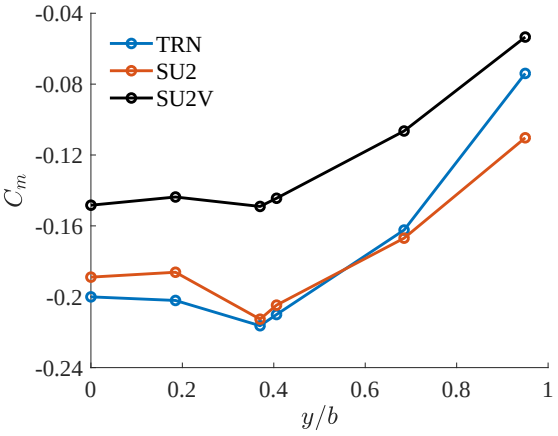
(a) Linear models (Mach 0.50, $C_L = 0.80$).



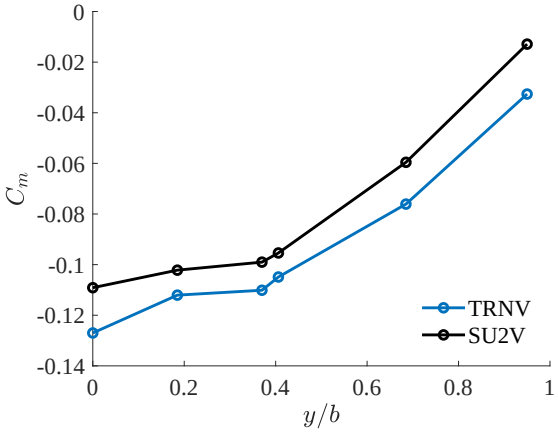
(b) Linear models (Mach 0.78, $C_L = 0.60$).



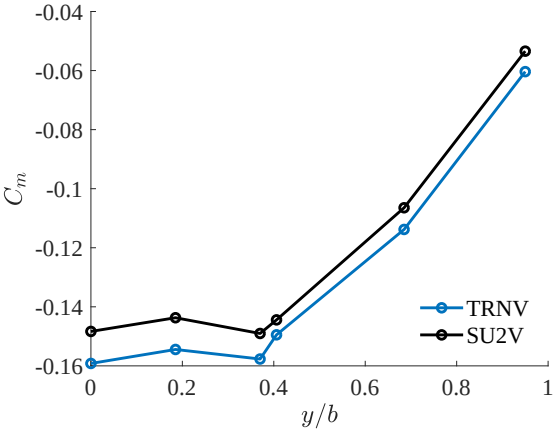
(c) Inviscid models (Mach 0.50, $C_L = 0.80$).



(d) Inviscid models (Mach 0.78, $C_L = 0.60$).



(e) Viscous models (Mach 0.50, $C_L = 0.80$).



(f) Viscous models (Mach 0.78, $C_L = 0.60$).

Figure 2.4.5: Sectional aerodynamic moment coefficient along the span of the Embraer benchmark wing for low and high-speed maneuvers obtained from the different levels of fidelity.

2.4.2 Computational performance

The same computers were used as for the Onera M6 wing. The mesh size and the computational time are given in Tables 2.4.3a and 2.4.3b. Again, the time needed to compute the reference Euler solution required by NASC is not displayed in the corresponding row. Compared

to the Onera M6 wing, the Embraer wing has a higher aspect ratio and unstructured meshes need to be denser to achieve the same resolution. This has an impact on the runtime: SU2 is now about 60 times slower than TRN. As in the Onera M6 case, TRNV is significantly faster than SU2V. Overall, the linear models PAN, NAS and NASC remain very fast, while higher-fidelity SU2 and SU2V based on the Euler and RANS equations are significantly slower. The full potential models TRN and TRNV offer a good tradeoff between accuracy and computational time.

Model	n. cells	n. threads	wall-clock time	cpu time
PAN	$1\,400$	1	10 s	10 s
NAS	800	1	25 s	25 s
NASC	800	1	25 s	25 s
TRN	$500\,000$	1	5 min	5 min
SU2	$1\,300\,000$	12	25 min	5 h
TRNV	$500\,000$	1	12 min	12 min
SU2V	$1\,500\,000$	60	48 h	120 d

(a) $M = 0.50$ and $C_L = 0.80$.

Model	n. cells	n. threads	wall-clock time	cpu time
PAN	$1\,400$	1	10 s	10 s
NAS	800	1	25 s	25 s
NASC	800	1	25 s	25 s
TRN	$500\,000$	1	5 min	5 min
SU2	$1\,300\,000$	12	30 min	6 h
TRNV	$500\,000$	1	12 min	12 min
SU2V	$1\,500\,000$	60	48 h	120 d

(b) $M = 0.78$ and $C_L = 0.60$.

Table 2.4.3: Mesh size and computational time required by the different models for the Embraer benchmark case.

2.4.3 Cruise flight conditions

The aerodynamic load distributions predicted by the different models used to compute the flow over the Embraer benchmark wing in the low-speed, nominal and high-speed cruise conditions, as well as their computational cost, are available in appendix A. The Mach numbers corresponding to these three flight conditions are $M = 0.70$, $M = 0.78$ and $M = 0.89$, and the corresponding prescribed lift coefficients are $C_L = 0.58$, $C_L = 0.47$ and $C_L = 0.39$, respectively.

In the low-speed cruise case, the freestream Mach number is small and the flow remains subsonic over the whole wing. As a result, the linear models are in good agreement with the nonlinear inviscid models, except for NAS, which neglects the camber of the wing. Compared to inviscid models, viscous models predict a higher pressure peak at the leading edge, and yield lower magnitudes of the moment coefficient, due to the presence of the boundary layer. For this case without shocks, the computational cost of the nonlinear models is lower than that for flow solutions in which shocks are present.

In the nominal cruise case, the viscous models predict a weak shock. Note that this is *by de-*

sign, as the wing shape has been optimized for this particular flight condition. On the other hand, nonlinear inviscid models predict a stronger shock, located downstream, which noticeably changes the pressure distribution, as well as the angle of attack needed to achieve the prescribed lift coefficient. The predictions of the linear models differ even more, since they do not predict the shock. This has a noticeable impact on the lift and moment distributions, especially near the kink of the wing. For this case with a weak shock, the computational cost of all the different models remains low.

The discrepancies between the models greatly increase as the freestream Mach number rises to $M = 0.89$. In this high-speed cruise condition, strong shocks are observed on both the suction and pressure sides of the wing. In such cases, there is strong interaction between the shock and the boundary layer, and neglecting the viscosity does not yield accurate and reliable results. Although both the full potential and the viscous-inviscid coupling formulations should break down for such transonic and partly separated flows, `Tranair` (TRNV) still converges and yields predictions comparable to RANS (SU2V) solutions obtained from `SU2`. For this case with strong shocks, the computational cost of the different models increases significantly, particularly that of TRNV.

2.5 Discussion

For both test cases considered in the present chapter, the presence of the shock and the boundary layer affected the results significantly. At high speed, the shock was found to change the behavior of the pressure distribution and to impact the lift, drag and moment coefficients. Furthermore, at both low and high speeds, the presence of the boundary layer was found to decrease the lift, and impacted the angle of attack such that it needed to be about one degree higher to reach a prescribed lift coefficient. At high speed, the boundary layer was also found to affect both the shock location and its strength, which in turn affected the aerodynamic coefficients. Moreover, accounting for the fluid viscosity introduces shear stresses and slightly modifies pressure forces, which significantly increases the magnitude of the drag coefficient. Results also show that using lattice methods, which do not account for the wing thickness and camber, has a significant impact on the solution. Correcting such methods with a nonlinear solution improves all results except for the drag, but the computational time becomes slightly higher than the one needed to obtain the nonlinear solution.

The results presented up to this point demonstrate that, in the presence of shocks, linear methods predict the lift with reasonable accuracy but underestimate the drag and the pitching moment and yield nonphysical pressure distributions. They are therefore not suited for transonic aerodynamic computations and optimization in preliminary aircraft design. On the other hand, the Euler and Reynolds-Avergared Navier-Stokes equations correctly capture the physics but have a high computational cost. Full potential models are found to give reliable results for a moderate computational cost. More particularly, the full potential equation was found to predict results comparable to the Euler equations for less than a tenth of the computational cost. When

coupled to the boundary layer equations, the full potential computations accuracy were similar to RANS results for less than a thousandth of the computational cost.

Since most aircraft fly in the transonic regime, the presence of shocks is very likely. Even though linear models are able to yield reasonable estimates of the lift distribution, they fail to give a good prediction for the drag and pitching moment. Moreover, neglecting the shock in transonic wing design could lead to badly shaped wings. Comparing the different nonlinear equations showed that the full potential model accuracy is similar to that of higher-fidelity models, but at a much lower cost, provided that shocks are not too strong. In practice, strong shocks are not desirable as they cause a dramatic increase in drag, and thus in fuel consumption. Consequently, modern transport aircraft wings are designed so that only weak shocks occur in nominal cruise conditions. As such, the flow isentropicty assumption underlying the full potential equation is not a critical limitation. The full potential formulations and their solution techniques will therefore be investigated in chapter 3.

Chapter 3

The full potential equation

In the present chapter, the different potential formulations and the numerical methods to solve them will be investigated. Attention will also be drawn to the main challenges that arise when solving these equations. Traditional field methods, such as the finite difference/element/volume methods, will first be reviewed, as they constitute the main techniques used to solve the full potential equation. The field panel method will then be briefly introduced and reviewed.

3.1 Potential formulations

3.1.1 Equations and solution methods

The different steady potential formulations can be written in conservative form,

$$\nabla \cdot \mathbf{F} = 0. \tag{3.1.1}$$

In the case of the full (nonlinear) potential equation, the flux vector reads,

$$\mathbf{F} = \rho \nabla \phi, \tag{3.1.2}$$

where ϕ is the total velocity potential, and ρ is the density, depending on the velocity and given by Eq. 1.2.10. Combining Equations 3.1.1 and 3.1.2 gives the full potential equation 1.2.9, which can be rearranged in the form of a Poisson equation, such that the nonlinear terms are grouped on the right-hand side,

$$\begin{aligned} &\nabla \cdot (\nabla \phi) = \sigma, \\ &\sigma = \frac{1}{\rho} \nabla \rho \cdot \nabla \phi. \end{aligned} \tag{3.1.3}$$

The full potential equation can also be cast into its non-conservative form,

$$(a^2 - u^2)\phi_{xx} + (a^2 - v^2)\phi_{yy} + (a^2 - w^2)\phi_{zz} - 2uv\phi_{xy} - 2uw\phi_{xz} - 2vw\phi_{yz} = 0, \tag{3.1.4}$$

where a is the local speed of sound and u, v and w are the local velocity components. Equation 3.1.4 can further be simplified by considering that lifting bodies induce small disturbances compared to the mean flow. The total potential is first decomposed into a freestream potential, ϕ_∞, and a perturbation potential assumed to be small compared to the total potential. The

steady Transonic Small Disturbance (TSD) equation is then obtained by neglecting small terms compared to the mean flow,

$$\left[1 - M_\infty^2 - M_\infty^2 \left(\gamma + 1\right) \frac{\varphi_x}{u_\infty}\right] \varphi_{xx} + \varphi_{yy} + \varphi_{zz} = 0, \tag{3.1.5}$$

where $\varphi = \phi - \phi_\infty$ is the perturbation potential , M_∞ is the freestream Mach number, u_∞ is the freestream velocity along the x direction and γ is the specific heat ratio. The non-conservative form of the full potential and the TSD equations are not the main focus of the present work, but are worth attention since many breakthroughs in transonic flow computation were first achieved using these formulations.

Partial differential equations can be solved with various numerical methods, usually categorized as boundary element or field methods. The oldest field method is the finite difference method, in which the equations are discretized on a set of points in the field around the geometry. Even though the finite difference method is simple in essence, it is severely limited by the grid generation process since the points should form a structured stencil. The finite element and finite volume methods were developed later on. Rather than being discretized directly on a grid, the equations are discretized on elements or volumes representing some relatively small portions of the field. These elements and volumes are then locally mapped to a simpler computational space, in which the computations are actually carried out. These methods thus circumvent the limitation of the finite difference method, since the grid no longer needs to be structured. Complex geometries can then be studied more easily. If the equations are linear, a cheaper alternative consists in using a boundary element method. In this approach, only the boundary of interest is discretized. Rather than solving the equations directly, elementary solutions (*i.e.* singularities) are superimposed on the surface of the geometry and are set to enforce its impermeability. In the case of aerodynamic modeling, this technique is usually referred to as a panel method. Its cost effectiveness makes it one of the standard tools in preliminary aircraft design, even today. In an effort to model flow nonlinearities while retaining the low computational cost of the boundary element method, researchers developed the field panel method, an extension of the panel method solving the full (nonlinear) potential equation. The solution procedure is the same as in a panel method, except that a non-conforming Cartesian grid is added around the geometry. The right-hand side of Equation 3.1.3 is then computed in the cells of this Cartesian grid by means of traditional finite differences. This term acts as a source term representing compressibility and allows the modeling of nonlinear flows. The field panel method can be considered as a middle way between a field technique and a boundary element method. More details about the formulation will be given in chapter 4.

3.1.2 Main challenges

Two challenges naturally arise when solving a potential formulation. Firstly, a potential flow is irrotational and cannot produce loads, which is exactly what engineers are trying to predict. A potential formulation thus has an infinite number of solutions, and an additional condition should be enforced. The Kutta condition is based on the physical observation that real fluids

are viscous and cannot turn around sharp corners, such as the trailing edge of a wing. There are many formulations of the Kutta condition, such as finite fluid velocity at the trailing edge, zero vorticity at the trailing edge, equality of pressures on the suction and pressure sides at the trailing edge, etc. The second challenge is the prediction of transonic flows. When the flow is subsonic, the pressure waves inside the fluid propagate in every direction. However, when the flow is supersonic, they can only propagate downstream. This change in the physics can be clearly seen mathematically in the non-conservative form of the full potential equation. By computing the characteristics of Equation 3.1.4, it can be demonstrated that the nature of the equation switches from elliptic to hyperbolic as the signs of the leading terms switch from positive to negative, which occurs when the flow becomes supersonic. The interested reader is directed to Holst's work [34] for a complete mathematical development. This change in the physical nature of the flow and mathematical nature of the equation must be reflected in the numerical implementation. Over the years, several approaches have been developed to handle this challenge, as will be illustrated in the next section.

3.2 Field potential solvers

This section is mainly based on Holst's excellent literature review on nonlinear potential methods [34]. It briefly describes the early work done on the nonlinear potential equation during the seventies and eighties, as well as the major breakthroughs that made the full potential model evolve into a mature technology.

3.2.1 Early work on the transonic small disturbances equation

The effort to solve transonic flows started in the late sixties with the TSD equation 3.1.5 as a model problem. The first major breakthrough was the "physics-dependent" differentiation, proposed by Murman and Cole in 1971 [59]. They implemented an algorithm reflecting the change in the physical and mathematical nature of the equation between the subsonic and supersonic regions of the flow. The algorithm switches from central differencing in subsonic regions to upwind differencing in supersonic regions. This allowed the computation of transonic flows, which was impossible before.

In two dimensions[1], the flux vector in Equation 3.1.1 for the steady TSD equation reads

$$\mathbf{F} = \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} (1 - M_\infty^2)\varphi_x - \frac{\gamma-1}{2} M_\infty^2 \varphi_x^2 \\ \varphi_y \end{bmatrix}. \tag{3.2.1}$$

The equation can then be rewritten following the Murman and Cole algorithm as

$$\frac{\bar{f}_{i+1/2,j} - \bar{f}_{i-1/2,j}}{\Delta x} + \frac{g_{i,j+1/2} - g_{i,j-1/2}}{\Delta y} = 0, \tag{3.2.2}$$

where i and j denote the position on the grid, such that $x = i\Delta x$ and $y = j\Delta y$, and where the

[1] All the developments will be particularized to two dimensions for conciseness.

flux $\bar{f}$ is defined as

$$\bar{f}_{i+1/2,j} = \mu_i f_{i+1/2,j} + (1 - \mu_i) f_{i-1/2,j}, \tag{3.2.3}$$

and where the switching operator μ is 1 for a local Mach number lower than 1, and 0 otherwise. Note that the fluxes at midpoints are reconstructed by an averaging procedure.

The algorithm was first implemented to solve flows around airfoils and was extended to more complex configurations by various authors. The computations were carried out using either the conservative or the non-conservative form of the equation. Examples of applications include computations around axisymmetric bodies [60], three-dimensional isolated wing [61, 62], wing-body [63, 64, 65], wing-body-store [66, 67] and wing-body-canard [68] configurations, and turbomachinery cascades [69, 70]. The assumptions behind the TSD equation however restrict its application to thin bodies at low angles of attack, which is not often the case in practice. Moreover, the shockwaves were not always accurately captured, as reported by Bailey and Ballhaus [71] for example. Researchers thus turned to the full potential equation.

3.2.2 Early work on the full potential equation

The first two-dimensional codes solving the full potential equation were developed by Steger and Lomax [72] and Garabedian and Korn [73]. The authors used finite differences to discretize the equation on grids built using conformal mapping of a circle into an airfoil. The solution was obtained through successive line over-relaxation [74], and the codes were extended to incorporate a boundary layer correction method [75]. Later in the seventies, the algorithms solving the full potential equation were sucessfully generalized to handle more complex geometries, such as axisymmetric blunt bodies [76, 77], inlets [78, 79, 80, 81], and turbomachinery airfoil cascades [82, 83], and to feature inverse design options [84].

In 1974, Jameson and Caughey [85, 86] developed `FLO22`, the first three-dimensional full potential solver. The solver uses finite differences and the grid is built using a shear-parabolic conformal mapping [87]. The iteration scheme is also based on successive line over-relaxation. Moreover, Jameson enhanced the Murman and Cole algorithm and developed the concept of rotated difference. The key idea was to match the computational domain of dependence, *i.e.* the region upon which the solution at a point depends, to the physical and mathematical one. The rotated difference scheme allows to use upwind differentiation only in the streamwise direction and is thus more stable and accurate.

All codes developed so far were based on the non-conservative form of the full potential equation 3.1.4. In such codes, the solution, and more particularly the shock strength and position, was found to be affected by numerical parameters. Jameson thus extended his research work to the conservative form of the full potential equation 1.2.9 and developed the first conservative solver in 1975 [88]. The solver was extended with the help of Caughey and their work resulted in several codes between 1977 and 1980: `FLO27` [89], `FLO28` [90] and `FLO30` [91]. These codes use a similar grid generation process and iteration scheme as `FLO22`, but the discretization is based on finite volumes instead. While `FLO27` is only able to discretize isolated wings

mounted on an infinite cylinder representing the fuselage, FLO28 and FLO30 feature more advanced techniques to treat the fuselage. Verhoff and O'Neil [27] performed a comparative study of the FLO codes. More specifically, the authors investigated the different fuselage modeling capabilities implemented in theses codes. Over the years, various authors succesfully enhanced these codes. FLO27 was extended by Chen et al. [92] to include a boundary layer correction method. Moreover, the authors were able to improve the grid generation process to discretize pylons and nacelles, and performed transonic computations with power-off and power-on engines. FLO28 was extended in a similar way, using a grid generation technique developed earlier by Thompson et al. [93], as described by Yu [94]. Finally, viscous correction methods were also implemented in FLO30 by Street [95] and Woodson et al. [96].

In Jameson and Caughey's work, the rotated difference scheme is implemented by splitting the fluxes into physical terms f and g, and artificial viscosity terms P and Q. The full potential flux vector in Equation 3.1.2 then reads,

$$\mathbf{F} = \begin{bmatrix} f + P \\ g + Q \end{bmatrix}. \tag{3.2.4}$$

The full potential equation is then discretized as

$$\overleftarrow{\delta}_{\xi} \left(f_{i+1/2,j} + P_{i+1/2,j} \right) + \overleftarrow{\delta}_{\eta} \left(g_{i,j+1/2} + Q_{i,j+1/2} \right) + A_{i,j} = 0, \tag{3.2.5}$$

where $\overleftarrow{\delta}$ denotes a backward derivative. The components of the flux vector are computed as

$$\begin{aligned} f_{i+1/2,j} &= \frac{1}{2} \left[\frac{\rho u}{J}|_{i+1/2,j+1/2} + \frac{\rho u}{J}|_{i+1/2,j-1/2} \right], \\ g_{i,j+1/2} &= \frac{1}{2} \left[\frac{\rho v}{J}|_{i+1/2,j+1/2} + \frac{\rho v}{J}|_{i-1/2,j=1/2} \right], \end{aligned} \tag{3.2.6}$$

where u and v are the velocity components and J is the determinant of the Jacobian matrix mapping the cell from the physical space (x, y) to the computational space (ξ, η). The P and Q terms acting as artificial viscosity are defined as

$$\begin{aligned} P_{i+1/2,j} &= \begin{cases} \frac{\mu\rho}{Ja^2} \left(u^2 \delta_{\xi\xi} + uv \delta_{\xi} \delta_{\eta} \right) \phi_{i,j}, & u_{i+1/2,j} > 0, \\ -\frac{\mu\rho}{Ja^2} \left(u^2 \delta_{\xi\xi} + uv \delta_{\xi} \delta_{\eta} \right) \phi_{i,j}, & u_{i+1/2,j} < 0, \end{cases} \\ Q_{i,j+1/2} &= \begin{cases} \frac{\mu\rho}{Ja^2} \left(uv \delta_{\xi} \delta_{\eta} + v^2 \delta_{\eta\eta} \right) \phi_{i,j}, & v_{i,j+1/2} > 0, \\ -\frac{\mu\rho}{Ja^2} \left(uv \delta_{\xi} \delta_{\eta} + v^2 \delta_{\eta\eta} \right) \phi_{i,j}, & v_{i,j+1/2} < 0, \end{cases} \end{aligned} \tag{3.2.7}$$

where a is the speed of sound. The switching function μ allows to activate the viscous terms in the supersonic regions and is defined as

$$\mu = \max \left(0, 1 - \frac{M_{\mathrm{C}}^2}{M^2} \right), \tag{3.2.8}$$

where M_{C} is a user-specified cut-off Mach number. In Equation 3.2.5, the term $A_{i,j}$ restores

the continuity in the solution between adjacent cells, hence effectively removing odd-even decoupling [89].

As demonstrated by Klopfer and Nixon [97], an interesting feature of solving the conservative form of the equation is the ability to add an entropy correction. The correction is based on the Rankine-Hugoniot relation [98], and allows full potential calculations to capture stronger shocks more accurately and to achieve the same accuracy as solving the Euler equations. Applications for both steady and unsteady flows are available in a work by Parrineello and Mantegazza [99].

3.2.3 Transonic computation techniques

Following the successful work of Jameson on the rotated difference scheme, several authors [100, 101, 102] established a density upwinding procedure during 1978 and 1979. The numerical viscosity term is not added to the equation, but an upwind bias is directly added to the density instead. This is mathematically equivalent to the rotated difference scheme, and physically consistent with the local hyperbolic nature of the equation.

In the artificial density method, the physical density ρ in the potential equation is replaced by a biased (*i.e.* upwinded) density $\tilde{\rho}$, which can be expressed as

$$\tilde{\rho} = \rho - \mu \overleftarrow{\delta_s} \rho \Delta s, \tag{3.2.9}$$

where $\overleftarrow{\delta_s}$ denotes a derivative in the opposite direction of the flow and Δs is the local cell size. The streamline upwind derivative of the density can be evaluated in different ways. For example, Hafez et al.[102] used

$$\overleftarrow{\delta_s} \rho \Delta s = \frac{1}{q} \left(u \overleftarrow{\Delta_x} \rho + v \overleftarrow{\Delta_y} \rho \right), \tag{3.2.10}$$

where q is the velocity magnitude, and where $\overleftarrow{\Delta_x} \rho$ and $\overleftarrow{\Delta_y} \rho$ are backward differences of the density in the x and y directions, respectively. In Equation 3.2.9, the amount of bias is controlled by the switching function μ, which is generally defined as

$$\mu = \mu_{\mathrm{C}} \max \left(0, 1 - \frac{M_{\mathrm{C}}^2}{M^2} \right), \tag{3.2.11}$$

where μ_{C} and M_{C} are user-specified constants, and controls the amplification of the bias and the extent of the region where the bias is to be applied.

A further improvement to the density upwinding procedure is the flux upwinding scheme which was implemented by various authors [103, 104, 105, 106, 107, 108, 109]. In this method, the entire mass flux is upwinded to produce smooth gradients through the sonic regions, which results in better shock capturing. Studies comparing the density and flux upwinding schemes made by Habashi and Hafez [110], Volpe [111] and Dulikravich [112] indicate that the flux upwinding procedure tends to yield better results for weak shocks. The convergence characteristics of the two methods are however very similar.

In the flux upwinding scheme, the physical density ρ is replaced by a biased density $\tilde{\rho}$ in the potential equation, which is expressed as

$$\tilde{\rho} = \rho - \frac{\Delta s}{q} \overleftarrow{\delta}_s \left(\overline{\rho \mathbf{q}} \right), \tag{3.2.12}$$

where Δs is the cell size. Similar to the density upwinding procedure, the upwinded flux is approximated by

$$\overleftarrow{\delta}_s \left(\overline{\rho \mathbf{q}} \right) = \frac{1}{q} \left(u \overleftarrow{\delta}_x \left(\overline{\rho \mathbf{q}} \right) + v \overleftarrow{\delta}_y \left(\overline{\rho \mathbf{q}} \right) \right), \tag{3.2.13}$$

where the mass flux is expressed as

$$\overline{\rho \mathbf{q}} = \begin{cases} 0, & if M < 1, \\ \rho \mathbf{q} - \rho^\star \mathbf{q}^\star & if M \geq 1, \end{cases} \tag{3.2.14}$$

where $\mathbf{q}$ is the velocity vector and where starred quantities denote sonic values. The above formulation is second order in subsonic regions and first order in supersonic regions. It can further be complemented by a limiter to switch between first and second order formulations at local extrema, thus reducing oscillations in the solution. The density and flux upwinding procedures are efficient and practical to implement, since only the density needs to be modified in the full potential equation. As a consequence, current codes solving the equation are usually based upon these techniques.

Another original method for capturing shockwaves was developed by Bristeau et al. in 1985 [113]. It expresses the solution of the full potential equation as a constrained minimization problem to be solved by a least squares conjugate gradient method under a finite element formulation. The constrain consists in a term penalizing expansion shocks, but not affecting compression shocks. Though Bristeau and his colleagues obtained good results, the method is harder to implement and thus not widespread.

3.2.4 Kutta condition implementation

Few details are usually given on the implementation of the Kutta condition, and some authors do not even mention it. In two-dimensional cases, such as flow over airfoils, the Kutta condition is usually enforced by computing the potential jump, *i.e.* the circulation, at the trailing edge and imposing it to be constant on a wake extending downstream. The wake is thus modeled as an infinitesimal gap in the mesh, where the potential can be discontinuous. For example, Bristeau et al. [113] used this approach. This strategy can be extended to three dimensions, by considering each spanwise section of a wing to be an airfoil. The circulation can then be computed at discrete locations of the wing trailing edge and imposed to be constant on spanwise slices of the wake. This implementation finds its roots in the panel method, in which only the boundary needs to be discretized. It is in essence restricted to structured grids, but can be extended to unstructured grids. Each wake node must first be projected on the trailing edge. The circulation must then be interpolated from adjacent trailing edge nodes to the projected

wake node location. The interpolated value must finally be imposed on the original wake node. The procedure is schematized in Figure 3.2.1. However, this implementation is not practical and alternative implementations will be explained in section 3.2.6.

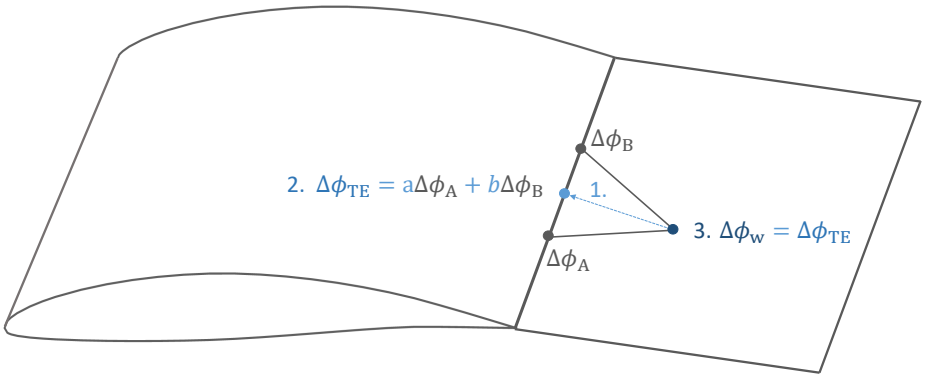


Figure 3.2.1: Illustration of basic Kutta condition implementation for three-dimensional unstructured grids. 1) project the wake node on the trailing edge. 2) interpolate the potential jump from adjacent trailing edge nodes to the projected wake node location. 3) impose the interpolated value of the circulation on the wake node.

3.2.5 Commercial software

With the advent of more efficient and robust full potential solvers, engineers became interested in studying more complex configurations, such as full aircraft with nacelles. For such cases, the finite volume and finite element methods are natural choices since they allow more freedom in the grid generation process. However, even today, some well-established codes still use finite difference methods combined with dedicated complex grid generators. Codes featuring advanced solution techniques, viscous correction methods and design features were then developed.

The most widely-used full potential solver is probably `Tranair`, developed by Johnson et al. [45, 46] at NASA and The Boeing Company in the last two decades of the 20th century. `Tranair` uses finite elements on non-conforming Cartesian grids to discretize the full potential equation. The variational formulation is based on the Bateman principle, which is modified for cells cut by a solid boundary. The simulation starts with a user-specified uniform grid, which is then refined using an Octree method. The refinement can be driven by the solution in order to capture sharp gradients accurately. `Tranair` uses either density upwinding or flux upwinding, the latter being first or second order. Since the surface grid is structured, the Kutta condition can be implemented in a panel-like fashion. The potential jump at the trailing edge of each spanwise station is computed and imposed on every wake node downstream of that spanwise station, as explained in the previous section. The resulting set of equations is solved using a quasi-Newton

method with the Bank and Rose [47] line search algorithm and the linear iteration scheme is GMRES [114]. `Tranair` has been coupled to the integral form of the boundary layer equations. Drela's two-dimensional code, `ISES` [16, 115], was initially used, but the formulation was later extended by Mughal [18] to handle quasi-2D boundary layers, *i.e.* two-dimensional boundary layers extended with a crossflow model, which are typically encountered in aeronautical flows. The inviscid and viscous equations are solved in a fully-simultaneous fashion, as described in Lock's work [20]. `Tranair` also features a design and optimization procedure. The geometric sensitivities with respect to the objective function, constraints and flow variables are computed using finite differences. The effect on the geometry is then modeled through surface transpiration. More specifically, the impermeability boundary condition is adjusted to allow a blowing or a suction on the boundary. The methodology is similar to that employed in viscous correction techniques. Details about the formulation can be found in several works by the `Tranair` development team [116, 117, 118, 119]. `Tranair` finally features a basic dynamic aeroelactic modeling capability. The unsteady flow is modeled by solving for linear harmonic perturbations about a pre-computed steady flow. Imposing the mode shapes of a flexible wing as a boundary condition allows to compute the flutter speed for a given flight condition. The equations are given in chapter 7 and further details about the formulation can be found in the manual [120].

Another commercially available code is `blwf` developed by Karas and Kovalev [121] at the Central Hydrodynamics Institute (TsAGI) in Russia. The code discretizes the conservative form of the full potential equation using finite volumes, which is then solved by the approximate factorization scheme [34, 122]. Transonic flows are stabilized with first or second order multiplicative artificial viscosity based on Engquist and Osher's work [123]. The code includes an algebraic grid generator able to map wing/body configurations to a C-grid topology. If tails or nacelles are present, a chimera approach, similar to that described by Holst [124], is used to superpose the different sets of grid. `blwf` also offers the possibility to solve the boundary layer equations, either in their integral formulation, like in `Tranair`, or discretized using finite differences. The viscous-inviscid coupling is performed in a quasi-simultaneous fashion [20]. As in the case of `Tranair`, `blwf` has been extended to solve unsteady flows modeled by linear harmonic small perturbations to compute the flutter speed. Furthermore, the code incorporates a simple beam modeling technique to discretize the structure of the wing, allowing to compute static aeroelastic wing deflections. Finally, `blwf` has been extended to solve the Euler equations.

The `VFP` software [125] was developed at the Aircraft Research Association[2] and RAE/DERA (now QinetiQ[3]). It uses a finite difference discretization on a structured grid with a form of relaxation as an iteration scheme. The boundary layer is modeled by solving the integral form of the boundary layer equations, which are coupled to the inviscid equations through a semi-inverse method [20]. The code is currently able to handle wing/body configurations, without tails or nacelles, and is still under development [126].

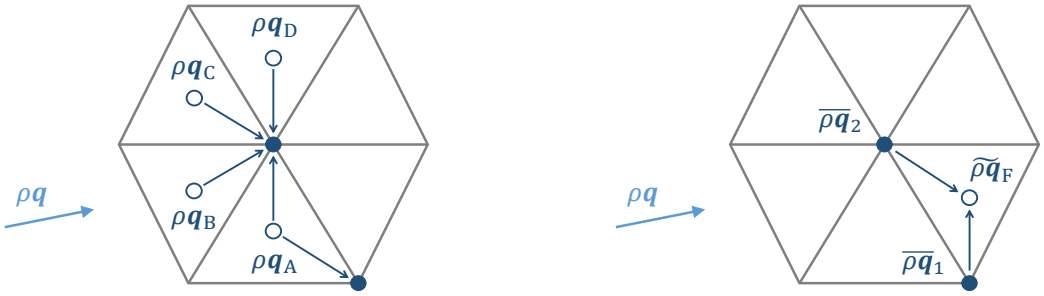
[2] https://www.ara.co.uk/
[3] https://www.qinetiq.com

3.2.6 Research codes

In parallel to the commercial solvers referenced in the previous section, researchers developed and implemented their own codes, improving existing characteristics and proposing new features. Although the first potential solvers were developed thirty to forty years ago and the technology is mature, new codes are still emerging today.

In 1995, Nishida [12] developed a full potential solver in the context of a fully simultaneous viscous-inviscid coupling simulation framework. His implementation is based on a finite element formulation with structured grids and uses a simple density upwinding scheme, where the upwind element is taken to be the previous one in the streamwise direction. Moreover, he implemented the Kutta condition in a similar way to periodic boundary conditions, making it grid independent, as opposed to `Tranair`. Nishida also coupled his inviscid code to the integral form of the boundary layer equations, which are solved in a fully simultaneous fashion. The author performed transonic computations on several wings, and showed that his full potential solver yielded results similar to experimental data when the viscous correction method was used. Discrepancies in shock location were however observed depending on which spanwise station was considered. Nishida's work was further extended to deal with unstructured grids in 2017 by Galbraith et al. [13]. Moreover, Galbraith and his colleagues adapted Parrinello and Mantegazza finite volume formulation [127, 99] to finite elements. In their work, they explored a new way of upwinding the density, which is considered as a second variable instead of a function, explicitly computed from the potential by the isentropic flow relationship 1.2.10. Since the density is approximated to be linear inside an element, the streamline upwind derivative can be computed locally and the upwind bias (Equation 3.2.9) can be added without extending the computational stencil, *i.e.* without taking additional upstream elements into account. This is consistent with the finite element method, since neighboring elements are not needed, and the data structure does not need to be further enriched. Another advantage of this method is that it can be easily extended to higher order. Galbraith et al. tested their code on isolated wings and wing-fuselage configurations, at both subsonic and transonic speeds. In each case, they reported consistent results.

During his Master and Ph.D. work, Kinney [128, 129] developed an unstructured finite element implementation allowing the modeling of complex geometries, such as full aircraft configurations. He used a second order flux upwinding scheme with limiters. The upwinding is performed in two sweeps and depicted in Figure 3.2.2. During the first sweep, the mass flux computed on elements A to D is stored into their nodes 1 and 2. The second sweep then consists in upwinding the mass flux of the element F by only considering the upwind nodes (1 and 2) of this element. The upwinded mass flux is constructed by averaging the mass flux stored in the upwind nodes. A detailed explanation can be found in the work by Kinney et al. [130]. Similarly to the work by Galbraith et al., this method of upwinding the flux only needs an element-to-nodes data structure, which is naturally present in the finite element code.



(a) First sweep.

(b) Second sweep.

Figure 3.2.2: Flux upwinding (adapted from Kinney et al. [130]). The bar and the tilde symbols denote averaged and upwinded quantities, respectively.

Later in 1997, Kinney et al. [131] coupled their inviscid solver with the integral form of the boundary layer equations. The boundary layer is assumed to be two-dimensional and computed along several strips along the span of a wing. The viscous-inviscid iterations are performed in a fully simultaneous fashion, and wall transpiration is used to model the effect of the viscous flow on the inviscid flow. Results computed using the different solvers by Kinney et al. on various aircraft configurations can be found in several papers [130, 131, 132, 133, 134].

Later, Neel (1997) [9] and Liegl (2005) [10], used a finite volume formulation to solve the full potential equation on unstructured grids. Both implementations use a density upwinding scheme. The Kutta condition is enforced by computing the circulation at spanwise stations of the wing, and then used to compute a potential jump through the wake extending downstream of the lifting surface. This is basically an extension of the algorithm used by `Tranair` for unstructured grids, which is described in Figure 3.2.1. In order to improve the convergence speed of his code, Liegl used a panel method to initialize the flow, and more specifically, the value of the circulation, before the actual finite volume computation. Both authors compared their codes on two and three-dimensional cases, and compared their results to those obtained from other implementations based on the Euler and RANS equations, as well as to experimental data. They reported good convergence characteristics and solution times about one order of magnitude faster than Euler calculations. Particularly, Neel performed transonic computations on the Onera M6, and obtained similar results as those presented in chapter 2, in Figure 2.3.3.

More recently, in 2012, Eller [135] implemented a variable-order finite element method on unstructured grids. To ensure robustness, the density is modified when the flow becomes supersonic. Shock waves are therefore not captured and the method is limited to subcritical flows. The Kutta condition is formulated in the same fashion as Nishida, but with the help of a least squares method. Eller performed subsonic computations on the DLR-F4 wing-fuselage configuration [136] and obtained good agreement with experimental results. Davari et al. [137]

extended Eller's work in 2018 by embedding the wake sheet into the mesh, rather than modeling it explicitly by an infinitesimal gap. The potential jump across the wake is represented using cut elements. These elements have duplicated degrees of freedom on their nodes and use enriched shape functions. This way of handling the wake has great advantages for both the grid generation process and aeroelastic simulations, since the wake does not have to be geometrically modeled. Their implementation is also based on a least squares finite element method and is currently restricted to airfoils. The results obtained by Davari et al. showed that the wake modeling method they proposed features improved robustness compared to classical techniques.

In 2017, Lyu et. al. [11] developed a finite volume full potential solver based on an adaptive Octree Cartesian grid. In order to obtain a good initial grid, they also implemented a grid generator which uses a cell-cutting and merging algorithm to process the geometry. The mesh is then further coarsened or refined based on the different solutions obtained through the simulation. As the mesh is non-conforming, the impermeability boundary condition is enforced by using ghost-cells. The method uses a density upwinding scheme and the Kutta condition is enforced in a novel way by computing gradients on wake faces rather than directly assigning a potential jump, which is reported to accelerate the convergence. However, each wake node must still be mapped to its corresponding spanwise station on the trailing edge. The authors validated their codes on various three-dimensional configurations, such as the Onera M6, the DLR-F4 and a blended wing body. In each case, they obtained good agreement with Euler results or experimental data. More particularly, in the case of the Onera M6, the results are similar to those obtained in chapter 2, Figure 2.3.3.

3.3 Field panel methods

While researchers were focusing on the development of finite volume or element methods, the main numerical tool used in industrial aircraft design was the panel method[4], solving the linear potential equation. It thus felt natural to extend boundary element methods to solve transonic flows. This gave rise to the field panel method, in which a panel method is coupled to a finite difference method that represents flow nonlinearities coming from compressibility. This section briefly reviews the historical developments of the field panel method. Details about the theory and the numerical method will be given in chapter 4. For more information, the reader is directed to Rottgermann's thesis [138].

3.3.1 Early coupling between boundary elements and finite differences

The research by Piers and Slooff [139] laid the basis for coupling panel methods to finite difference algorithms [140]. In 1979, they performed transonic flow computations around airfoils with a method based on an integral formulation of the TSD equation. Casting Equation 3.1.5

[4]The panel method is still routinely used in today's aircraft design process due to its very low computational cost.

into its Poisson form (Equation 3.1.3) allows to rewrite it as

$$\begin{aligned} &\nabla \cdot (\nabla \varphi) = \sigma, \\ &\sigma = M_\infty^2 \left(1 + (\gamma + 1)\frac{\varphi_x}{u_\infty}\right) \varphi_{xx}. \end{aligned} \tag{3.3.1}$$

In the two-dimensional field panel method proposed by Piers and Slooff, the left-hand side of Equation 3.3.1 is a linear operator and is solved using a panel method. The geometry is approximated as a thin line made of flat panels. The right-hand side, referred to as field source, contains the nonlinear terms and is discretized in a Cartesian mesh with finite differences. Since the body is made of thin and flat panels, they can be aligned with the grid cell centers. The panel method and the finite difference algorithm are then used iteratively: the potential obtained by solving the first equation in Equation 3.3.1 is used to compute the right-hand side term, σ, which then modifies the impermeability boundary condition on the geometry in the panel method. Piers and Slooff applied their methods to two-dimensional nonlifting geometries and tried different forms of upwinding in their algorithm. They were able to capture shockwaves and obtained a good match with finite differences codes.

In 1982, Johnson et al. [141], working at Boeing, went a step further and proposed a variant of the field panel method for solving the full potential equation in Poisson form 3.1.3. In their implementation, the thick geometry is embedded in a Cartesian mesh. A least squares technique is then used to reduce the solution of the equation to a sequence of Poisson problems, solved by fast Fourier transforms in combination with a traditional panel method. They also extended their work to use the Euler equations in place of the full potential equation. This methodology was further extended by Young et al. in 1986 [142] to improve its computational efficiency. Johnson et al. used their code on various airfoils, and obtained a good match with Jameson's `FLO22` code for both subsonic and supersonic flows. Around the same time, in 1985, Erickson and Strande [143] working at NASA, extended `Panair` [48] to solve transonic flows with similar ideas. They formulated the method for three-dimensional configurations, but only applied it to two-dimensional transonic airfoils. They were able to match results from Jameson's `FLO36` [144] and Flores `TAIR` [145] full potential codes, and Pulliam's [146] Euler code. The joint work of Boeing and NASA researchers would later result in `Tranair`. Although `Tranair` is a finite element code, its roots lie in the field panel method from which it borrows some features, such as the imposition of the farfield boundary condition.

3.3.2 Recent field panel methods

In 1986, Sinclair [147] developed a two-dimensional code that he extended to three dimensions in 1988 [148]. Sinclair recognized that integral equation procedures were efficient for boundary computations but required a lot of computational time and memory for field computations. He therefore completely segregated the boundary and field computations and used advanced iteration schemes to solve the system of equations resulting from the finite difference discretization.

In 1994, Rottgermann and Wagner developed ROFPM [149]. They extended their in-house vortex lattice code [150] to account for wing thickness and to model transonic flow occurring at the tip of helicopter rotor blades. They embedded surface panels into a Cartesian grid, and added an artificial viscosity, similar to that produced by Jameson's rotated difference scheme, explicitly to the field source term in order to stabilize supersonic regions of the flow. They first implemented a field panel method by only using integral equation procedures. Then, based on Sinclair's findings, they split the boundary and field computations. More specifically, they used the approximate factorization scheme to compute the field sources inside the Cartesian grid, and a trilinear interpolation method to modify the impermeability boundary condition. With these techniques, they were able to reduce the computational time to 19% and the memory requirement to 2% of their first implementation. They used their code to compute the flow around the CARADONNA rotor [151] and were able to match both Euler and experimental results. Moreover, the authors reported that their code only required 5% of the computational time needed to obtain an Euler solution.

Later in 2002, Gebhardt et al. implemented a field panel method, with the objective of creating a fast aerodynamic modeling tool for transonic aircraft design [152]. As opposed to Sinclair and Rottgermann, they only used integral equation computations, but they introduced some improvements to the method. First, the authors added a minigrid technique to improve the accuracy on coarse grids and to easily compute field derivatives near the body. They also introduced the concept of sub-paneling to remove the oscillatory behavior of the integral equation near the body. Further details about these two techniques will be presented in chapter 4. In order to accelerate convergence, Gebhardt and his colleagues split the field source term into an harmonic and non-harmonic part, which allowed to use an explicit-implicit iteration scheme, featuring better convergence characteristics. Finally, they implemented a solution-driven adaptive mesh technique. The authors used a rectangular NACA 0012 and the Onera M6 wings to validate their code. They were able to capture weak shocks, but they were smeared and displaced downstream. This was attributed to a lack of grid refinement in the shock region. Gebhardt and his fellow researchers are the last known people to have worked on the field panel method before the present work.

3.4 Discussion

Extensive research on nonlinear potential flow models was carried out between the seventies and nineties. The early breakthrough on type-dependent differentiation applied by Murman and Cole to the TSD equation led various authors to develop finite difference, finite volume and, later, finite element methods to solve transonic flows. More specifically, the simple, yet effective, switching between central and upwind differences implemented by Murman and Cole was extended to the rotated difference scheme by Jameson, and further generalized to the density and flux upwinding schemes by various authors. The full potential equation was used to solve external flow problems ranging from two-dimensional airfoils to complete aircraft configu-

rations, as well as internal flows, such as turbomachinery cascades. The full potential equation was also coupled to the boundary layer equations to account for viscous effects and correct inviscid flow predictions. In all cases, the authors reported reliable and consistent results, comparable to higher fidelity equations or even experimental results, for a smaller computational cost.

Aside these developments, researchers also tried to improve the widely used and cost effective panel method by developing the field panel method. However, less research has been carried out on this technique, and, while authors like Sinclair, Rottgermann and Wagner reported that the field panel method was able to deliver accurate results with little computational resources, other researchers like Gebhardt and his colleagues reported a lack of resolution near sharp gradients and a prohibitive computational cost.

With the increase in available computer power, researchers nowadays usually turn to the Euler or Navier-Stokes equations. As a result, the majority of the developments on nonlinear potential methods stopped in the nineties and today's well established commercial codes are `Tranair`, `blwf` and `VFP`, all developed in those years. However, the importance of nonlinear potential solvers is still recognized in industrial aircraft design, as they are able to quickly deliver meaningful results: a full potential solver is typically at least one order of magnitude faster than an Euler solver. On the other hand, aircraft geometries are now usually described by complex computer aided design models. Moreover, optimization loops and coupled physics simulations are now performed in the early design stages. This encouraged some researchers to turn back towards full potential methods and to extend them with various features, such as using unstructured grids, computing adjoint solutions for efficient gradient computations, and ensuring that the solver is compatible for fluid-structure interaction computations. More specifically, Galbraith, Davari and their co-workers explored these features and proposed new ways of implementing the Kutta condition, details on which often lack in the literature.

Overall, though nonlinear potential technology is mature, solvers are still being developed to easily handle complex aircraft configurations for optimization or coupled physics simulations in the context of preliminary aircraft design [13, 137]. However, with the notable exception of the work by Davari et. al., embedded in the open source framework Kratos Multiphysics [153, 154], no code is either in the public domain, or freely available. Since full potential methods are able to quickly deliver meaningful results, two solvers will be developed and presented in the next two chapters. In chapter 4, a field panel method will first be implemented, as it is a simple extension to the well known panel method. In chapter 5, a finite element method will then be implemented.

Chapter 4

Field panel solution of the full potential equation

In this chapter, a field panel method will be developed and presented. The theory and the implementation will first be described, and computational examples will then be given to illustrate the advantages of the method as well as its limitations.

4.1 Theory

This section presents the theory underlying the field panel method. The nonlinear potential equation is first reformulated into an integro-differential equation and manipulated so that it can be solved by combining a boundary element method and a field technique.

4.1.1 Formulation

The full potential equation written in Poisson's form 3.1.3 can be integrated over a domain V enclosed by a surface S. The integral equation can then be transformed using Green's third identity to yield

$$\phi = \phi_\infty \underbrace{-\frac{1}{4\pi}\int_S \left[\tau\frac{1}{r} - \mu\mathbf{n}\cdot\nabla(\frac{\mathbf{1}}{\mathbf{r}})\right] dS}_{\varphi_\mathrm{b}} \underbrace{-\frac{1}{4\pi}\int_V \left[\sigma\frac{1}{r}\right] dV}_{\varphi_\mathrm{f}}, \tag{4.1.1}$$

where $\mathbf{r}$ is the distance vector $[r_x, r_y, r_z]$, r is its norm defined by $\sqrt{r_x^2 + r_y^2 + r_z^2}$, and $\mathbf{n}$ is a unit vector normal to surface S pointing inwards V.

In Equation 4.1.1, the total potential ϕ can be considered as the superposition of the freestream potential ϕ_∞, a surface-induced potential φ_b, and a field-induced potential φ_f. The freestream potential is given by a uniform, undisturbed flow at a given angle of attack α, and can be computed as

$$\phi_\infty = x\cos\alpha + z\sin\alpha, \tag{4.1.2}$$

where the x and z coordinates are computed using a Cartesian frame of reference, and where the angle α lies between the horizontal plane and the freestream velocity vector. Note that Equation 4.1.2 has been normalized by the freestream velocity. The surface induced-potential can be modeled by source singularities τ and doublet singularities μ, and their strength can be computed by a panel method. The field-induced potential can be modeled by field sources

σ, whose strength is given by the second equation in Equation 3.1.3 and is usually computed using finite differences. The two equations 3.1.3 can then be solved iteratively by a panel method and a field module.

4.1.2 Panel method

The theory concerning the panel method described in the present chapter mainly comes from Katz and Plotkin [155]. The panel method is a boundary element method solving the integral form of the linear potential equation. Since the equation is linear, only boundary (surface) terms need to be retained in Equation 4.1.1, yielding

$$\phi = \phi_\infty \underbrace{-\frac{1}{4\pi}\int_S \left[\tau\frac{1}{r} - \mu\mathbf{n}\cdot\nabla(\frac{\mathbf{1}}{\mathbf{r}})\right] dS}_{\varphi_\mathrm{b}}. \tag{4.1.3}$$

Note that Equation 4.1.3 is a form of Equation 1.2.13 for a particular choice of the kernel function.

Two boundary conditions must be enforced to solve the integral linear potential equation 4.1.3. The first is the farfield boundary condition, which states that any perturbation potential decays far from the body. Since the integral is proportional to the inverse of the distance between the body and the boundaries, the farfield boundary condition is automatically fulfilled, provided that the singularities τ and μ are zero on the farfield boundary. As a result, no singularities are placed on the farfield boundary. The second boundary condition is the impermeability, which enforces zero normal velocity on the geometry's surface, and is formulated as a Neumann boundary condition,

$$\nabla\phi_\mathrm{b}\cdot\mathbf{n} = 0 \Leftrightarrow \nabla\varphi_\mathrm{b}\cdot\mathbf{n} = -\nabla\phi_\infty\cdot\mathbf{n}. \tag{4.1.4}$$

Equation 4.1.4 allows to determine the sources τ, since they induce a discontinuity in the potential which is equal to

$$\Delta\phi_\tau = \frac{\partial(\phi-\phi_\mathrm{i})}{\partial n}, \tag{4.1.5}$$

where ϕ_i is the potential inside the geometry. Consequently, setting $\phi_\mathrm{i} = 0$ directly allows to set the surface sources to include the normal component of the freestream velocity. Note that, since the flow velocity is normal to the iso-potential lines, this is equivalent to imposing a Dirichlet boundary condition as demonstrated by Katz and Plotkin [155].

The Kutta condition must be enforced in order to allow a potential (irrotational) flow to smoothly leave the geometry and to generate aerodynamic loads. In the panel method, this is accomplished by adding a wake sheet, extending horizontally from the trailing edge of the geometry. To accurately represent the physics, the wake should be shed and follow the flow, thus representing a force-free wake, instead of being flat. However, flat and force-free wakes produce very similar results for non-rotating steady flows, see Holst [34]. Doublet singularities are then placed on the wake sheet, and their magnitude is set to match the circulation around the geometry, which is computed as the difference between the strengths of the doublets lying on the

suction and the pressure sides at the trailing edge of the geometry. The wake doublets are thus linked to the surface doublets, and no additional unknowns are introduced.

The final step consists in discretizing the geometry and the wake into panels, which contain the source and doublet singularities, grouped in two vectors $\boldsymbol{\tau}$ and $\boldsymbol{\mu}$. The surface integral in Equation 4.1.3 can then be replaced by a discrete sum of elementary integrals on the panels. Combining the discrete form of Equations 4.1.3 and 4.1.4 yields the set of linear equations,

$$\mathbf{A}\boldsymbol{\mu} + \mathbf{B}\boldsymbol{\tau} = \mathbf{0}, \tag{4.1.6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the aerodynamic influence coefficient matrices, depending solely on the geometry. The expression of the influence coefficients will be defined below. Note that the influence coefficients related to wake panels are directly included in the matrix $\mathbf{A}$. Also note that the only unknown in Equation 4.1.6 is $\boldsymbol{\mu}$, since $\boldsymbol{\tau}$ is fully determined by the impermeability boundary condition 4.1.4.

4.1.3 Field module

The field module supplements the panel method and models the compressibility of the flow. The theory concerning the field module of the field panel method mainly comes from Gebhardt et al. [152] and Chu et al. [156].

The role of the field module is to compute the volume integral in Equation 4.1.1. The volume is discretized using a rectangular Cartesian grid enclosing the geometry. Since the farfield boundary condition is automatically satisfied by the panel method, the domain size only needs to be large enough to contain the nonlinearities in the flow. Each mesh cell is considered as a field panel, which contains a field source σ. Note that the grid is non-conforming, and zero-strength field sources are associated with field panels located inside the geometry. Explicit finite differences are used to discretize the second equation in Equation 3.1.3 and to compute the vector of field sources, $\boldsymbol{\sigma}$. The volume integral in Equation 4.1.1 can then be replaced by a discrete sum of elementary integrals on the field cells, and the vector of total potential in the field $\boldsymbol{\phi_f}$ is computed as

$$\boldsymbol{\phi_f} = \boldsymbol{\phi_\infty} + \mathbf{A}_\mathrm{f}\boldsymbol{\mu} + \mathbf{B}_\mathrm{f}\boldsymbol{\tau} + \mathbf{C}\boldsymbol{\sigma}, \tag{4.1.7}$$

where $\mathbf{A}_\mathrm{f}$, $\mathbf{B}_\mathrm{f}$, and $\mathbf{C}$ are influence coefficient matrices, and where the vector of surface singularities $\boldsymbol{\mu}$ and $\boldsymbol{\tau}$ are updated by the panel method. To close the iterative procedure and to enforce the impermeability boundary condition, the surface source singularities must now include the normal component of the freestream velocity as well as the normal component of the velocity induced by the field sources. The new boundary condition reads,

$$\nabla\varphi_\mathrm{b} \cdot \mathbf{n} = -\nabla(\phi_\infty + \varphi_\mathrm{f}) \cdot \mathbf{n}, \tag{4.1.8}$$

where $\varphi_\mathrm{f} = \phi_\mathrm{f} - \phi_\infty$.

Figure 4.1.1 depicts a typical configuration for a field panel computation. Even though the the-

ory has been developed for three-dimensional computations, the configuration is drawn in two dimension, for clarity. Four surface panels and seven field cells, as well as their associated singularities, are drawn in blue. In practice, the farfield boundary, drawn in black, is located several chord lengths away from the body so that the domain encloses all the nonlinear phenomena, such as shock waves. The wake is made of one long panel that extends beyond the downstream boundary, which contains only one doublet singularity. Its value is set using the doublet strengths at the trailing edge.

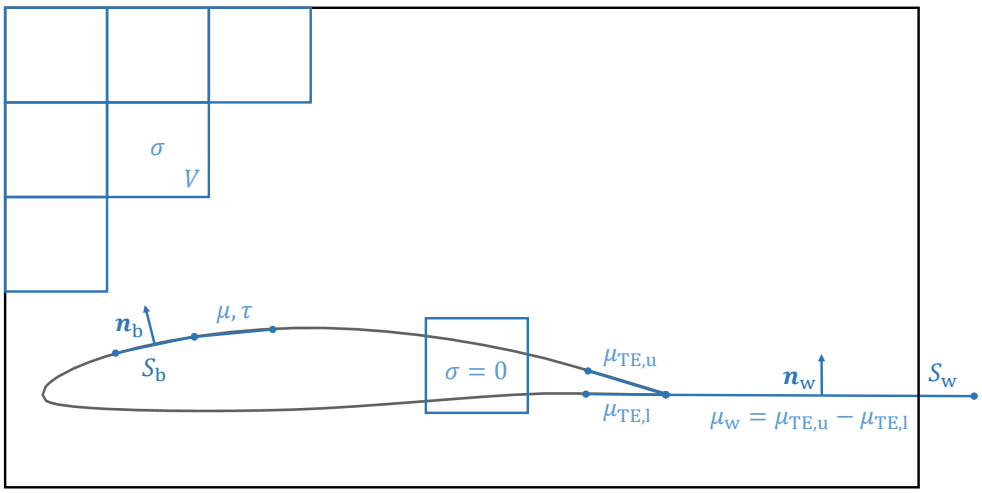


Figure 4.1.1: Field panel discretization.

4.1.4 Influence coefficients

The different matrices given in the previous section contain the aerodynamic influence coefficients needed by the field panel method. Their mathematical expression is given below.

Surface coefficients

The influence coefficients for the velocity potential induced by rectangular surface panels were derived by Hess and Smith [157] and are used to compute matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{A}_\mathrm{f}$, $\mathbf{B}_\mathrm{f}$.

The computations are carried out in the (flat) panel reference frame shown in Figure 4.1.2, whose collocation point is noted $O(x_0, y_0)$, corner points (numbered cyclically) are noted by (x_k, y_k) with $k = 1, 2, 3, 4$ and surface is noted S. The target point onto which the potential is sought is denoted by $P(x, y, z)$. In order to perform the change of frame of reference, the rotation matrix $\mathbf{R}$ is used to pre-multiply the vector $\mathbf{OP}$.

$$\mathbf{R} = \begin{bmatrix} \mathbf{e_x} \cdot \mathbf{e_X} & \mathbf{e_x} \cdot \mathbf{e_Y} & \mathbf{e_x} \cdot \mathbf{e_Z} \\ \mathbf{e_y} \cdot \mathbf{e_X} & \mathbf{e_y} \cdot \mathbf{e_Y} & \mathbf{e_y} \cdot \mathbf{e_Z} \\ \mathbf{e_z} \cdot \mathbf{e_X} & \mathbf{e_z} \cdot \mathbf{e_Y} & \mathbf{e_z} \cdot \mathbf{e_Z}, \end{bmatrix} \tag{4.1.9}$$

where $\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z}$ is the reference frame attached to the panel and $\mathbf{e_X}, \mathbf{e_Y}, \mathbf{e_Z}$ is the global reference frame.

To simplify the expression of the influence coefficients, the following variables are defined,

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{4.1.10}$$

$$m_{ij} = \frac{y_j - y_i}{x_j - x_i} \tag{4.1.11}$$

$$r_k = \sqrt{(x - x_k)^2 + (y - y_k)^2 + z^2} \tag{4.1.12}$$

$$e_k = (x - x_k)^2 + z^2 \tag{4.1.13}$$

$$h_k = (x - x_k)(y - y_k). \tag{4.1.14}$$

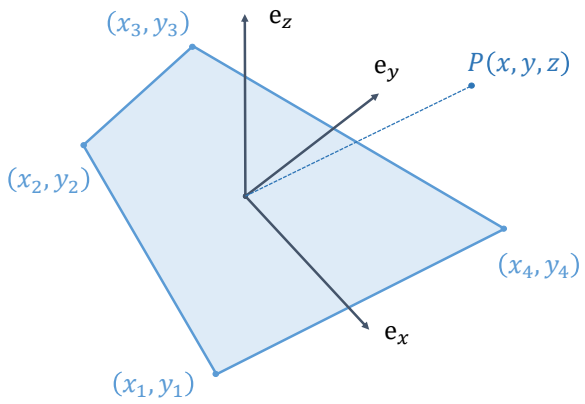


Figure 4.1.2: Panel and notations (adapted from Katz and Plotkin [155]).

The influence coefficients corresponding to source terms used to compute matrices $\mathbf{B}$ and $\mathbf{B}_\mathrm{f}$

are given by

$$\begin{aligned}\mathcal{AIC}_\tau = \frac{-1}{4\pi}\Bigg\{\Bigg[&\frac{(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)}{d_{12}}\log\frac{r_1+r_2+d_{12}}{r_1+r_2-d_{12}}\\ &+\frac{(x-x_2)(y_3-y_2)-(y-y_2)(x_3-x_2)}{d_{23}}\log\frac{r_2+r_3+d_{23}}{r_2+r_3-d_{23}}\\ &+\frac{(x-x_3)(y_4-y_3)-(y-y_3)(x_4-x_3)}{d_{34}}\log\frac{r_3+r_4+d_{34}}{r_3+r_4-d_{34}}\\ &+\frac{(x-x_4)(y_1-y_4)-(y-y_4)(x_1-x_4)}{d_{41}}\log\frac{r_4+r_1+d_{41}}{r_4+r_1-d_{41}}\Bigg]\\ -|z|\Bigg[&\arctan\left(\frac{m_{12}e_1-h_1}{zr_1}\right)-\arctan\left(\frac{m_{12}e_2-h_2}{zr_2}\right)\\ &+\arctan\left(\frac{m_{23}e_2-h_2}{zr_2}\right)-\arctan\left(\frac{m_{23}e_3-h_3}{zr_3}\right)\\ &+\arctan\left(\frac{m_{34}e_3-h_3}{zr_3}\right)-\arctan\left(\frac{m_{34}e_4-h_4}{zr_4}\right)\\ &+\arctan\left(\frac{m_{41}e_4-h_4}{zr_4}\right)-\arctan\left(\frac{m_{41}e_1-h_1}{zr_1}\right)\Bigg]\Bigg\}.\end{aligned} \tag{4.1.15}$$

The influence coefficients corresponding to doublet terms used to compute matrices $\mathbf{A}$ and $\mathbf{A}_\mathrm{f}$ are given by

$$\begin{aligned}\mathcal{AIC}_\mu = \frac{1}{4\pi}\Bigg[&\arctan\left(\frac{m_{12}e_1-h_1}{zr_1}\right)-\arctan\left(\frac{m_{12}e_2-h_2}{zr_2}\right)\\ &+\arctan\left(\frac{m_{23}e_2-h_2}{zr_2}\right)-\arctan\left(\frac{m_{23}e_3-h_3}{zr_3}\right)\\ &+\arctan\left(\frac{m_{34}e_3-h_3}{zr_3}\right)-\arctan\left(\frac{m_{34}e_4-h_4}{zr_4}\right)\\ &+\arctan\left(\frac{m_{41}e_4-h_4}{zr_4}\right)-\arctan\left(\frac{m_{41}e_1-h_1}{zr_1}\right)\Bigg].\end{aligned} \tag{4.1.16}$$

Field coefficients

The influence coefficients for the potential and the velocity induced by field panels were derived respectively by Seidov and Skvirsky [158], and Chu et al. [156]. They are used to compute the matrix $\mathbf{C}$, as well as the second term on the right-hand side in Eq 4.1.8.

A rectangular parallelepipedic field cell is considered. The origin of the frame of reference attached to the cell is located at the center of gravity of the cell and the axes are aligned with respect to the cell edges. The field cell is bounded by the planes $\xi=\xi_1$, $\xi=\xi_2$, $\eta=\eta_1$, $\eta=\eta_2$, $\zeta=\zeta_1$ and $\zeta=\zeta_2$, where ξ, η and ζ denote the coordinates in the frame of reference. If the cell holds a constant source singularity of strength σ, the potential induced by this field cell at an arbitrary point $P(x,y,z)$ located inside or outside of the cell is given by

$$\phi_\sigma(x,y,z) = \frac{-\sigma}{4\pi}\int_{\xi_1}^{\xi_2}\int_{\eta_1}^{\eta_2}\int_{\zeta_1}^{\zeta_2}\frac{1}{\sqrt{(x-\xi)^2+(y-\eta)^2+(z-\zeta)^2}}d\zeta d\eta d\xi. \tag{4.1.17}$$

To simplify the result of this integral, the following variables are defined,

$$A = x - \xi_i \tag{4.1.18}$$

$$B = y - \eta_j \tag{4.1.19}$$

$$C = z - \zeta_k \tag{4.1.20}$$

$$R = \sqrt{A^2 + B^2 + C^2}. \tag{4.1.21}$$

The potential influence coefficients induced by a field cell used to compute matrix $\mathbf{C}$ is given by

$$\begin{aligned} \mathcal{AIC}_\sigma = \frac{-1}{4\pi} \sum_{i,j,k=1}^{2} (-1)^{i+j+k} \Bigg[& BC \log(A+R) - \frac{A^2}{2} \arctan \frac{BC}{AR} \\ & + CA \log(B+R) - \frac{B^2}{2} \arctan \frac{CA}{BR} \\ & + AB \log(C+R) - \frac{C^2}{2} \arctan \frac{AB}{CR} \Bigg] \end{aligned} \tag{4.1.22}$$

The velocity influence coefficients induced by a field cell and used to compute the second term on the right-hand side in Equation 4.1.8 are given by

$$\mathcal{AIC}_{x,\sigma} = \frac{1}{\sigma}\frac{\partial \phi_\sigma}{\partial x} = \frac{1}{4\pi} \sum_{i,j,k=1}^{2} (-1)^{i+j+k} \left[\frac{B}{2} \log \frac{(R+C)}{R-C} + \frac{C}{2} \log \frac{(R+B)}{R-B} - A \arctan \frac{BC}{AR} \right] \tag{4.1.23}$$

$$\mathcal{AIC}_{y,\sigma} = \frac{1}{\sigma}\frac{\partial \phi_\sigma}{\partial y} = \frac{1}{4\pi} \sum_{i,j,k=1}^{2} (-1)^{i+j+k} \left[\frac{C}{2} \log \frac{(R+A)}{R-A} + \frac{A}{2} \log \frac{(R+C)}{R-C} - B \arctan \frac{CA}{BR} \right] \tag{4.1.24}$$

$$\mathcal{AIC}_{z,\sigma} = \frac{1}{\sigma}\frac{\partial \phi_\sigma}{\partial z} = \frac{1}{4\pi} \sum_{i,j,k=1}^{2} (-1)^{i+j+k} \left[\frac{A}{2} \log \frac{(R+B)}{R-B} + \frac{B}{2} \log \frac{(R+A)}{R-A} - C \arctan \frac{AB}{CR} \right] \tag{4.1.25}$$

4.2 Implementation

The theoretical procedure described in the previous section has been implemented into a new code developed for this work, called `Aero` [159, 160]. The code has been written in the high-level, scientific and efficient C++ language. Extensive usage of structures has been made to keep the code organized and simple to use. Overall, the code is fairly modular and can be modified quite easily if further development is needed. More details about the code can be found in appendix B.

4.2.1 Geometry treatment

Before actual computations can be carried out, the geometry and the field must be discretized and stored in the data structure. Additional treatment of the field cells and the surface panels

is also required to ensure solution consistency and convergence. The surface panels are flat quadrilateral surfaces defined by their four corner points, their center of gravity, and three unit orthogonal vectors: longitudinal, transverse and normal. The field cells are defined by their centroids and the cell size in the x, y and z directions. Note that the mesh is uniform, *i.e.* the field panels all have the same size. The surface and the field panels data are stored into matrices, which are regrouped into two structures. If the code needs to be extended to handle several networks of surface or field panels, these structures can be easily vectorized.

Mapping of field panels

Since the volume grid is Cartesian and does not conform to the body, cells lying inside the body, *i.e.* internal cells, must be distinguished from cells outside the body , *i.e.* external cells. Moreover, two adjacent external cells may be separated by a wake surface or several body surfaces. This will be problematic when using finite differences to compute the derivatives of the potential in the field to obtain the source term, since the potential is discontinuous across these surfaces. Several techniques can be used to address this issue while allowing complex geometries to be handled. These techniques include: minigrid, jump relations, and cell sorting. The first technique has been proposed by Gebhardt et al. [152] and has been tested in the present code. It is detailed in appendix B. The second technique consists in identifying the cells adjacent to any surface and implementing jump relations inside the finite differences. In this way, the discontinuity in the potential is taken into account. This method was first proposed by Rottgermann and Wagner [149]. Finally, the third technique, which has been used in the present code, consists in identifying the problematic cells and disabling the derivative in the problematic direction.

In order to identify the situation of a cell, a three-dimensional adaptation of the point-in-polygon algorithm [161] and ray casting algorithm [162] is used:

1. Cast a ray from the center of the cell in the x-direction (to infinity)
2. Count the number of surface panels the ray crosses
    - check if the ray is not parallel to the plane containing the panel
    - check if the ray intersects this plane and compute the intersection
    - check that this intersection is included in the panel
3. If the number of valid intersections is even, the point lies outside the body

To check if the derivative can be computed in a given direction for any external cell, the above algorithm is also used, except that the ray is cast from the center of the cell to the center of adjacent cells. If one valid intersection between the ray and a surface panel is detected, the derivative in the given direction is immediately disabled and set to zero.

Sub-paneling

The aerodynamic influence coefficients related to constant singularities of a surface panel are singular and discontinuous on the panel edges. If the potential needs to be computed close to the surface panel edges, the solution might be oscillatory. In the panel method, this never happens in practice, since the solution is only needed at the surface panel center. However, for a field panel method, the potential is also required at the field cell center, which might be located close to the surface. In `Aero`, a subpaneling technique has been used to remove the oscillations appearing under such conditions. The method was also used by Gebhardt et al. and was originally proposed as a *subvortex technique* by Maskew [163]. It consists in using a linear singularity strength distribution instead of constant singularities for surface panels which are close to field cells. In this way, constant singularities, which are computationally cheap, are used to compute the majority of the influence coefficients, while linear singularities, more expensive, are used only to handle problematic panels, where extra stability is required.

In practice, the sub-paneling is performed in two steps. First, if a field cell center is located too close to the surface of the geometry, the surface panel is split into a user-defined number of sub-panels, with constant singularity strength. Then, at each iteration, when the singularity strength is known on each surface panel, the singularity is interpolated linearly on the sub-panels. The bilinear interpolation, also performed in two steps, is illustrated in Figure 4.2.1. The contribution to the potential in the field at cell i by the surface panel j (split in n_s sub-panels) can then be computed as

$$\varphi_{\mathrm{f}}|_{i,j} = \sum_{k}^{n_s} A_{\mathrm{f},k}\mu_k + B_{\mathrm{f},k}\tau_k, \tag{4.2.1}$$

where k is the subpanel index.

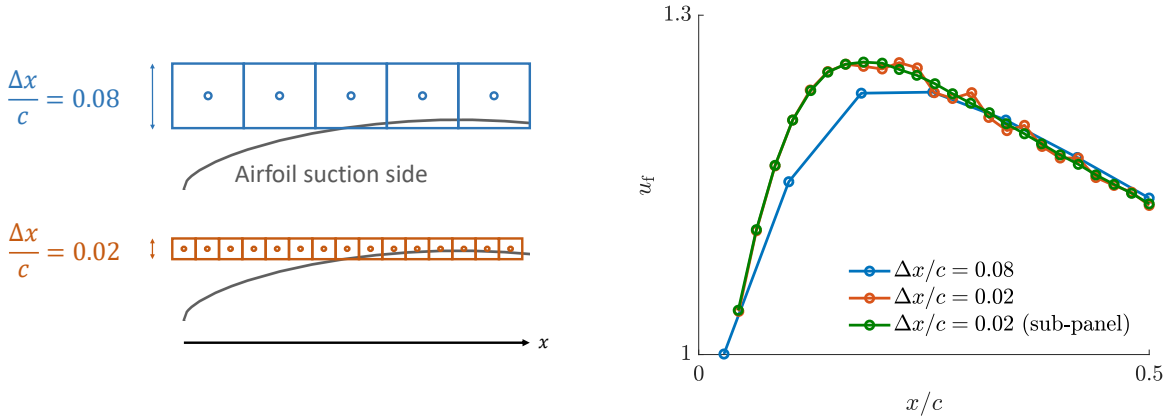
(a) From panel centers to panel vertices.

(b) From panel vertices to sub-panel centers.

Figure 4.2.1: Singularity bilinear interpolation on sub-panels.

The effect of the sub-paneling technique on the solution smoothness is illustrated in Figure 4.2.2. Figure 4.2.2a shows the first layer of field cells located directly above the first half of the chord of a wing section. The x-component of the velocity computed at the center of these

field cells, u_f, is plotted along the horizontal direction in Figure 4.2.2b. For large field panel sizes ($\Delta x/c = 0.08$, in blue), the solution is not converged and the mesh must be refined. However, when the cell size is too small ($\Delta x/c = 0.02$, in red), oscillations caused by the discontinuity in the surface singularities across surface panels start to appear. Since the velocity is used to compute the source term, which will in turn correct the panel method and drive the solution process, these oscillations can lead to the divergence of the algorithm. The oscillations can be effectively removed by the sub-paneling technique and a smooth solution can be obtained, as shown by the green curve.



(a) First layer of field cell above the airfoil suction side for two grid sizes.

(b) x-component of the field velocity for two grid sizes, with and without sub-paneling.

Figure 4.2.2: Effect of Grid size and sub-paneling on velocity for near-field cells.

4.2.2 Numerical treatment

Once the geometry and the mesh are defined and treated, the actual computations can be carried out.

Aerodynamic influence coefficient matrices

The current implementation of the field panel method relies on four groups of matrices, each stored into different structures: body-to-body, body-to-field, field-to-field and field-to-body aerodynamic influence coefficients. The matrices are referred to as $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A}_\mathrm{f}$ and $\mathbf{B}_\mathrm{f}$, $\mathbf{C}$, and $\mathbf{C}_\mathrm{b}$ respectively. Their source and target panels are illustrated in Figure 4.2.3. Each element of these matrices represents the influence of a panel onto another and depends solely on geometric parameters. The assembly of the matrices is therefore performed before the iteration loop, and can be reused outside the field panel method for other purposes, such as computation of sensitivities in the context of optimization. Note that the size of the matrices grows as N^2, where N is the number of panels. The number of panels should therefore be kept to a minimum. When a body-to-field influence coefficient of a panel that will be split needs to be computed, this coefficient is set to 0 instead, and the influence coefficients of the corresponding sub-panels are computed and stored into another matrix. The matrices related to the sub-panels are regrouped into a fifth structure.

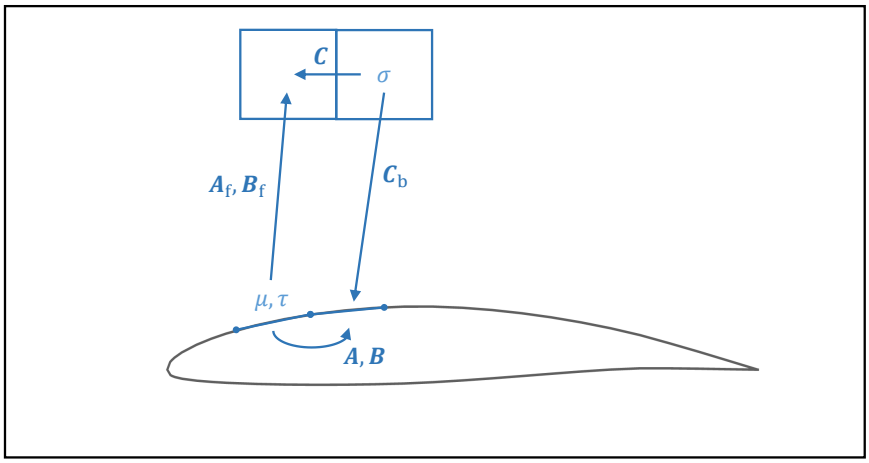


Figure 4.2.3: Source and target panels to compute the aerodynamic influence coefficient matrices.

Surface singularity computation

At each iteration, the surface source singularities τ are computed to fulfill the augmented impermeability boundary condition 4.1.8. The set of linear equations 4.1.6 can then be solved for the surface doublet singularities μ. In the present implementation, the set of equations is solved with the linear algebra library `Eigen` [164]. A direct solver, either based on a LU or a QR decomposition is used. Since the number of surface panels is not large, the use of a direct solver is appropriate.

Field variables computation

At each iteration, the velocity in the field must first be computed in order to obtain the field source singularities. Several options exist to compute the velocity in the field. The potential can either be computed first with aerodynamic influence coefficients using Equations 4.1.15, 4.1.16 and 4.1.22, then numerically differentiated with finite differences in the Cartesian grid. The second option consists in differentiating analytically the influence coefficients, so that they can be used directly to compute the velocity using Equations 4.1.23 to 4.1.25.

Both approaches have been implemented, tested and compared in the present work. The second approach is more accurate since it is purely analytic, but it involves the computation of three influence matrices instead of one (three components of the velocity instead of a scalar potential). Moreover, the velocity influence coefficients are more sensitive than the potential coefficients. As a result, near the body surface, the discontinuity between the singularity values on each panel induces oscillations in the velocity distribution in the field, despite the subpaneling technique. On the other hand, if a sufficiently fine grid is used, the finite differences of the potential give a very good approximation of the velocity computed directly with the analytic coefficients. Numerical experiments showed that the potential formulation, *i.e.* potential

computed with the one influence matrix and numerically differentiated with finite differences, required 40% less memory and was 35% faster than the second approach to achieve a similar solution. It has therefore been retained in the present work.

The potential in the field is computed with Equation 4.1.7. For field panels lying close to the surface of the geometry, the surface contributing terms are replaced by the right-hand side of Equation 4.2.1. The potential is then differentied with central finite differences, except near the boundaries, where one-sided differences are used. The density is computed from the velocity using the isentropic gas formula 1.2.10. The field sources can then be computed using the second equation in Equation 3.1.3, which is also discretized using finite differences. To close the iterative procedure, the impermeability boundary condition must be updated following Equation 4.1.8. In `Aero`, field-to-body aerodynamic influence coefficient matrices are used to obtain the velocity induced by the field cells on the surface panels, and to compute the second term of the right-hand side in Equation 4.1.8.

Supersonic flow treatment

As explained in chapter 3, the physical and mathematical nature of the flow changes in supersonic regions. Consequently, these changes must be reflected in the numerical method and supersonic flow regions must be stabilized. In the present work, three techniques have been tested: derivative upwinding, artificial density and artificial viscosity.

The simplest way to allow transonic flow computation is to use central differences in subsonic regions and backward (upwind) differences in supersonic regions, as in the Murman and Cole algorithm [59]. Since the grid is Cartesian, the upwinding is accomplished only along x, which is the main flow direction. This effectively stabilizes the iterative procedure, but the accuracy can be further improved. In order to properly upwind the derivatives, backward derivatives should be used in the local streamline direction of the flow, as prescribed by Jameson's rotated difference scheme [85, 88].

Rotated differences can be introduced in the solution by biasing the density in the upwind direction. In this procedure, the physical density is replaced by a biased density, which is computed using Equations 3.2.9, 3.2.10 and 3.2.11. In a very similar way, an artificial viscosity can be directly added to the field source term, as shown by Rottegermann and Wagner [149]. Using the artifical viscosity procedure, the field sources are computed as,

$$\tilde{\sigma} = \sigma + \max\left(0, 1 - \frac{M_{\mathrm{C}}^2}{M^2}\right) \frac{\partial \sigma}{\partial s} \Delta s, \tag{4.2.2}$$

where the cut-off Mach number has been chosen from the literature as $M_{\mathrm{C}} = 0.95$, and the streamline upwind derivative of the field sources is computed similarly to Equation 3.2.10,

$$\frac{\partial \sigma}{\partial s} \Delta s = \frac{1}{q} \left(u \overleftarrow{\Delta_x} \sigma + v \overleftarrow{\Delta_y} \sigma \right). \tag{4.2.3}$$

In the present work, both approaches have been tested and yielded identical results. The artificial viscosity approach has been retained since it is more practical to implement.

4.2.3 Solution procedure

This section describes the solution procedure currently implemented. It should be recalled that all the variables are vectors, since they are stored at each point of the grid. For clarity, the bold vector notation is only used to denote true vector variables, such as the velocity, whereas the aerodynamic influence coefficient matrices are underlined. The remaining variables are denoted in italic. Note that the field variables are normalized by setting the magnitude of the freestream velocity to 1.

Initialization

The geometry is first processed. Then, the field panels are created and mapped, and the subpaneling is performed. Afterwards, the matrices containing the influence coefficients are computed: body-to-body ($\underline{A}$ and $\underline{B}$), field-to-field ($\underline{C}$), body-to-field ($\underline{A}_{\mathrm{f}}$ and $\underline{B}_{\mathrm{f}}$) and field-to-body ($\underline{C}_{x,\mathrm{b}}$, $\underline{C}_{y,\mathrm{b}}$, $\underline{C}_{z,\mathrm{b}}$). All the relevant field variables, such as the field sources $\tilde{\sigma}$, and the normal component of the field-induced velocity on the body $u_{n,\sigma}$, are also initialized to zero.

1 - surface sources

The first step of the iterative process consists in setting the surface source singularities so that they include the non-lifting normal velocity component, according to Equation 4.1.8, as

$$\tau = -(\mathbf{U}_\infty \cdot \mathbf{n} + u_{n,\sigma}). \tag{4.2.4}$$

2 - surface doublets

The second step consists in solving the set of linear equations 4.1.6,

$$\underline{A}\mu + \underline{B}\tau = 0, \tag{4.2.5}$$

to obtain the surface doublet singularities, μ.

3 - field variables

The third step consists in computing the different field variables. The total potential in the field is first computed using the surface and field singularities as

$$\phi_{\mathrm{f}} = \phi_\infty + \underline{A}_{\mathrm{f}}\mu + \underline{B}_{\mathrm{f}}\tau + \underline{C}\tilde{\sigma}, \tag{4.2.6}$$

and is then differentiated to obtain the total velocity in the field $\mathbf{U}_\mathrm{f}$. The speed of sound a, the Mach number M and the density ratio $\frac{\rho}{\rho_\infty}$ are then computed using the isentropic relations

$$\begin{aligned} a^2 &= a_\infty^2 + \frac{\gamma-1}{2} - \frac{\gamma-1}{2}|\mathbf{U}_\mathrm{f}|^2 \\ M &= \frac{|\mathbf{U}_\mathrm{f}|}{a} \\ \frac{\rho}{\rho_\infty} &= \left[1 + \frac{\gamma-1}{2} M_\infty^2 \left(1 - |\mathbf{U}_\mathrm{f}|^2\right)\right]^{\frac{1}{\gamma-1}}. \end{aligned} \tag{4.2.7}$$

4 - field sources

The fourth step consists in updating the field source singularities using

$$\tilde{\sigma} = \nabla\left(\frac{\rho}{\rho_\infty}\right) \cdot \mathbf{U}_\mathrm{f} + \max\left(0, 1 - \frac{M_\mathrm{C}^2}{M^2}\right) \frac{\partial \sigma}{\partial s} \Delta s, \tag{4.2.8}$$

where the streamline upwind derivative of the field sources is computed using Equation 4.2.3.

5 - boundary condition update

The fifth step consists in updating the normal component of the field source induced velocity as,

$$u_{n,\sigma} = [\underline{C}_{x,\mathrm{b}}\tilde{\sigma}, \underline{C}_{y,\mathrm{b}}\tilde{\sigma}, \underline{C}_{z,\mathrm{b}}\tilde{\sigma}] \cdot \mathbf{n}. \tag{4.2.9}$$

Stopping criterion

Steps 1 to 5 are repeated until convergence. The iterative procedure is stopped when the maximum variation of the field sources drops below a user-defined tolerance. The stopping criterion is defined as

$$\max|\tilde{\sigma}^\mathrm{n} - \tilde{\sigma}^{\mathrm{n}-1}| < \varepsilon, \tag{4.2.10}$$

where n is the iteration counter and ε is the user-defined tolerance.

Finalization

The final step is to compute the surface velocity and the pressure coefficient. The surface velocity is the sum of the freestream velocity $\mathbf{U}_\infty$, the surface perturbation velocity $\mathbf{u}_\mathrm{b}$ and field perturbation velocity $\mathbf{u}_\mathrm{f}$ vectors. The latter has already been calculated to update the boundary condition. The surface perturbation velocity can be computed by differentiating the potential on the surface. If l is the chordwise tangent, m the spanwise tangent and n the normal unit vector of a surface panel, then

$$\begin{aligned} u_{l,\mathrm{b}} &= -\frac{\partial \mu}{\partial l} \\ u_{m,\mathrm{b}} &= -\frac{\partial \mu}{\partial m} \\ u_{n,\mathrm{b}} &= \tilde{\sigma}. \end{aligned} \tag{4.2.11}$$

The obtained velocity vector is then rotated to the global axes and used to compute the total

surface velocity,

$$\mathbf{U}_\mathrm{b} = \mathbf{U}_\infty + \mathbf{u}_\mathrm{b} + \mathbf{u}_\mathrm{f}. \tag{4.2.12}$$

The pressure coefficient can subsequently be computed as,

$$C_p = \frac{2}{\gamma M_\infty^2} \left\{ \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - |\mathbf{U}_\mathrm{b}|^2) \right]^{\frac{\gamma}{\gamma - 1}} - 1 \right\}. \tag{4.2.13}$$

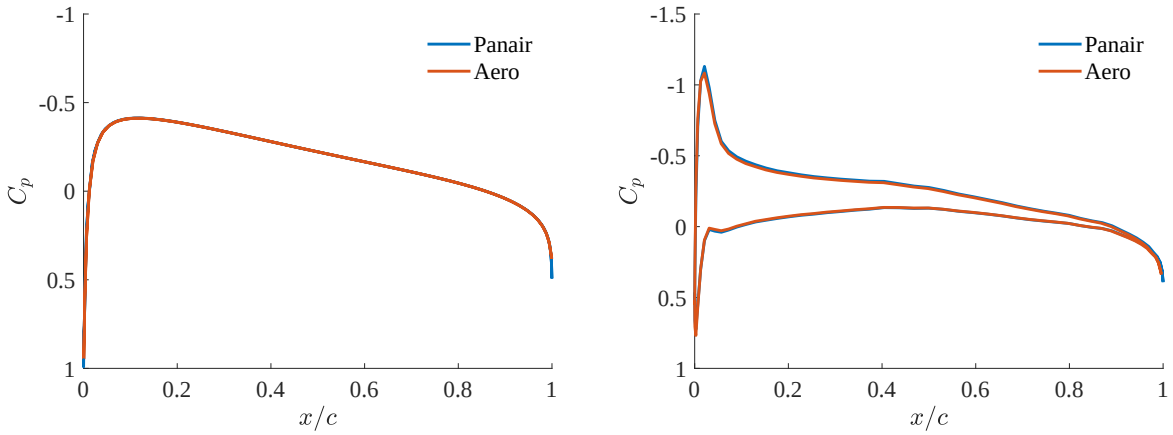
4.3 Validation

In this section, `Aero` is validated by computing the flow on two test cases in different flow regimes. The first test case is a rectangular wing with a constant NACA 0012 airfoil and an aspect ratio of 10 that is used for nonlifting computations. The second wing is the Onera M6 and it is used to test lifting computations. Both cases are tested in incompressible, subcritical and supercritical flow regimes. The predictions are compared to `Tranair`, which solves the nonlinear potential equation, and to `Panair`, which solves the linear potential equation and scales its solution using a compressibility correction. The field panel computations carried out using `Aero` are considered converged when the maximum difference in the field sources between two consecutive iterations drops below 10^{-5}.

4.3.1 Incompressible flow

The flow is first assumed to be incompressible and the Mach number is set to $M = 0$. Under such conditions, the field module is deactivated and only the panel method is tested. Both the NACA and Onera wings have been discretized with 1000 panels: 100 along the chord and 10 along the span. The angle of attack, α, of the NACA 0012 is set to $0°$ to obtain a nonlifting flow while the angle of attack of the Onera M6 is set to $3.06°$.

Figure 4.3.1 shows the pressure distribution along the mean aerodynamic chord of the NACA 0012 and the Onera M6 wings at zero Mach number. The results for both lifting and nonlifting flows perfectly match those obtained with `Panair`, hence demonstrating the validity of the implemented panel method. Since the method is linear, only one iteration is required. On a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz), the computational time required by `Panair` is about 10 seconds, while it is 1 second for `Aero`.



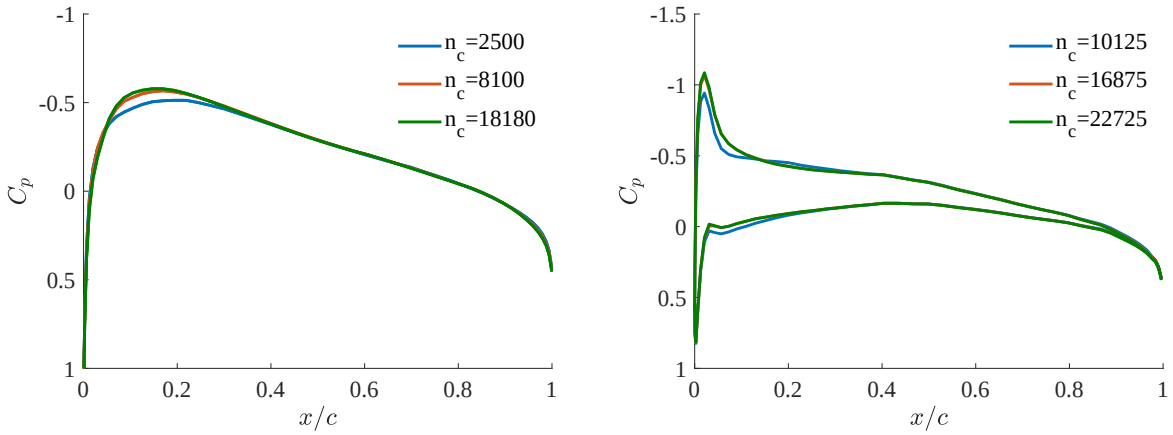
(a) NACA 0012: $\alpha = 0°$, $M = 0.0$.

(b) Onera M6: $\alpha = 3.06°$, $M = 0.0$.

Figure 4.3.1: Pressure distribution along the mean aerodynamic chord of the NACA 0012 and Onera M6 wings at incompressible speed.

4.3.2 Subcritical flow

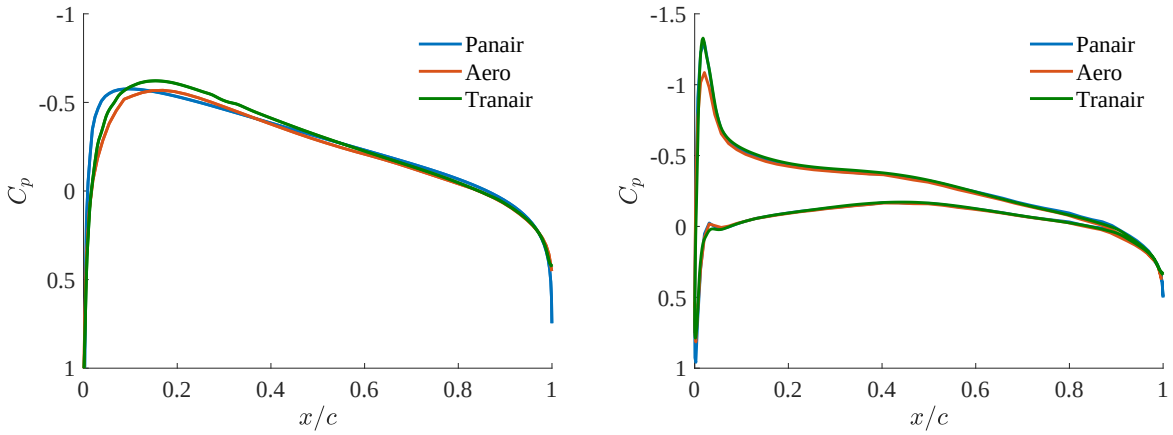
The Mach number is set so that the flow is compressible but remains subcritical for both wings and the angle of attack is kept to $0°$ and $3.06°$, respectively. The surface mesh is maintained to 1000 panels for both wings. In the case where the flow is subcritical, the domain size can be small, as the field sources decay quickly as the distance increases. For both cases, the domain boundaries are placed at 0.5 chord length away of the body in the streamwise and normal directions, and to 0.5 span length away in the spanwise direction. These sizes have been checked *a posteriori* by verifying that the magnitude of the field sources is sufficiently small in the farfield. A convergence study is first performed to find a suitable field mesh size for each wing. Figure 4.3.2 shows the pressure distribution along the mean aerodynamic chord of the NACA 0012 at $M = 0.7$ and the Onera M6 at $M = 0.6$ for a coarse, a medium, and a fine grid. The number of cells is denoted by n_c. In this case, the convergence is reached for the medium-density meshes for both wings. Since the Onera M6 is swept, a denser cell distribution is required along the span.



(a) NACA 0012: $\alpha = 0°$, $M = 0.70$. (b) Onera M6: $\alpha = 3.06°$, $M = 0.60$.

Figure 4.3.2: Pressure distribution along the mean aerodynamic chord of the NACA 0012 and Onera M6 at subcritical speed for three different grids.

Following the convergence study, the fields around the NACA 0012 and the Onera M6 are discretized using $n_c = 8\,100$ and $n_c = 16\,875$ cells, respectively. In both cases, the field cell size is around 4% and 2% of the chord in the streamwise and normal directions, respectively. Figure 4.3.3 shows the pressure distribution along the mean aerodynamic chord of the NACA 0012 and the Onera M6 at compressible but subcritical flow conditions. Globally, the field panel method shows a good agreement with `Tranair` except near the suction peak, which is underestimated. In the NACA case, the predictions are also improved compared to `Panair` linear solution, while it is the opposite in the Onera M6 case. The computations were performed on a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz) and 16 GB of memory. The NACA 0012 case converged in 22 iterations and required 525 MB of memory and 160 seconds of computational time, while the Onera M6 case converged in 12 iterations and required 2.3 GB of memory and 380 seconds of computational time.

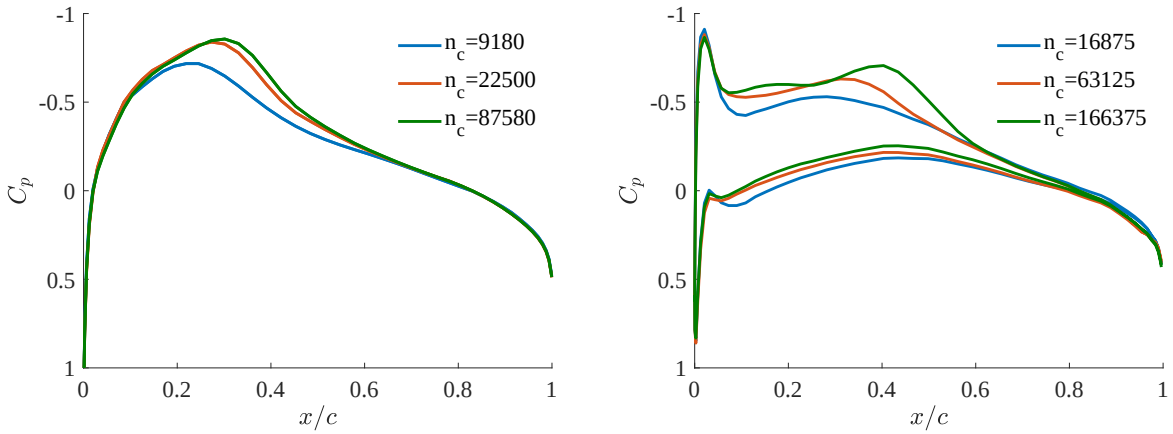


(a) NACA 0012: $\alpha = 0°$, $M = 0.70$. (b) Onera M6: $\alpha = 3.06°$, $M = 0.60$.

Figure 4.3.3: Pressure distribution along the mean aerodynamic chord of the NACA 0012 and Onera M6 at subcritical speed.

4.3.3 Supercritical flow

The Mach number is set so that the flow is supercritical and the angle of attack of both wings is kept to $0°$ and $3.06°$ respectively. The surface mesh is maintained to 1000 panels. Since shocks are expected for these cases, the domain sizes have been increased: they now extend two chord lengths away from the wing in the streamwise and normal directions. These sizes have been validated *a posteriori*, by ensuring that the shocks are fully contained in the domain and do not interact with the farfield boundary, and that the magnitude of the field sources is sufficiently small in the farfield. A convergence study is performed to find a suitable mesh size for each wing. Figure 4.3.4 shows the pressure distribution along the mean aerodynamic chord of the NACA 0012 at $M = 0.8$ and the Onera M6 at $M = 0.839$ for a coarse, a medium, and a fine grid. Since the numerical viscosity, ensuring the stability of the method, is proportional to the grid size, the mesh cannot be indefinitely refined when shocks are present in the solution. The fine grid presented on Figure 4.3.4 is the finest grid for which a converged solution could be obtained. In the NACA case, convergence could be attained on the fine grid, while for the Onera case, the solution on the fine grid still largely differs from the solution on the medium grid. Further to the stability issue, the memory required by the fine grid almost reaches the maximum memory available on the machine.



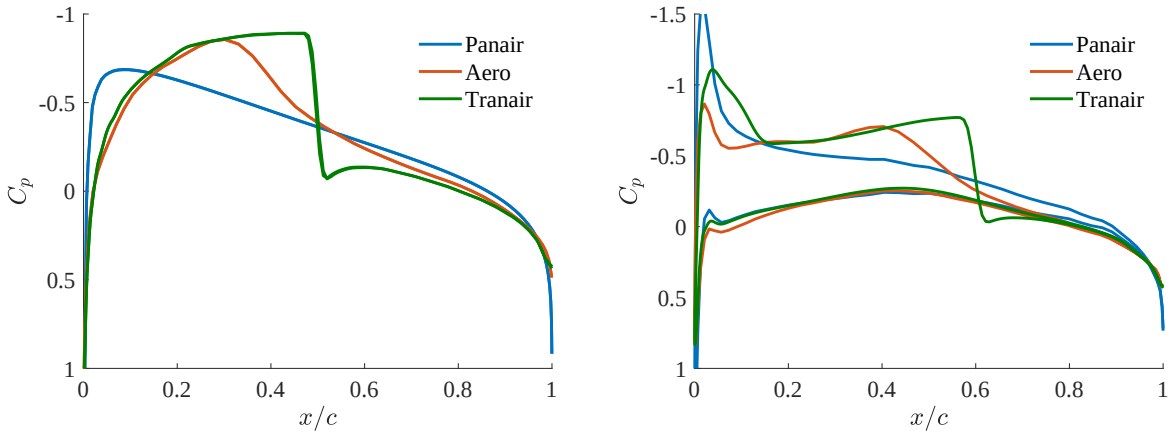
(a) NACA 0012: $\alpha = 0°$, $M = 0.80$.

(b) Onera M6: $\alpha = 3.06°$, $M = 0.839$.

Figure 4.3.4: Pressure distribution along the mean aerodynamic chord of the NACA 0012 and Onera M6 at supercritical speed for three different grids.

Following the convergence study, the finest meshes are retained, and the NACA is discretized with $n_c = 87\,580$ field panels while the Onera grid counts $n_c = 166\,375$ cells. Figure 4.3.5 shows the pressure distribution along the mean aerodynamic chord of the NACA 0012 and the Onera M6 at supercritical conditions. When the flow exhibits a shock, the accuracy of the method is degraded. Figures 4.3.5a and 4.3.5b both show that the field panel method tends to predict a shock that is smeared and displaced upstream compared to `Tranair` full potential solution. Even if the field panel solution shows significant improvement over the linear potential solution predicted by `Panair`, this is at the cost of the computational time and memory required to compute and store the aerodynamic influence coefficients. The NACA 0012 case converged

in 121 iterations and required 65 GB of memory and 10 hours of computational time, while the Onera M6 case converged in 240 iterations and required 225 GB of memory and more than one day of computation time. Since these computations required an extensive amount of memory, they were performed on a cluster equipped with AMD Bulldozer processors (2.1 GHz) and 256 GB of memory per node. This is to be compared to the solution time required by `Tranair`, which is of about five minutes. The tendency to smear the shock, as well as the high memory requirement were also reported by Gebhardt et al. [152].



(a) NACA 0012: $\alpha = 0°$, $M = 0.80$.

(b) Onera M6: $\alpha = 3.06°$, $M = 0.839$.

Figure 4.3.5: Pressure distribution along the mean aerodynamic chord of the NACA 0012 and Onera M6 at supercritical speed.

4.3.4 Challenges and attempted solutions

As shown in Figures 4.3.3 and 4.3.5, `Aero` tends to underestimate the pressure peak at the leading edge and to smear and displace the shock upstream. The underestimation of the pressure peak only occurs on certain geometries. While it is clearly visible on the Onera M6, it is nearly absent on the NACA 0012 wing. It could be due to insufficient local grid refinement. Finer grids could not be tested due to prohibitive computational costs. Implementing a grid stretching capability could solve this issue. The impossibility to accurately capture the shock however poses a serious problem. Computing the field sources from `Tranair`'s solution and using them to correct `Aero`'s incompressible predictions in a single iteration allowed to recover results similar to `Tranair`'s. The issue thus seems to be located in the iterative procedure or in the field module, rather than in the formulation of the method. Several solutions have been attempted to fix the issue, among which an adaptation of the artificial viscosity and an adaptation of the aerodynamic influence coefficient matrices. The adaptation of the artificial viscosity consisted in scaling the switching function by the magnitude of the gradient of the density in Equation 4.2.2. The adaptation of the influence matrices consisted in removing the coefficients related to surface and field panels located upstream of the target cell. However, neither of these solutions improved the results, neither in terms of accuracy nor of computational time.

4.4 Discussion

`Aero`, a field panel method, was developed and implemented to solve the full potential equation, and quickly model transonic flows. The code was tested on two test cases for incompressible, subcritical and supercritical flow regimes. For incompressible and subcritical flows, the agreement between the field panel method and other solution techniques is excellent, except near the pressure peak, which tends to be underestimated. When the flow is supercritical, the shock tends to be smeared and displaced upstream, even though the solution is improved compared to the linear potential prediction. Moreover, the grid refinement needed to capture the shock leads to excessive memory usage and computational time. Several techniques can be used to decrease the computational requirements. A fast multipole method [165, 166] could be implemented. This technique was already combined to panel methods and proved to be efficient [167, 168]. As proposed by Sinclair [147, 148], the boundary and field computations can also be separated. Since Sinclair and Rottegermann [149], who used this approach, also reported better results near shocks, finite difference based computations might also increase the accuracy of the field panel method. However, such techniques are complex to implement, and the algorithmic complexity becomes similar to that of a traditional finite element or volume method.

Despite being a promising, simple and straightforward extension to the panel method, the field panel method suffers from a major drawback. It tends to smear shocks compared to traditional field solvers, such as `Tranair` [152, 159]. Therefore, the field panel method, as it is implemented in `Aero` at least, is only applicable to the design of aircraft which fly at high compressible, but subcritical speeds. In order to accurately capture shocks and to carry out transonic flow computations, a finite element solution of the full potential equation will be developed in the next chapter.

Chapter 5

Finite element solution of the full potential equation

In this chapter, a finite element solution of the full potential equation will be developed and presented. The theory and the implementation will first be described, and computational examples will then be given to illustrate the capabilities of the method as well as its limitations. The full potential equation is elliptic over a large portion of the flow, except in supersonic regions, where it becomes hyperbolic. Since the finite element method is well adapted to solving elliptic partial differential equations, the formulation only needs to be adapted in these supersonic regions. Moreover, the University of Liège has significant expertise in finite element modeling. The finite element method has thus been chosen over the more traditional finite volume method.

5.1 Theory

A finite element formulation relies on two main aspects: the weak form of a problem governed by partial differential equations and their boundary conditions, and its discretization. The weak formulation is first obtained from the strong form of the equations by multiplying it with test functions and integrating over a given domain. The domain is then divided into small elements, onto which the weak formulation is discretized using interpolation functions. The solution to the problem is finally obtained by requiring that the equations are satisfied for any values of the test functions, which is equivalent to seeking a solution in the sense of a distribution.

5.1.1 Weak formulation

Consider a domain Ω enclosed by a surface $\Gamma = \Gamma_\mathrm{u} \cup \Gamma_\mathrm{f} \cup \Gamma_\mathrm{b}$, as depicted in Figure 5.1.1. Assuming the potential ϕ to belong to the Sobolev space, the full potential equation 1.2.9 can be multiplied by a test function ψ, and integrated by parts over the domain Ω. The weak form of the equation reads,

$$\int_\Omega \rho \nabla\phi \cdot \nabla\psi \, dV - \int_\Gamma \overline{\rho\nabla\phi} \cdot \mathbf{n}\psi \, dS = 0, \qquad \forall \psi \in H^1(\Omega), \tag{5.1.1}$$

where the density ρ is given by the isentropic flow relationship 1.2.10, and where $\mathbf{n}$ is the unit vector normal to Γ pointing inwards. $H^1(\Omega)$ denotes the Sobolev space of real square-integrable functions defined over Ω, $L^2(\Omega)$, whose distributional derivative is also real square-

integrable,

$$H^1(\Omega) = \left\{ f \in L^2(\Omega) : \nabla f \in L^2(\Omega) \right\}. \tag{5.1.2}$$

Note that the linear potential equation is recovered if the density is considered to be constant and equal to the freestream density.

Boundary conditions

The boundary surface, Γ, is split into an upstream boundary Γ_u, a farfield boundary Γ_f, and the body boundaries Γ_b, as depicted in Figure 5.1.1. Numerical experiments showed that the solution was less sensitive to Neumann boundary conditions, hence allowing to use smaller domains. However, imposing only such boundary conditions could result in an ill-conditioned set of equations. Consequently, a Neumann boundary condition is applied on the farfield and body boundaries, while a Dirichlet boundary condition is applied on the upstream boundary. The Neumann boundary condition, imposing a flux through the boundaries of the domain, is directly recovered in the second term of the weak formulation of the full potential equation 5.1.1. Since the derivative of the potential is the velocity, the weak form of the Neumann boundary condition can be written as

$$\begin{aligned} \int_{\Gamma_\mathrm{f}} \overline{\rho \nabla \phi} \cdot \mathbf{n} \psi \, dS &= \int_{\Gamma_\mathrm{f}} \rho_\infty \mathbf{U}_\infty \cdot \mathbf{n} \psi \, dS, \\ \int_{\Gamma_\mathrm{b}} \overline{\rho \nabla \phi} \cdot \mathbf{n} \psi \, dS &= 0, \end{aligned} \tag{5.1.3}$$

where ρ_∞ and $\mathbf{U}_\infty$ are the density and the velocity vector in the freestream. The Dirichlet boundary condition is enforced by requiring that the test function ψ vanishes on the upstream boundary, and that

$$\overline{\phi}|_{\Gamma_\mathrm{u}} = \phi_\infty, \tag{5.1.4}$$

where ϕ_∞ is the freestream potential at coordinates (x, y, z), and is defined as

$$\phi_\infty = x \cos\alpha \cos\beta + y \sin\beta + z \sin\alpha \cos\beta, \tag{5.1.5}$$

where α is the angle of attack and β is the angle of sideslip. Equation 5.1.4 and the first equation in Equation 5.1.3 are referred to as farfield boundary conditions, while the second equation in Equation 5.1.3 is referred to as the impermeability boundary condition. If three-dimensional symmetric flows are studied, *i.e.* symmetric configurations at zero angle of sideslip, the boundary Γ is further composed of a symmetry plane Γ_s onto which a symmetry boundary condition is applied. Under potential flow assumptions, this boundary condition reduces to the impermeability boundary condition. A typical two-dimensional domain is illustrated in Figure 5.1.1.

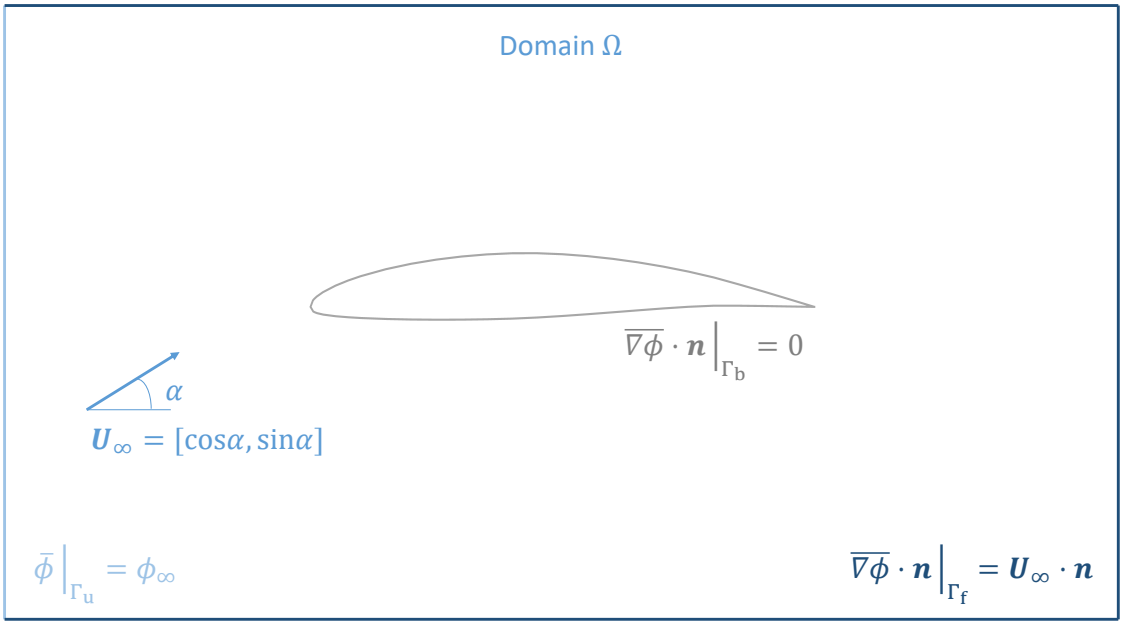


Figure 5.1.1: Typical domain used for a finite element computation.

Kutta condition

As explained in chapter 3, the Kutta condition needs to be enforced to allow potential flows to produce aerodynamic loads. In the present work, the Kutta condition is imposed following Nishida's [12] and Galbraith et al. [13] ideas.

A flat wake sheet, denoted Γ_w and extending from the trailing edge of any lifting body to the farfield boundary located downstream of these bodies, is created. The unknown potential value on this wake is duplicated, hence allowing the potential to be discontinuous across the wake. If no additional boundary condition is explicitly applied on the wake, homogeneous Neumann boundary conditions are naturally enforced on both sides of the wake, and it acts as a solid wall. In order to restore the continuity in the flow, the two supplementary boundary conditions in Equation 1.2.12 must be enforced. The following procedure is similar to the formulation of periodic boundary conditions. The first condition prescribes the equality of the mass flux on the upper and lower sides of the wake,

$$\int_{\Gamma_{w,u}} \rho \nabla\phi \cdot \mathbf{n} \, dS = -\int_{\Gamma_{w,l}} \rho \nabla\phi \cdot \mathbf{n} \, dS, \tag{5.1.6}$$

where subscripts u and l refer to the upper and lower sides of the wake, respectively. Substituting from Equation 5.1.1 in the above equation allows to replace the surface terms by volume terms. As a consequence, continuity in the mass-flux can be enforced by adding the volume terms computed on the upper side of the wake to the volume terms lying on the lower side of

the wake, that is

$$\int_{\Omega_{\mathrm{w,l}}} \rho \nabla\phi \cdot \nabla\psi \, dV + \int_{\Omega_{\mathrm{w,u}}} \rho \nabla\phi \cdot \nabla\psi \, dV = 0, \qquad \forall\psi \in H^1(\Omega_{\mathrm{w,l}}). \tag{5.1.7}$$

The second condition prescribes the equality of the pressure across the wake. Similar to the density, the pressure is given by the isentropic flow relationship,

$$p = p_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - |\nabla\phi|^2\right)\right]^{\frac{\gamma}{\gamma-1}}, \tag{5.1.8}$$

where p_∞ is the freestream pressure and γ is the heat capacity ratio. Since the pressure only depends on the L^2 norm of the velocity, the continuity in the pressure can be written as,

$$\int_{\Gamma_{\mathrm{w}}} (\psi + \Psi) [\![|\nabla\phi|^2]\!] \, dS = 0, \qquad \forall\psi \in H^1(\Gamma_{\mathrm{w,u}}), \tag{5.1.9}$$

where the double square bracket indicates a jump between the quantities on the upper and lower sides of the wake. Equation 5.1.9 introduces a convective term, which can lead to oscillations in the solution. For three-dimensional flows, these oscillations can cause the solution to diverge. In such cases, an effective way to stabilize the solution is to use a Petrov-Galerkin formulation. The test function ψ is then supplemented by an additional term,

$$\Psi = \frac{1}{2} \frac{h}{U_\infty} \left(\mathbf{U}_\infty \cdot \nabla\psi\right), \tag{5.1.10}$$

where U_∞ is the norm of the freestream velocity vector and h is a characteristic length. For two-dimensional flows, Ψ can be set to zero. Equations 5.1.7 and 5.1.9 can further be supplemented by a third equation, reflecting the Kutta condition locally on the trailing edge Γ_{TE},

$$\int_{\Gamma_{\mathrm{TE,u}}} |\nabla\phi|^2 \psi \, dS - \int_{\Gamma_{\mathrm{TE,l}}} |\nabla\phi|^2 \psi \, dS = 0, \qquad \forall\psi \in H^1(\Gamma_{\mathrm{TE,u}}). \tag{5.1.11}$$

Using Equation 5.1.11 allows to align the wake sheet horizontally, which simplifies the geometry generation procedure. However, numerical experiments performed on various wings showed that adding Equation 5.1.11 induces oscillations for three-dimensional flows. Various techniques have been tried to stabilize the flow, such as, normalizing the integral, similarlly to Galbraith et al. [13], using a Petrov-Galerkin stabilization, and using the same discretization on the upper and lower sides of the trailing edge, but none of them produced satisfying results on realistic wing shapes. As a result, Equation 5.1.11 can only be used for two-dimensional flows, and the wake should be aligned with the trailing edge bisector in three-dimensional cases. A parametric study, given in section 5.4, shows that the wake inclination has a non-negligible influence on the results obtained with the present method. Further possibilities of improvement will be discussed in chapter 7.

Supersonic flow treatment

As explained in chapter 3, supersonic regions of the flow need to be stabilized. A density upwinding procedure, similar to Eberle [100] and Hafez et al. [102], is used in the present work. The upwinded density can be written as,

$$\tilde{\rho} = \rho - \mu \overleftarrow{\delta_s} \rho \Delta s. \tag{5.1.12}$$

Details about the computation of the switching function, μ, and the streamline upwind derivative of the density, $\overleftarrow{\delta_s} \rho \Delta s$, will be given in the next section. A flux upwinding procedure, similar to Kinney [129, 130], has also been implemented, but the gain in accuracy was negligible compared to the increase in computational cost and algorithmic complexity.

The implemented density upwinding procedure works well up to local Mach numbers around 1.3, followed by shockwaves. If no shockwave occurs, local Mach numbers of about 2.0 can be reached. Flows featuring such high Mach numbers are not often computed in practice since nonlinear potential theory is theoretically restriced to local Mach numbers below 1.3. However, in some cases, tiny portions of the flow can exhibit a very high local Mach number that will not affect the entire solution, but that can cause divergence of the numerical procedure. As an example, the vortex generated at the wingtip trailing edge of three-dimensional lifting configurations induces a local infinite velocity. For high-speed freestream flows, this translates into a large local Mach number and a low density at this location. Such a flow is illustrated and discussed in appendix D. To alleviate this issue, the density associated to large velocity magnitudes is clamped according to the following Padé approximation [13],

$$\rho = \frac{\rho_{\text{crit}}}{1 + \kappa \left(\frac{U}{U_{\text{crit}}} - 1 \right)}, U > U_{\text{crit}}, \tag{5.1.13}$$

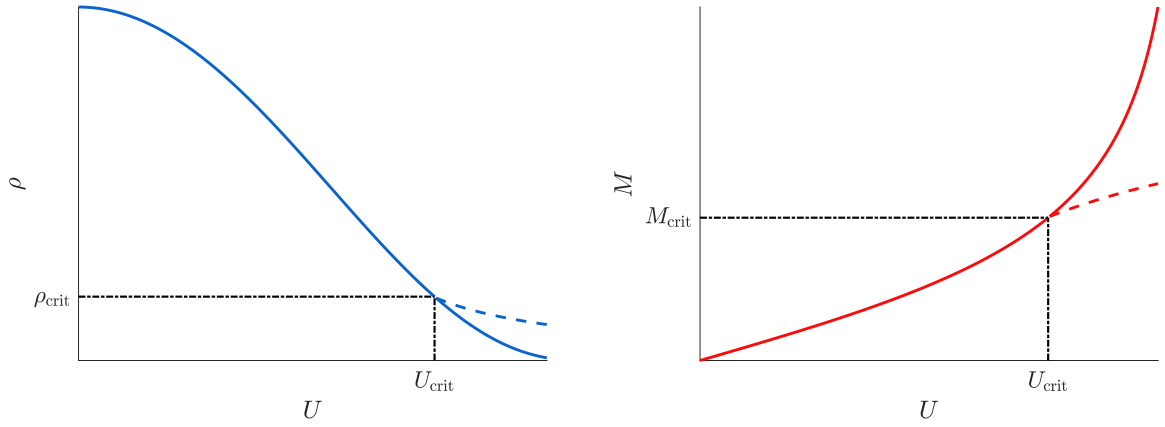
where U is the norm of the velocity, ρ_{crit} is the critical density, U_{crit} is the norm of the critical velocity, and

$$\kappa \equiv \left. \frac{\frac{\partial \rho}{\partial U}}{-\rho} \right|_{U=U_{\text{crit}}} = \frac{M_\infty^2 U_{\text{crit}}}{1 + \frac{\gamma-1}{2} M_\infty^2 \left(1 - U_{\text{crit}}^2\right)}. \tag{5.1.14}$$

Similarly to the density, the Mach number depends solely on the velocity magnitude and is defined as

$$M = \sqrt{\frac{|\nabla \phi|^2}{\frac{1}{M_\infty^2} + \frac{\gamma-1}{2}(1 - |\nabla \phi|^2)}}. \tag{5.1.15}$$

As a result, the critical density and velocity, ρ_{crit} and U_{crit}, can be determined by choosing a critical Mach number, which is a user-specified input, usually set to $M_{\text{crit}} \sim \sqrt{5}$. Figure 5.1.2 compares the dependence of the density and Mach number on the velocity, with and without the Padé approximation, and demonstrates the effectiveness of the density clamping procedure.



(a) Density.

(b) Mach number.

Figure 5.1.2: Evolution of the density and the Mach number as a function of the velocity. The solid line represents the physical values and the dashed line represents the Padé approximation.

Aerodynamic loads

The resultant aerodynamic force coefficient is computed by integrating the normalized pressure force on the body surface,

$$\mathbf{C_F} = \frac{1}{S_{\text{ref}}} \int_{\Gamma_b} C_p \mathbf{n} \, dS, \tag{5.1.16}$$

where S_{ref} is a reference area, and where the pressure coefficient is given by

$$C_p = \frac{2}{\gamma M_\infty^2} \left(\rho^\gamma - 1 \right). \tag{5.1.17}$$

The aerodynamic load coefficients are then obtained by projecting $\mathbf{C_F}$ on the lift, drag and sideforce directions, yielding

$$C_L = \mathbf{C_F} \cdot \mathbf{e_L}, \qquad C_D = \mathbf{C_F} \cdot \mathbf{e_D}, \qquad C_S = \mathbf{C_F} \cdot \mathbf{e_S}, \tag{5.1.18}$$

where the directions are defined with respect to the angle of attack α, and the angle of sideslip β,

$$\mathbf{e_L} = \begin{bmatrix} -\sin\alpha \\ 0 \\ \cos\alpha \end{bmatrix}, \qquad \mathbf{e_D} = \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}, \qquad \mathbf{e_S} = \begin{bmatrix} -\cos\alpha\sin\beta \\ \cos\beta \\ -\sin\alpha\sin\beta \end{bmatrix}. \tag{5.1.19}$$

The pitching moment coefficient is obtained by integrating the y-component of the normalized moment caused by the pressure force on the body surface,

$$C_M = \frac{1}{S_{\text{ref}} c_{\text{ref}}} \int_{\Gamma_b} \left[C_p \mathbf{n} \wedge \mathbf{l} \right] \cdot \mathbf{e_y} \, dS, \tag{5.1.20}$$

where c_{ref} is a reference length, $\mathbf{e_y}$ is a unit vector pointing in the y direction, and l is the lever arm measured from a fixed reference point.

Aeroelastic capability

Since `Flow` is also designed to perform aeroelastic computations, it needs to implement a mesh deformation procedure. An efficient way to deform the grid for the kind of wing deflections considered in practical aeroelasticity, is to use linear elasticity theory. The grid is assumed to behave like an elastic body, rigid near the deforming boundaries, and flexible elsewhere. Moreover, the linear elasticity equations can be easily solved by the finite element method, and require little supplementary implementation work.

For an elastic solid, the equilibrium between the internal and external forces, Equation 1.2.22, can be written in weak form as

$$\int_\Omega \nabla \boldsymbol{\sigma}_\mathrm{s} \cdot \nabla \psi \, dV - \int_\Gamma \overline{\nabla \boldsymbol{\sigma}_\mathrm{s}} \cdot \mathbf{n} \psi \, dS = \int_\Omega \mathbf{f}_\mathrm{s} \, dV, \qquad \forall \psi \in H^1(\Omega), \tag{5.1.21}$$

where the internal stress $\boldsymbol{\sigma}_\mathrm{s}$ can be related to the displacement $\mathbf{u}_\mathrm{s}$ using Hooke's constitutive law for linear isotropic solids,

$$\boldsymbol{\sigma}_\mathrm{s} = \frac{E\nu}{(1+\nu)(1-2\nu)} \mathrm{tr}\left(\nabla \mathbf{u}_\mathrm{s} + \nabla \mathbf{u}_\mathrm{s}^{\mathrm{T}}\right) \mathbf{I} + \frac{E}{1+\nu}\left(\nabla \mathbf{u}_\mathrm{s} + \nabla \mathbf{u}_\mathrm{s}^{\mathrm{T}}\right). \tag{5.1.22}$$

The Young modulus E and the Poisson's ratio ν are constitutive parameters. In the present work, they are set to $1/V$ and 0, respectively, as suggested by Dwight [169]. As a result, the mesh behaves as a linear elastic solid, rigid close to the wing where the elements are small, and flexible in the farfield where the elements are large.

In the mesh deformation framework there is no body force, nor natural Neumann boundary condition. Only a Dirichlet boundary condition is imposed on boundary surfaces. The values of the displacement are fixed to zero, except on the body where they are prescribed according to the body motion,

$$\overline{\mathbf{u}_\mathrm{s}}|_{\Gamma_\mathrm{b}} = \mathbf{u}_\mathrm{b}. \tag{5.1.23}$$

On the wake surface, periodic boundary conditions are used to enforce the continuity of the displacement field across the wake,

$$\begin{gathered} \int_{\Omega_{\mathrm{w,l}}} \nabla \boldsymbol{\sigma}_\mathrm{s} \cdot \nabla \psi \, dV + \int_{\Omega_{\mathrm{w,u}}} \nabla \boldsymbol{\sigma}_\mathrm{s} \cdot \nabla \psi \, dV = 0, \qquad \forall \psi \in H^1(\Omega_{\mathrm{w,l}}), \\ \mathbf{u}_{\mathrm{s}_{\mathrm{w,u}}} - \mathbf{u}_{\mathrm{s}_{\mathrm{w,l}}} = 0. \end{gathered} \tag{5.1.24}$$

5.1.2 Finite element discretization

The domain Ω and its boundary Γ are discretized using continuous Galerkin finite elements. An unstructured grid strategy is chosen in order to easily mesh three-dimensional complex shapes,

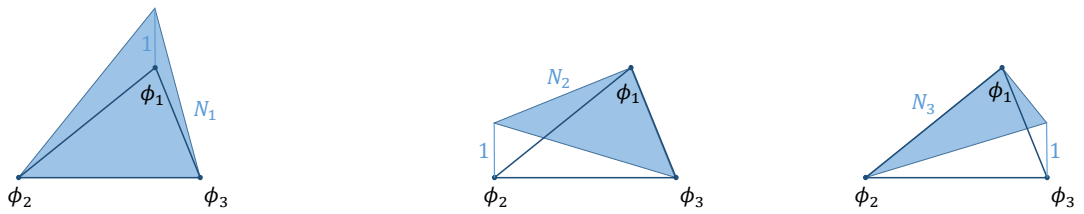
and to easily implement the Kutta condition. In three dimensions, the volume Ω is discretized with linear tetrahedra and Γ is discretized with triangles, while in two dimensions, triangles and lines are used. The potential and test functions are expressed as

$$\begin{aligned} \phi &= N_i \phi_i, \\ \psi &= N_i \psi_i, \end{aligned} \tag{5.1.25}$$

where N_i are the shape functions associated to an element, and interpolate the nodal values ϕ_i and ψ_i of the potential and the test functions on that element. These shape functions are assumed to belong to $H^1(\Omega)$, and have a compact support: they are defined to be nonzero only on an element. Furthermore, they are constrained by the partition of unity, that is

$$\sum_i N_i = 1, \tag{5.1.26}$$

over their support. In particular, a function N_i is defined so that it is equal to 1 on node i and zero on the other nodes. Such shape functions are illustrated in the case of linear triangles in Figure 5.1.3.



(a) N_1. (b) N_2. (c) N_3.

Figure 5.1.3: Shape functions for linear triangle elements.

Following the finite element discretization, the weak formulation of the full potential equation 5.1.1 can be written as

$$\sum_{\mathrm{e}} \int_{\Omega_{\mathrm{e}}} \tilde{\rho}_{\mathrm{e}} \nabla N_j \phi_j \cdot \nabla N_i \psi_i \, dV_{\mathrm{e}} - \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{e}}} \overline{\rho \nabla \phi}_{\mathrm{e}} \cdot \mathbf{n}_{\mathrm{e}} N_i \psi_i \, dS_{\mathrm{e}} = 0, \qquad \forall \psi_i, \tag{5.1.27}$$

where the subscript e refers to elemental quantities. The associated Neumann boundary conditions 5.1.3 become

$$\begin{aligned} \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{f_e}}} \overline{\rho \nabla \phi}_{\mathrm{e}} \cdot \mathbf{n}_{\mathrm{e}} N_i \psi_i \, dS_{\mathrm{e}} &= \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{f_e}}} \rho_\infty \mathbf{U}_\infty \cdot \mathbf{n}_{\mathrm{e}} N_i \psi_i \, dS_{\mathrm{e}}, \\ \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{b_e}}} \overline{\rho \nabla \phi}_{\mathrm{e}} \cdot \mathbf{n}_{\mathrm{e}} N_i \psi_i \, dS_{\mathrm{e}} &= 0. \end{aligned} \tag{5.1.28}$$

The Dirichlet boundary conditions 5.1.4 are imposed as

$$\overline{\phi_i}|_{\Gamma_{\mathrm{u}}} = \phi_\infty. \tag{5.1.29}$$

Kutta condition

Each element and node lying on the wake sheet is duplicated in order to allow the potential to be discontinuous across the wake. For three-dimensional configurations, the potential is continuous at the tip of the wake, *i.e.* the free edge that extends downstream of the wingtip. As a consequence, the nodes lying on this edge are not duplicated. The continuity in the mass flux through the wake is expressed by adding the upper volume terms to the lower shape functions instead of to the upper shape functions. Equation 5.1.7 can then be discretized as

$$\sum_{\mathrm{e}} \int_{\Omega_{\mathrm{w,l_e}}} \tilde{\rho}_{\mathrm{e}} \nabla N_j \phi_j \cdot \nabla N_i \psi_{i_{\mathrm{w,l}}} \, dV_{\mathrm{e}} + \sum_{\mathrm{e}} \int_{\Omega_{\mathrm{w,u_e}}} \tilde{\rho}_{\mathrm{e}} \nabla N_j \phi_j \cdot \nabla N_i \psi_{i_{\mathrm{w,l}}} \, dV_{\mathrm{e}} = 0, \qquad \forall \psi_i \in \Omega_{\mathrm{w,l}}. \tag{5.1.30}$$

Similarly, the continuity in pressure across the wake is expressed by adding the lower surface integral contributions to the upper shape functions. The discretized form of Equations 5.1.9 and 5.1.11 reads

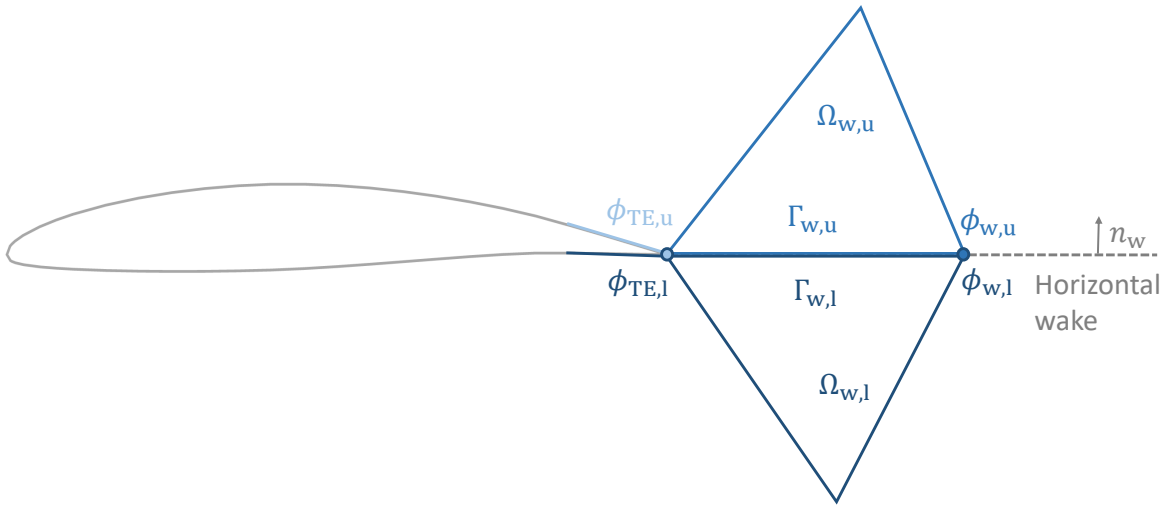
$$\begin{aligned} \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{we}}} (\psi + \Psi)_{\mathrm{w,u}} \left([\nabla\phi \cdot \nabla N_j \phi_j]_{\mathrm{w,u}} - [\nabla\phi \cdot \nabla N_j \phi_j]_{\mathrm{w,l}} \right) dS_{\mathrm{e}} = 0, \qquad & \forall (\psi + \Psi)_{\mathrm{w,u}}, \\ \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{TE,u_e}}} \nabla\phi \cdot \nabla N_j \phi_j \psi_{i_{\mathrm{TE,u}}} \, dS_{\mathrm{e}} - \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{TE,l_e}}} \nabla\phi \cdot \nabla N_j \phi_j \psi_{i_{\mathrm{TE,u}}} \, dS_{\mathrm{e}} = 0, \qquad & \forall \psi_i \in \Gamma_{\mathrm{TE,u}}. \end{aligned} \tag{5.1.31}$$

Note that the second equation is not used for three-dimensional flows. The stabilized shape functions associated with the upper wake surface are computed as,

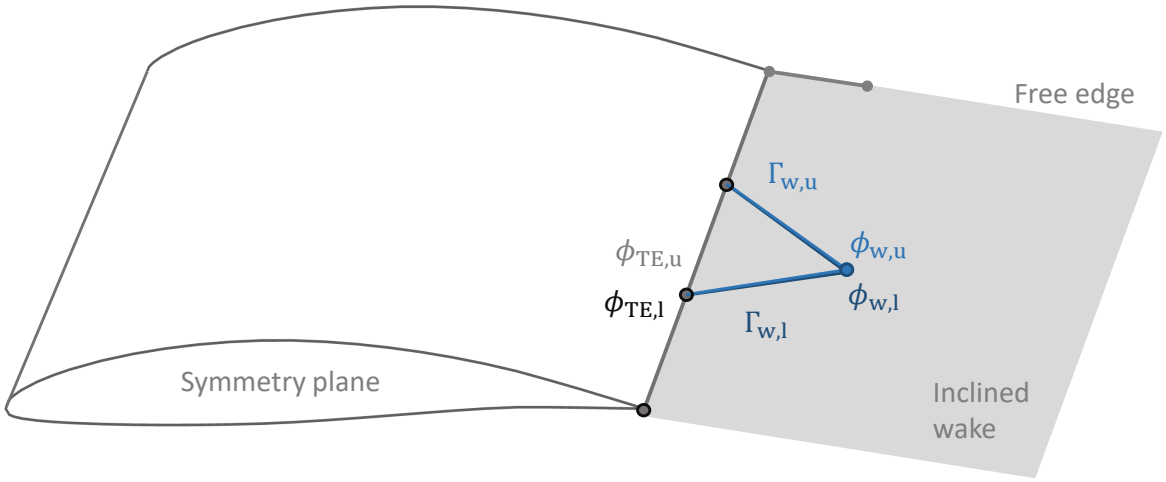
$$(\psi + \Psi)_{\mathrm{w,u}} = N_i \psi_i + \frac{1}{2} \frac{h}{U_\infty} \left(\mathbf{U}_\infty \cdot \nabla N_i \psi_i \right), \qquad \psi \in \Gamma_{\mathrm{w,u}}. \tag{5.1.32}$$

The derivatives of the shape functions ∇N_i appearing in Equations 5.1.31 and 5.1.32 are computed in the volume elements attached to the wake surface Γ_{w}, but are evaluated and integrated on that surface instead.

Figure 5.1.4 shows the discretization used to implement the Kutta condition, both for two and three-dimensional flows. In particular, the degrees of freedom are duplicated on the trailing edge of three-dimensional configurations, but no supplementary condition is enforced there, as opposed to two-dimensional configurations. Also note that the degrees of freedom on the free edge are not duplicated.



(a) Two-dimensional configuration.



(b) Three-dimensional configuration.

Figure 5.1.4: Discretization for the Kutta condition.

Density upwinding

In the implemented density upwinding procedure, the physical density is replaced by an upwinded density which is biased by the streamline upwind derivative of the density, as given by Equation 5.1.12. Two techniques have been implemented to compute the streamline upwind derivative. Firstly, the derivative was computed as in Equation 3.2.10. The term was reconstructed using a Green-Gauss integration procedure, similar to Pelz and Jameson [170]. This

approach is similar to that used in finite volume schemes and is not well adapted to a finite element framework. It was therefore not retained in the current implementation. Secondly, the derivative was approximated by a difference in the density. The upwinded density is then given by

$$\tilde{\rho} = \rho - \mu(\rho - \rho_{\mathrm{U}}), \tag{5.1.33}$$

where the switching function is defined as

$$\mu = \mu_{\mathrm{C}} \max\left(0, 1 - \frac{M_{\mathrm{C}}^2}{M^2}\right). \tag{5.1.34}$$

The constants μ_{C} and M_{C} are controlled by the numerical scheme, as described in the next section. Using finite elements with linear shape functions implies that the density is constant over each element, and allows to easily compute the density at the upwind point, ρ_{U}, appearing in Equation 5.1.33. Several techniques have been tested. The simplest one, consists in taking the density computed on the adjacent element whose centroid is located closest to the opposite main flow direction, *i.e.* the x-direction. This element should be identified only once before starting the computations, but is not the true upwind element, especially near highly curved geometries, *e.g.* wing leading edges. As a result, the domain of dependence of the flow is not correct, and this might lead to inaccurate solutions. An improvement consists in identifying the upwind element exactly. Since the grid is unstructured, this is accomplished by using the local velocity vector of the current element. This method is more accurate, but the flow vector has to be recomputed at each iteration. Yet another approach consists in taking a weighted average of the density computed on all the adjacent elements located upwind of the current element. The three techniques have been tested in the present work. The last two yielded similar results, more accurate than those obtained using the first technique. The second strategy was retained due to its simplicity and is depicted in Figure 5.1.5. The local velocity vector of the current element is depicted in blue. The green and red arrows illustrate the vectors joining the centroid of the current element to the centroid of the adjacent elements. As the element pointed by the green arrow is the closest to the opposite flow direction, depicted by the gray axis, it is retained, while the other elements are discarded. This procedure is restricted to linear elements, but could be extended to higher-order meshes with a fair amount of work. Implementation details will be given in the next section.

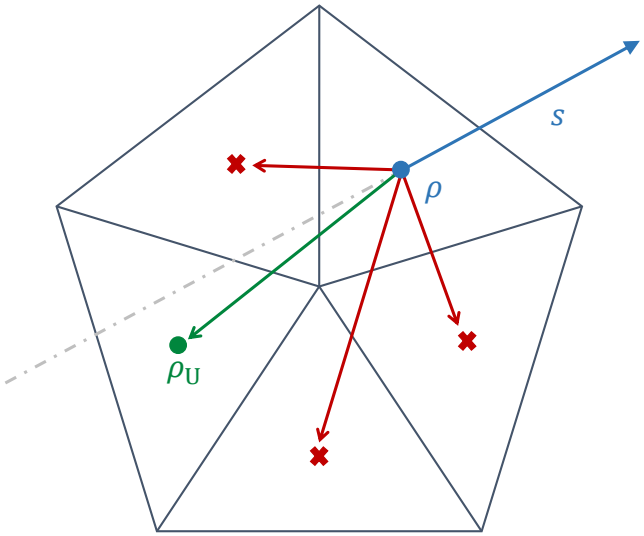


Figure 5.1.5: Identification of upwind elements for density upwinding procedure. The adjacent element whose centroid is located closest to the opposite flow direction, pointed by the green arrow, is retained. The other elements, pointed by red arrows, are discarded.

Grid deformation

Using the same discretization as for the potential equation, the linear elasticity law for the grid deformation can be written as

$$\sum_{\mathrm{e}} \int_{\Omega_{\mathrm{e}}} \left[\frac{E_{\mathrm{e}}\nu_{\mathrm{e}}}{(1+\nu_{\mathrm{e}})(1-2\nu_{\mathrm{e}})} \partial_k N_l u_{\mathrm{s}_{l,k}} \delta_{ij} + \frac{E_{\mathrm{e}}}{1+\nu_{\mathrm{e}}} \left(\partial_j N_l u_{\mathrm{s}_{l,i}} + \partial_i N_l u_{\mathrm{s}_{l,j}} \right) \right] \partial_j N_l \psi_l \, dV_{\mathrm{e}} = 0, \qquad \forall \psi_l, \tag{5.1.35}$$

The Dirichlet boundary condition on the deforming surface are enforced as,

$$\overline{u_{\mathrm{s}_i}}|_{\Gamma_{\mathrm{b}}} = u_{\mathrm{b}_i}. \tag{5.1.36}$$

On the wake, the periodic boundary conditions are discretized as follows. The upper wake volume element contributions are added to the lower wake equations, and the upper wake

unknowns are prescribed to match the lower wake unknowns,

$$\begin{aligned}
&\sum_{\mathrm{e}} \int_{\Omega_{\mathrm{w,l_e}}} \left[\frac{E_\mathrm{e}\nu_\mathrm{e}}{(1+\nu_\mathrm{e})(1-2\nu_\mathrm{e})} \partial_k N_l u_{\mathrm{s}_{l,k}} \delta_{ij} + \frac{E_\mathrm{e}}{1+\nu_\mathrm{e}} \left(\partial_j N_l u_{\mathrm{s}_{l,i}} + \partial_i N_l u_{\mathrm{s}_{l,j}} \right) \right] \partial_j N_l \psi_{l_{\mathrm{w,l}}} \, dV_\mathrm{e} \\
&+ \sum_{\mathrm{e}} \int_{\Omega_{\mathrm{w,u_e}}} \left[\frac{E_\mathrm{e}\nu_\mathrm{e}}{(1+\nu_\mathrm{e})(1-2\nu_\mathrm{e})} \partial_k N_l u_{\mathrm{s}_{l,k}} \delta_{ij} + \frac{E_\mathrm{e}}{1+\nu_\mathrm{e}} \left(\partial_j N_l u_{\mathrm{s}_{l,i}} + \partial_i N_l u_{\mathrm{s}_{l,j}} \right) \right] \partial_j N_l \psi_{l_{\mathrm{w,l}}} \, dV_\mathrm{e} \\
&= 0, \qquad \forall \psi_i \in \Omega_{\mathrm{w,l}}, \\
&u_{\mathrm{s}_j}|_{\Gamma_{\mathrm{w,u}}} - u_{\mathrm{s}_j}|_{\Gamma_{\mathrm{w,l}}} = 0.
\end{aligned} \tag{5.1.37}$$

5.2 Implementation

`Flow` [14, 171], an open-source unstructured finite element method solving the full potential equation, has been developed following the formulation and discretization described in the previous section. The solver is embedded in `waves` [172], a finite element framework developed at the University of Liège. The code consists in C++ classes wrapped in Python modules through `SWIG` [173]. The wrapping allows to take advantage of both the high computing efficiency of the C++ language, and the pre and post-processing capabilities of the Python language, as well as its communication facilities. The code relies on `Eigen` [164] for linear algebra operations and on Intel `Threading Build Blocks` [174, 175] for shared memory parallelization. More details about the code can be found in appendix C.

5.2.1 Geometry modeling and meshing

`Flow` does not rely on any input geometry, but is directly based on a mesh input instead. Currently, the code only supports the `gmsh` [56, 57] native format. As a result, the geometry can be created with any computer-aided design software, but the meshes should be generated by `gmsh`.

Wake modeling

The wake is explicitly modeled as a surface extending from the trailing edge of lifting configurations to the downstream boundary. The wake sheet is meshed in `gmsh` and directly embedded into the volume grid. `Flow` then duplicates each element and node on the wake sheet, except on the free edge, as explained in the previous section. The duplicated, upper and lower, nodes and elements are mapped together such that the contributions of the Kutta conditions can be easily assembled later on. Equations 5.1.31 and 5.1.32 require to evaluate and integrate the shape function derivatives on the wake surface. However, in standard finite element implementations, the derivatives are evaluated only at the Gauss points of volume elements, and are therefore not readily available on the element faces. Fortunately, in the case of linear triangles and tetrahedra, the derivatives of the shape functions are constant over the elements. As a consequence, the derivatives computed at the Gauss points can be directly used to compute

the Kutta condition related terms. This makes the current implementation of the Kutta condition quite simple, but it restricts its application to linear triangular and tetrahedral meshes.

5.2.2 Numerical scheme

Noting that the discretized weak form of the full potential equation 5.1.27 must hold for any test function vector $\boldsymbol{\psi}$, it can be rewritten as a set of equations,

$$\mathbf{R} = 0, \tag{5.2.1}$$

where $\mathbf{R}$ is the residual vector. Since the full potential equation is nonlinear, it needs to be solved in an iterative fashion. A Taylor expansion around a solution vector $\boldsymbol{\phi}_\mathrm{s}$ allows to write

$$0 = \mathbf{R} + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\phi}} \boldsymbol{\Delta\phi} + \mathcal{O}(\boldsymbol{\Delta\phi}^2), \tag{5.2.2}$$

where $\boldsymbol{\Delta\phi} = \boldsymbol{\phi} - \boldsymbol{\phi}_\mathrm{s}$. Neglecting second order terms, and given a known solution estimate $\boldsymbol{\phi}_n$ at iteration n, a better estimate of the solution, $\boldsymbol{\phi}_{n+1}$, can be found by solving

$$\mathbf{J}_n(\boldsymbol{\phi}_{n+1} - \boldsymbol{\phi}_n) = -\mathbf{R}_n, \tag{5.2.3}$$

where the residual vector $\mathbf{R}_n$ is evaluated at $\boldsymbol{\phi}_n$, and the Jacobian matrix $\mathbf{J}_n$ is defined as

$$\mathbf{J}_n = \frac{\partial \mathbf{R}}{\partial \boldsymbol{\phi}}|_{\boldsymbol{\phi}_n}. \tag{5.2.4}$$

Two iterative schemes are implemented in `Flow` to solve Equation 5.2.3: the Picard iteration with relaxation, and a quasi-Newton algorithm combined with line search. If the density is considered to be constant, the potential equation becomes linear. However, the Kutta condition being nonlinear, the problem still needs to be solved iteratively. A linear set of equations must be solved at each iteration of the Picard and quasi-Newton algorithms. For the mesh sizes considered in the present work, direct or iterative linear solvers can be used. Intel MKL `Pardiso` [176], `MUMPS` [177] and a standard implementation of the GMRES alogirthm [114] are available in the current implementation. The results presented in this thesis are computed using `Pardiso`, and a comparison of the different linear solvers is available in appendix D.

The contributions of the Neumann boundary condition and the Kutta condition to the potential equation can be directly embedded in the Jacobian matrix and in the residual vector. The Dirichlet boundary condition is enforced in two steps. If the degree of freedom i lies on a surface onto which a Dirichlet boundary condition needs to be enforced, the i^{th} element of vector $\boldsymbol{\phi}$ is initialized to the imposed value. The i^{th} line of the Jacobian matrix is then filled with zeros, except for the diagonal element, which is set to 1, and the i^{th} element of the residual vector is set to zero. In this way, the degrees of freedom corresponding to a Dirichlet boundary condition are effectively pinned to the initially prescribed value.

Following the standard finite element approach, the shape functions and their derivatives are

first computed at the element level. More specifically, they are evaluated at each Gauss point inside an element, using a reference frame attached to that element. They are then combined to form the different terms of the potential equation and the boundary conditions, and integrated over the element volume using the Gauss integration procedure. The contribution of each element is finally assembled into the Jacobian matrix and the residual vector.

Picard iteration scheme

In the Picard, or fixed-point, iteration scheme, the residual vector is split into a part that depends on the solution and a constant part,

$$\mathbf{R}_n = \mathbf{A}_n \boldsymbol{\phi}_n - \mathbf{b}. \tag{5.2.5}$$

The Jacobian matrix is then chosen to be

$$\mathbf{J}_n = \mathbf{A}_n, \tag{5.2.6}$$

so that Equation 5.2.3 reduces to

$$\mathbf{A}_n \boldsymbol{\phi}_{n+1} = \mathbf{b}, \tag{5.2.7}$$

which is simply Equation 5.1.27 written in matrix form.

The entries of the matrix $\mathbf{A}$ are given by

$$A_{ij} = \int_{\Omega_\mathrm{e}} \rho_\mathrm{e} \partial_k N_j \partial_k N_i \, dV_\mathrm{e}, \tag{5.2.8}$$

and the entries of the vector $\mathbf{b}$ are given by

$$b_i = \int_{\Gamma\mathrm{e}} \overline{\rho \partial_k \phi}_\mathrm{e} \cdot n_{k_\mathrm{e}} N_i \, dS_\mathrm{e}. \tag{5.2.9}$$

The Kutta condition is enforced in two steps. Firstly, the contributions of the upper wake nodes are directly added to the rows of matrix $\mathbf{A}$ and vector $\mathbf{b}$ corresponding to lower wake nodes, except for the nodes lying on the trailing edge. Note that no contribution is assembled on the upper wake rows at this step. This enforces the continuity in the mass flux through the wake,

$$\begin{aligned} A_{ij}|_{\mathrm{w,l}} &= A_{ij}|_{\mathrm{w,l}} + A_{ij}|_{\mathrm{w,u}}, \\ b_i|_{\mathrm{w,l}} &= b_i|_{\mathrm{w,l}} + b_i|_{\mathrm{w,u}}. \end{aligned} \tag{5.2.10}$$

Secondly, the upper wake rows are replaced by the first equation in 5.1.31. The following terms are thus assembled to the upper wake rows,

$$A_{ij}|_{\mathrm{w,u}} = \int_{\Gamma_{\mathrm{we}}} \left(N_i + \frac{1}{2} \frac{h}{U_\infty} U_{\infty,k} \partial_k N_i \right)_{\mathrm{w,u}} \left([\partial_k \phi \partial_k N_j]_{\mathrm{w,u}} - [\partial_k \phi \partial_k N_j]_{\mathrm{w,l}} \right) dS_\mathrm{e}. \tag{5.2.11}$$

For two-dimensional flows, the upper trailing edge node contributions are also added to the

lower trailing edge rows, and the following terms are also added to the upper trailing edge rows,

$$A_{ij}|_{\mathrm{TE,u}} = \int_{\Gamma_{\mathrm{TE,u_e}}} \partial_k\phi\partial_k N_j N_{i_{\mathrm{TE,u}}}\, dS_\mathrm{e} - \int_{\Gamma_{\mathrm{TE,l_e}}} \partial_k\phi\partial_k N_j N_{i_{\mathrm{TE,u}}}\, dS_\mathrm{e}. \tag{5.2.12}$$

An under-relaxation technique has also been implemented in order to increase the stability of the Picard scheme. At each iteration, the solution is updated as,

$$\boldsymbol{\phi}_{n+1} = \lambda\boldsymbol{\phi}_{n+1} + (1-\lambda)\boldsymbol{\phi}_n \tag{5.2.13}$$

where λ is a user-defined relaxation parameter. In the case where $\lambda = 1$, the standard fixed-point iteration scheme is recovered. In practice, Picard's algorithm is simple to implement, but it has poor convergence characteristics. If shocks are present in the solution, it might not even converge, no matter the amount of relaxation used. In the current implementation, the algorithm is restricted to the computation of compressible, but not transonic, flows, and is mainly used for testing and debugging purposes.

Quasi-Newton algorithm

In the Newton-Raphson method, the Jacobian matrix is calculated either analytically or numerically. In the latter case, the matrix terms are obtained by introducing small perturbations in each degree of freedom of the solution vector, at each iteration. As such, the matrix entries must not be explicitly derived analytically, and the procedure is simple to implement. However, this is at the expense of computational performance. In the present implementation, the analytic approach has been chosen in order to reduce the computational requirements as much as possible. The Jacobian matrix is obtained by differentiating the weak form of the full potential equation 5.1.1,

$$\begin{aligned} J = \frac{\partial R}{\partial \phi} = &= \int_\Omega \frac{\partial}{\partial\phi}\left\{\tilde{\rho}\right\}\nabla\phi\cdot\nabla\psi\, dV + \int_\Omega \tilde{\rho}\frac{\partial}{\partial\phi}\left\{\nabla\phi\cdot\nabla\psi\right\} dV \\ &= \int_\Omega (1-\mu)\frac{\partial}{\partial\phi}\left\{\rho\nabla\phi\cdot\nabla\psi\right\} dV \\ &+ \int_\Omega \mu\left(\frac{\partial}{\partial\phi}\left\{\rho_\mathrm{U}\right\}\nabla\phi\cdot\nabla\psi + \rho_\mathrm{U}\frac{\partial}{\partial\phi}\left\{\nabla\phi\cdot\nabla\psi\right\}\right) dV \\ &+ \int_\Omega -(\rho-\rho_\mathrm{U})\frac{\partial}{\partial\phi}\left\{\mu\right\}\nabla\phi\cdot\nabla\psi\, dV, \end{aligned} \tag{5.2.14}$$

where the different derivatives are given by

$$\begin{aligned} \frac{\partial}{\partial\phi}\left\{\rho\right\} &= -M_\infty^2\rho^{2-\gamma}\nabla\phi\cdot\frac{\partial}{\partial\phi}\left\{\nabla\phi\right\}, \\ \frac{\partial}{\partial\phi}\left\{\mu\right\} &= \mu_\mathrm{C}M_\mathrm{C}^2\left(\frac{2}{M^3}\frac{\partial}{\partial\phi}\left\{M\right\}\right), \\ \frac{\partial}{\partial\phi}\left\{M\right\} &= M\left[\frac{1}{|\nabla\phi|^2} + \frac{\gamma-1}{2}\frac{1}{a^2}\right]\nabla\phi\cdot\frac{\partial}{\partial\phi}\left\{\nabla\phi\right\}. \end{aligned} \tag{5.2.15}$$

The speed sound is computed as

$$a = \sqrt{\frac{1}{M_\infty^2} + \frac{\gamma - 1}{2}\left(1 - |\nabla\phi|^2\right)}. \tag{5.2.16}$$

Note that the derivative of the switching function μ is only computed in supersonic regions, where it is nonzero and continuous. Using finite element discretization, Equation 5.2.14 becomes

$$\begin{aligned} J_{ij} = &\int_{\Omega_\mathrm{e}} (1-\mu)\left[-M_\infty^2 \rho_\mathrm{e}^{2-\gamma} \partial_k\phi\partial_k N_j \partial_k\phi\partial_k N_i + \rho_\mathrm{e}\partial_k N_j \partial_k N_i\right] dV_\mathrm{e} \\ &+ \int_{\Omega_\mathrm{e}} \mu\left[-M_\infty^2 \rho_\mathrm{U}^{2-\gamma} \partial_k\phi_\mathrm{U}\partial_k N_{j_\mathrm{U}} \partial_k\phi\partial_k N_i + \rho_\mathrm{U}\partial_k N_j \partial_k N_i\right] dV_\mathrm{e} \\ &- \int_{\Omega_\mathrm{e}} (\rho_\mathrm{e} - \rho_\mathrm{U})\left[\frac{2\mu_\mathrm{C} M_\mathrm{C}^2}{M_\mathrm{e}^3}\left(\frac{1}{\sqrt{\partial_k\phi^2 a_\mathrm{e}^2}} + \frac{\gamma-1}{2}\frac{\sqrt{\partial_k\phi^2}}{\sqrt[3]{a_\mathrm{e}^2}}\right)\partial_k\phi\partial_k N_j \partial_k\phi\partial_k N_i\right] dV_\mathrm{e}, \end{aligned} \tag{5.2.17}$$

where subscript U refers to the upwind element. Note that the analytic form of the Jacobian was verified by comparing it to the numerical Jacobian obtained using small perturbations. The residual vector is written as

$$R_i = \int_{\Omega_\mathrm{e}} \left[(1-\mu)\,\rho_\mathrm{e} + \mu\rho_\mathrm{U}\right]\partial_k\phi\partial_k N_i \, dV_\mathrm{e} - \int_{\Gamma_\mathrm{e}} \overline{\rho\partial_k\phi n_k} N_i \, dS_\mathrm{e}. \tag{5.2.18}$$

The Kutta condition is enforced in a similar way as for the Picard iteration scheme. The contributions of the upper wake nodes are directly added to the lower wake rows, instead of the upper wake rows, of the Jacobian matrix and residual vector,

$$\begin{aligned} J_{ij}|_{\mathrm{w,l}} &= J_{ij}|_{\mathrm{w,l}} + J_{ij}|_{\mathrm{w,u}}, \\ R_i|_{\mathrm{w,l}} &= R_i|_{\mathrm{w,l}} + R_i|_{\mathrm{w,u}}. \end{aligned} \tag{5.2.19}$$

The following terms are then assembled on the upper wake rows,

$$\begin{aligned} J_{ij}|_{\mathrm{w,u}} &= 2\int_{\Gamma_{\mathrm{we}}} \left(N_i + \frac{1}{2}\frac{h}{U_\infty} U_{\infty,k}\partial_k N_i\right)_{\mathrm{w,u}} \left([\partial_k\phi\partial_k N_j]_{\mathrm{w,u}} - [\partial_k\phi\partial_k N_j]_{\mathrm{w,l}}\right) dS_\mathrm{e}, \\ R_i|_{\mathrm{w,u}} &= \int_{\Gamma_{\mathrm{we}}} N_{i_{\mathrm{w,u}}} \left([\partial_k\phi\partial_k\phi]_{\mathrm{w,u}} - [\partial_k\phi\partial_k\phi]_{\mathrm{w,l}}\right) dS_\mathrm{e}. \end{aligned} \tag{5.2.20}$$

For two-dimensional flows, the trailing edge rows are also changed in a similar way,

$$\begin{aligned} J_{ij}|_{\mathrm{TE,u}} &= 2\int_{\Gamma_{\mathrm{TE,u_e}}} \partial_k\phi\partial_k N_j N_{i_{\mathrm{TE,u}}} dS_\mathrm{e} - 2\int_{\Gamma_{\mathrm{TE,l_e}}} \partial_k\phi\partial_k N_j N_{i_{\mathrm{TE,u}}} dS_\mathrm{e}, \\ R_i|_{\mathrm{TE,u}} &= \int_{\Gamma_{\mathrm{TE,u_e}}} \partial_k\phi\partial_k\phi N_{i_{\mathrm{TE,u}}} dS_\mathrm{e} - \int_{\Gamma_{\mathrm{TE,l_e}}} \partial_k\phi\partial_k\phi N_{i_{\mathrm{TE,u}}} dS_\mathrm{e}. \end{aligned} \tag{5.2.21}$$

The Newton-Raphson method converges more rapidly than Picard's iteration scheme, and exhibits a second-order convergence rate as it gets closer to the solution. However, it might be

unstable for transonic flow computations where the local Mach numbers are high. An effective way to stabilize the Newton method is to restrict the change in the solution using a line search procedure. In such a technique, the new solution vector is computed as

$$\boldsymbol{\phi}_{n+1} = \boldsymbol{\phi}_n + \lambda_n(\boldsymbol{\phi}_{n+1} - \boldsymbol{\phi}_n), \tag{5.2.22}$$

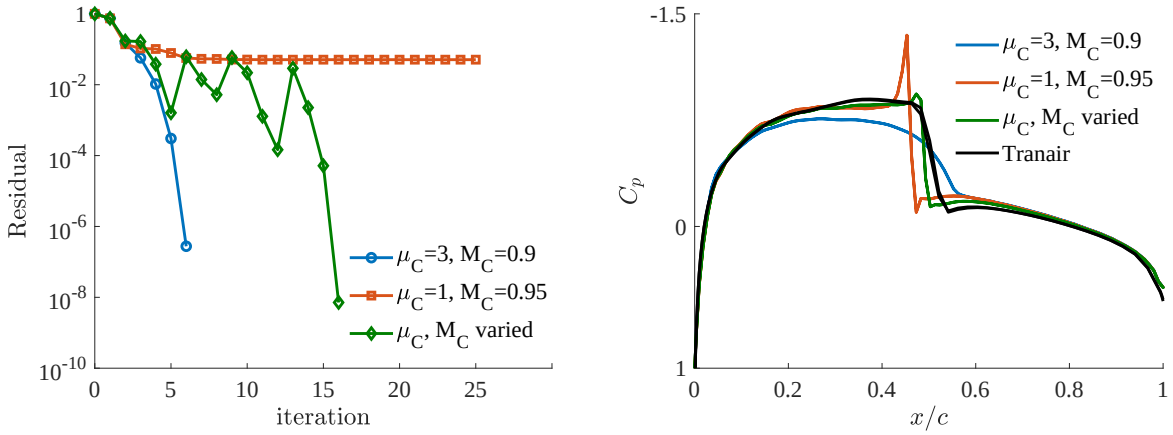
where λ_n is the step length of the line search. A quadratic line search [178] and the Bank and Rose [47] algorithms have been implemented to find the optimal λ_n for a given iteration. The former is more robust, but requires more function evaluations to compute the step length than the latter. For smooth meshes, the Rose and Bank algorithm is usually sufficient. The results presented in chapters 5 and 6 are computed with this line search algorithm. Both algorithms are shortly described in appendix C and compared in appendix D.

Density upwinding

Equation 5.1.33 requires to compute the density at an upwind point, for each element located in supersonic regions. Since linear shape functions are used in the finite element discretization, only one upwind element needs to be associated with each current element, as described in Figure 5.1.5. As the supersonic regions are not known *a priori*, and can vary during the computation procedure, an upwind element is first associated to any volume element of the mesh. As a result, the finite element data structure needs to be enriched, such that each element knows its direct neighbors. This operation is handled by C++ *Standard Template Library* containers and is very fast, even for large meshes. During the iterative procedure, the upwind element is determined by minimizing the scalar product between its direction relative to the current element and the local velocity vector. In practice, the local velocity vector does not vary greatly between two consecutive iterations. As a result, the upwind element can be updated infrequently. An adaptive density upwinding is implemented to improve the convergence and stability of the Newton method for transonic flow computations. The parameters μ_C and M_C of the switching function 5.1.34 are allowed to vary. As explained in chapter 3, the first parameter controls the amplification of the density bias, while the second controls the extent of the region where the bias is to be applied. In the current implementation, they are initialized to 2 and 0.92 in order to produce strong stabilization over a large portion of the flow. As the solution converges, these parameters are varied to 1 and 0.95. These final values were chosen from the literature, as they are suitable for most cases. The update frequency of the parameters is a user-defined input. Changing both parameters each time the relative residual of the full potential equation drops below 10^{-2} or 10^{-3} works usually well. Since changing the density upwinding has an impact on the solution, the upwind elements are identified each time the parameters vary.

The flow around a NACA 0012 airfoil is computed at zero angle of attack and Mach 0.8 to illustrate the effect of using variable parameters to bias the density. Figure 5.2.1a shows the evolution of the relative residual, computed as the L^2 norm of the unknown potential vector, as a function of the iteration count, while Figure 5.2.1b shows the pressure distribution along the chord of the airfoil. μ_C and M_C are first set to 3 and 0.90 to produce a high amount of bias

(blue curve). Figure 5.2.1 shows that the algorithm converges quite fast to a solution in which the shock is smeared and displaced upstream. On the other hand, setting the parameters directly to 1 and 0.95 to produce a low amount of bias (red curve) prevents the scheme from converging, and the obtained solution is oscillatory near the shock. Varying the parameters (green curve) allows the solution procedure to converge and yields a solution close to the `Tranair` predictions used as reference.

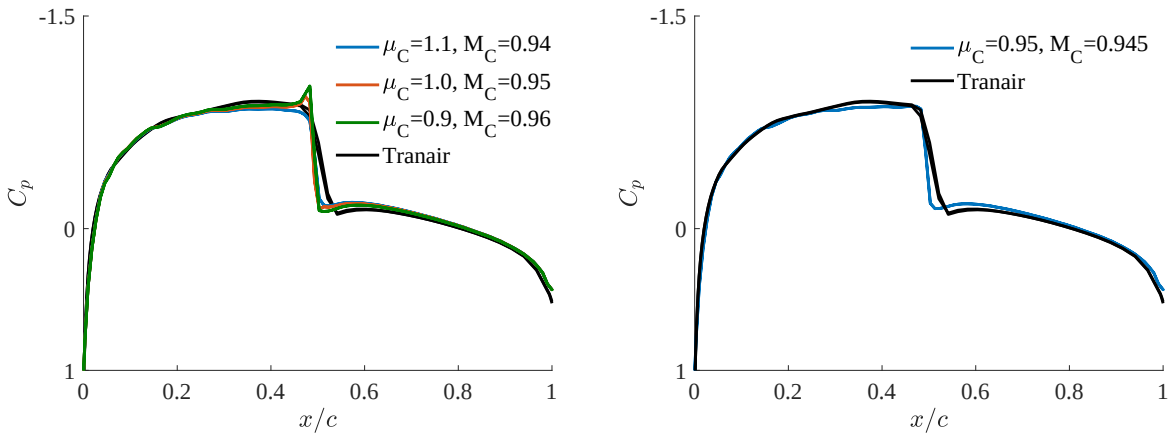


(a) Relative residual as a function of iteration count.

(b) Pressure distribution along the chord.

Figure 5.2.1: Effect of varying the upwinding parameters during the solution procedure. Flow around a NACA 0012 airfoil at zero angle of attack and Mach 0.8 compared to `Tranair`'s results.

In the adaptive density upwinding strategy described above, the final values taken by the parameters of the switching function still have to be chosen, as they will influence the results of a computation. Figure 5.2.2 shows the pressure distribution along the chord of a NACA 0012 airfoil at zero angle of attack and Mach 0.8 for different final values of the parameters μ_C and M_C. The results obtained using `Tranair` are also given for reference. For a given grid density, changing the values of the parameters allows to fine tune the strength and the location of the shock. In the present test case and for a grid size of $1/100$ of the chord on the airfoil surface, the optimal set of parameters has been found to be $\mu_C = 0.95$ and $M_C = 0.945$, as illustrated in Figure 5.2.2b. In practice however, the optimal values of the parameters are case-dependent and sensitive to the grid density, hence difficult to choose *a priori*. Furthermore, changing the grid density has a similar effect on the results, as will be demonstrated in section 5.4. Consequently, $\mu_C = 1.0$ and $M_C = 0.95$ have been retained as final values in the current implementation of `Flow`.



(a) Sensitivity analysis.

(b) Optimal parameters.

Figure 5.2.2: Pressure distribution along the chord of a NACA 0012 airfoil at zero angle of attack and Mach 0.8 for several final values taken by the parameters of the switching function and compared to `Tranair`'s results.

5.3 Validation

In this section, the solver implemented in `Flow` is tested and verified on various two and three-dimensional configurations. Convergence analyses are first performed, and the code is then compared to the full potential solver `Tranair`.

5.3.1 Domain and mesh convergence analyses

Convergence analyses are fist performed on both the domain and the mesh sizes. To demonstrate the convergence, transonic flow computations are performed on the RAE 2822 airfoil and the Onera M6 wing. The variation of the lift and drag coefficients, as well as of the location of the shock, with mesh size is studied. For the wing, the shock location is taken along the mean aerodynamic chord. In Table 5.3.1a, the domain size is noted in terms of its extent in the streamwise and normal directions, in chord lengths, respectively denoted by n_{X} and n_{Z}. In Table 5.3.2a, the domain extent is given in the streamwise, spanwise and normal directions in multiples of the root chord, span and root chord length, respectively, denoted by n_{X}, n_{Y} and n_{Z}. As the domain size is increased, the mesh size is also increased in order to maintain the same density. In Tables 5.3.1b and 5.3.2b, the mesh size is given in terms of the number of cells n_{C}.

RAE 2822 airfoil

Table 5.3.1 shows the lift c_l and drag c_d coefficients, as well as the shock location, given as a fraction of the chord $\frac{x_{\mathrm{S}}}{c}$, on the RAE 2822 airfoil at an angle of attack $\alpha = 2^{\circ}$ and a Mach number $M = 0.715$, for different domain and mesh sizes. For this two-dimensional case, Table 5.3.1a shows that using the second domain is enough to obtain a converged solution in terms of drag and shock location, though the lift still varies slightly. This suggests to place the boundaries 5 chord lengths away from a lifting configuration in two-dimensional cases. The solution obtained

using the second mesh in Table 5.3.1b is also converged in terms of drag and shock location, but as in the domain convergence study case, the lift still varies. The variation in the lift is mainly due to the difference in shock strengths captured using various domain and grid sizes. The second mesh is obtained by using a cell size of 0.01 of the chord on the airfoil surface and of 1 chord in the farfield. Note that refining the mesh further than the third mesh leads to divergence of the method. This occurs because the density upwinding process ensuring the stability of transonic flow computations is proportional to the mesh size. To complement this study, a mesh sensitivity analysis is given in section 5.4.

$n_\mathrm{X} \times n_\mathrm{Z}$	c_l	c_d	$\frac{x_\mathrm{s}}{c}$
5×4	0.896	0.0058	0.63
11×10	0.818	0.0025	0.56
21×20	0.811	0.0023	0.55

(a) Domain convergence (using mesh density corresponding to second mesh in Tab.b).

n_C	c_l	c_d	$\frac{x_\mathrm{s}}{c}$
$2\,000$	0.773	0.0034	-
$6\,000$	0.818	0.0025	0.56
$13\,000$	0.827	0.0025	0.57

(b) Mesh convergence (using second domain in Tab.a).

Table 5.3.1: Convergence analysis for the flow around a RAE 2822 airfoil at 2° angle of attack and Mach 0.715.

Onera M6 wing

Table 5.3.2 shows the lift C_L and drag C_D coefficients, as well as the shock location, noted as a fraction of the mean aerodynamic chord $\frac{x_\mathrm{s}}{c}|_\mathrm{MAC}$, on the Onera M6 wing at an angle of attack of 3.06° and a Mach number of 0.839, for different domain and mesh sizes. For this three-dimensional case, using the second domain in Table 5.3.2a yields a converged solution and suggests to place the domain boundaries 3 chord lengths away from the wing in the streamwise and normal directions, and 1 span length in the spanwise direction. Using the second mesh in Table 5.3.2b, obtained by using a local cell size of 0.005 and 0.01 of the chord at the leading and trailing edges of the wing, and of about 1 chord length in the farfield, is sufficient to get a converged solution. As in the two-dimensional case, note that using a finer mesh prevents the method from converging.

$n_\mathrm{X} \times n_\mathrm{Y} \times n_\mathrm{Z}$	C_L	C_D	$\frac{x_\mathrm{s}}{c}\|_\mathrm{MAC}$
$6 \times 1.5 \times 5$	0.312	0.0126	0.52
$8 \times 2 \times 7$	0.294	0.0111	0.61
$14 \times 5 \times 13$	0.289	0.0111	0.61

(a) Domain convergence (using mesh density corresponding to second mesh in Tab.b).

n_C	C_L	C_D	$\frac{x_\mathrm{s}}{c}\|_\mathrm{MAC}$
$200\,000$	0.273	0.0101	0.54
$600\,000$	0.294	0.0111	0.61
$1\,000\,000$	0.302	0.0114	0.60

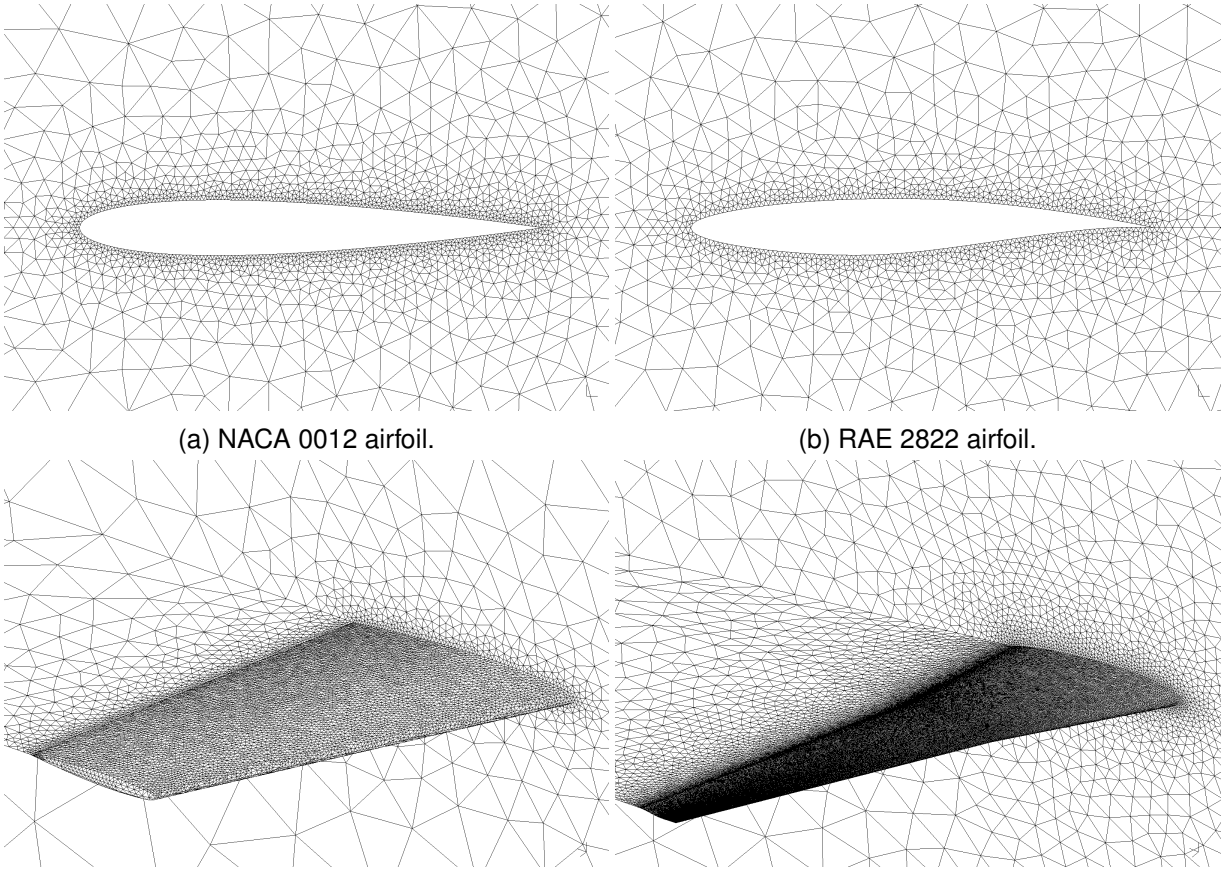
(b) Mesh convergence (using second domain in Tab.a)).

Table 5.3.2: Convergence analysis for the flow around the Onera M6 wing at 3.06° angle of attack and Mach 0.839.

5.3.2 Aerodynamic loads

Following the convergence study, the results presented in the rest of the present chapter are obtained on two-dimensional domains for which boundaries are placed 5 chord lengths away

from the airfoil, and on three-dimensional domains for which boundaries are placed 3.5 chord lengths away from the wing in the streamwise and normal directions and 1 span length away in the spanwise direction. The mesh size ranges from 0.005 to 0.01 of the chord on the surface of the airfoils and wings, and is set to 1 chord length in the farfield. The computational grids used for the different airfoils and wings in the present chapter are given in Figure 5.3.1.



(a) NACA 0012 airfoil.

(b) RAE 2822 airfoil.

(c) Onera M6 wing.

(d) Embraer benchmark wing.

Figure 5.3.1: Computational grids used for the different airfoils and wings.

NACA 0012 and RAE 2822 airfoils

The aerodynamic load coefficients acting on the NACA 0012 airfoil at an angle of attack $\alpha = 0°$ and Mach $M = 0.8$ are given in Table 5.3.3a, and the pressure distribution along the chord of the airfoil is shown in Figure 5.3.2a. For this flow, the lift should be zero and the shock should lie at 50% of the chord, as predicted by `Tranair`. Both results are recovered by `Flow`. For a two-dimensional transonic computation, the drag is only produced through the shock, and its magnitude varies with the shock strength. Compared to `Tranair`, `Flow` consistently predicts a slightly weaker shock, and slightly less drag. Another comparison is performed on the RAE 2822 airfoil. The flow is computed at $\alpha = 2°$ and $M = 0.715$. For this lifting flow, `Flow` tends to predict a lower lift coefficient, as shown in Table 5.3.3b. This lower lift can be related to the weaker shock and the lower magnitude of the pressure coefficient upstream of the shock predicted by `Flow`, as illustrated in Figure 5.3.2b. In both cases, the overall results obtained

using `Flow` compare well with those obtained with `Tranair`. However, `Flow` tends to predict weaker shocks than `Tranair`. This is mainly due to the different strategies implemented in the two solvers to stabilize transonic flow computations, and to the different types of mesh used by the two solvers. For two-dimensional computations, weaker shock predictions are found to have a slight impact on the drag and a noticeable impact on the lift. This difference in lift coefficient can also be attributed to the different implementations of the Kutta condition in the two codes.

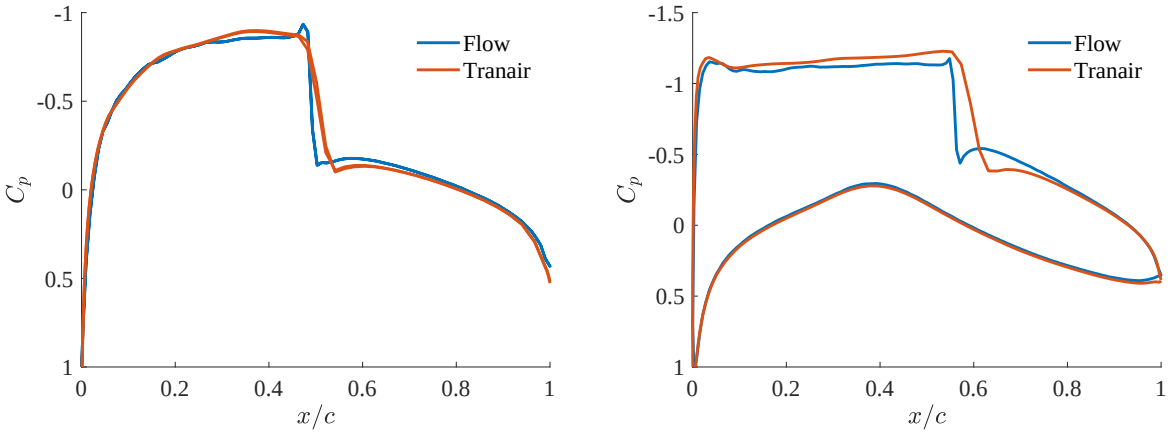
Code	c_l	c_d
`Flow`	0.000	0.0049
`Tranair`	-0.011	0.0059

(a) NACA 0012: $\alpha = 0°$, $M = 0.80$.

Code	c_l	c_d
`Flow`	0.818	0.0025
`Tranair`	0.847	0.0024

(b) RAE 2822: $\alpha = 2°$, $M = 0.715$.

Table 5.3.3: Aerodynamic load coefficients of two-dimensional airfoils in transonic flow obtained using `Flow` and compared to `Tranair`'s predictions.



(a) NACA 0012: $\alpha = 0°$, $M = 0.80$.

(b) RAE 2822: $\alpha = 2°$, $M = 0.715$.

Figure 5.3.2: Pressure distributions along the chord of two-dimensional airfoils in transonic flow obtained using `Flow` and compared to `Tranair`'s predictions.

Onera M6 and Embraer benchmark wing

The aerodynamic load coefficients of two three-dimensional wings are given in Table 5.3.4. The first is the Onera M6 wing, simulated at an angle of attack $\alpha = 3.06°$ and a Mach number $M = 0.839$. The second is the Embraer benchmark wing, for which the angle of attack is set to yield a lift coefficient $C_L = 0.60$ and the Mach is set to $M = 0.78$. In both cases, the moment coefficient is computed at the leading edge of the root chord. Overall, there is good agreement between the results obtained using `Flow` and `Tranair`.

Code	C_L	C_D	C_M
`Flow`	0.294	0.0111	-0.218
`Tranair`	0.288	0.0111	-0.212

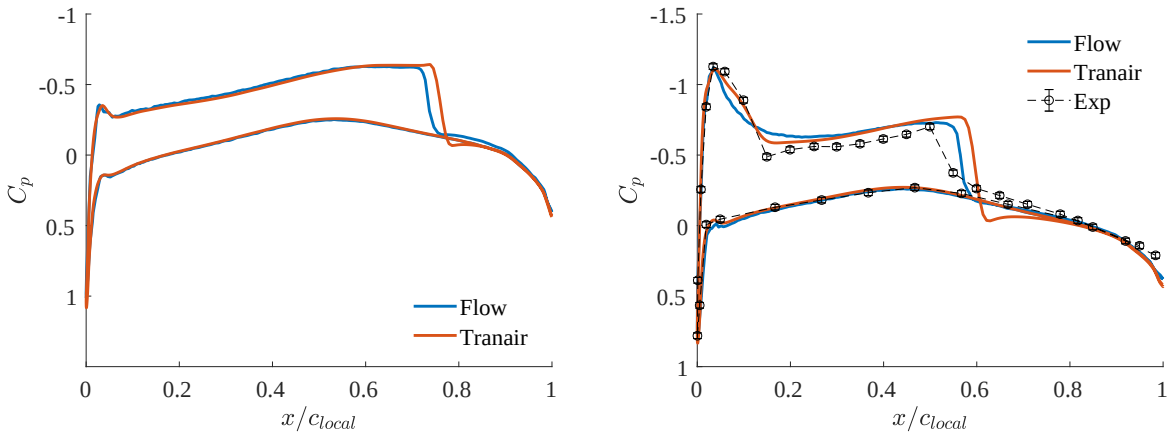
(a) Onera M6: $\alpha = 3.06°$, $M = 0.839$.

Code	α	C_D	C_M
`Flow`	$-0.8°$	0.0147	-0.853
`Tranair`	$-0.9°$	0.0159	-0.857

(b) Embraer benchmark wing: $C_L = 0.60$, $M = 0.78$.

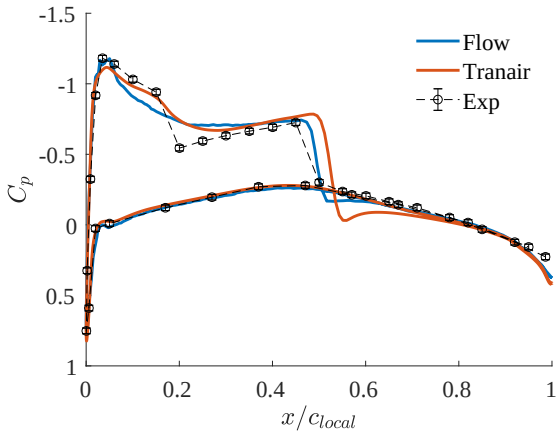
Table 5.3.4: Aerodynamic load coefficients of three-dimensional wings in transonic flow obtained using `Flow` and compared to `Tranair`'s predictions.

Figure 5.3.3 shows the pressure coefficient along the chord at various spanwise stations of the Onera M6. These stations are taken at the root, 44%, 65% and 95% of the semi-span of the wing. Note that the mean aerodynamic chord is located at $y/s = 0.44$, where y is the spanwise coordinate and s is the semi-span. The experimental data obtained by Schmitt and Charpin [54] are also included, except at the wing root, where they are not available. At the root and the mean aerodynamic chord stations, the solution obtained with `Flow` closely matches the predictions from `Tranair`, even though the shock predicted by `Flow` is located slightly upstream, as in the two-dimensional cases discussed earlier. At $y/s = 0.65$, the double shock pattern observed by Schmitt and Charpin [54] is not at all captured by `Flow`, but is almost captured by `Tranair`, even if it is heavily smeared. However, `Tranair` has the advantage of using an adaptive grid procedure, that allows the software to refine the grid to better capture flow gradients. At the tip section, although both solvers predict a similar solution, neither `Flow` nor `Tranair` recovers a shock as sharp as that observed experimentally. Note that the shock predicted by full potential solutions is expected to be stronger than that obtained experimentally, as observed in chapter 2.

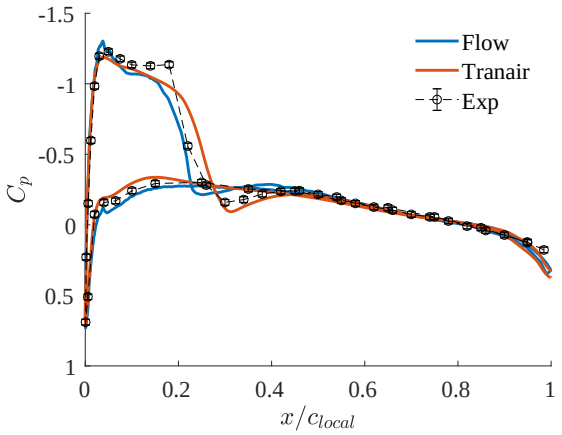


(a) $y/s = 0.$

(b) $y/s = 0.44$



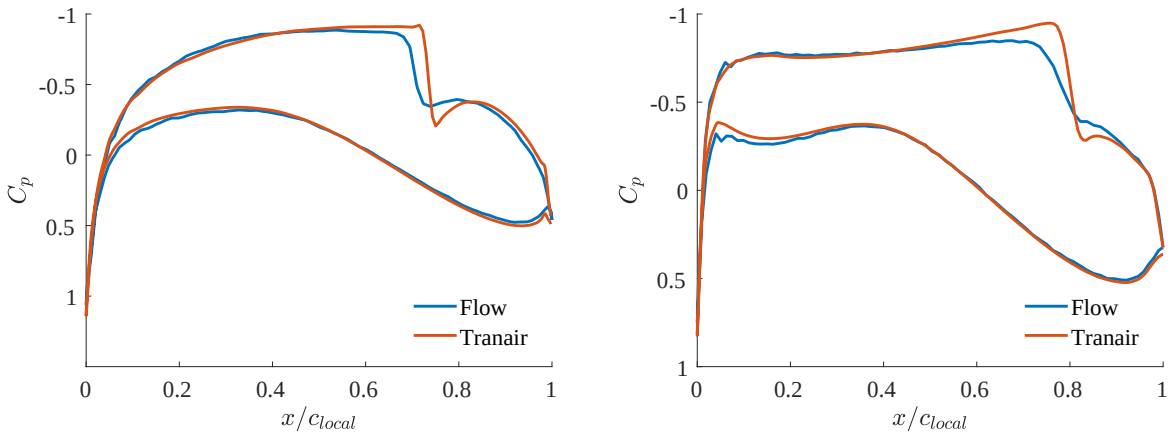
(c) $y/s = 0.65$



(d) $y/s = 0.95$

Figure 5.3.3: Pressure distributions along the chord at several spanwise stations of the Onera M6 at $\alpha = 3.06°$ and $M = 0.839$ obtained using `Flow` and comapred to `Tranair`'s predictions and experimental data [54].

Figure 5.3.4 shows the pressure coefficient along the chord at various spanwise stations of the Embraer benchmark wing. These stations are taken at the root, 37%, 41% and 95% of the semi-span of the wing. Note that the mean aerodynamic chord is located at $y/s = 0.41$. As in the Onera M6 case, there is an overall good agreement between the `Flow` and `Tranair` solutions, except at the shock location, which tends to be smeared and moved upstream by `Flow`. This can also be related to the lower drag coefficient predicted by `Flow`.



(a) $y/s = 0.$ (b) $y/s = 0.37$

(c) $y/s = 0.41$ (d) $y/s = 0.95$

Figure 5.3.4: Pressure distributions along the chord at several spanwise stations of the Embraer benchmark at $C_L = 0.60$ and $M = 0.78$ obtained using `Flow` and compared to `Tranair`'s predictions.

5.3.3 Computational performance

The test cases presented in this chapter were run on a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz). The mesh size and the computational time required by `Flow` and `Tranair` are given in Tables 5.3.5 and 5.3.6 for two and three-dimensional cases, respectively.

For two-dimensional flow computations, `Flow` is about three to five times faster than `Tranair`. This is due to the solution adaptive grid procedure implemented in `Tranair`, which needs to generate adapted meshes several time during the iterative process. For such small mesh sizes, the grid generation seems to take longer than the actual solution process. On the other hand, only one fine grid is used for `Flow`. The finite element matrix computation and assembly processes in `Flow` are multi-threaded using shared memory parallelization. When the test cases are run on four threads, the wall-clock time is reduced by a factor of two. The ideal factor of four cannot be reached as, first, the linear solver is not used in parallel, and second, little additional time is required for communication and synchronization.

Code	n. cells	n. threads	wall-clock time	cpu time
`Flow`	4 000	1	0.7 s	0.7 s
`Flow`	4 000	4	0.3 s	1.2 s
`Tranair`	6 500	1	6 s	6 s

(a) NACA 0012: $\alpha = 0°$, $M = 0.80$.

Code	n. cells	n. threads	wall-clock time	cpu time
`Flow`	6 000	1	1.8 s	1.8 s
`Flow`	6 000	4	1.0 s	4.0 s
`Tranair`	6 000	1	5 s	5 s

(b) RAE 2822: $\alpha = 2°$, $M = 0.715$.

Table 5.3.5: Computational performance of two-dimensional transonic flow computations preformed using `Flow` and compared to `Tranair`.

For three-dimensional flows, `Flow` requires approximately the same computational time as `Tranair` to compute the flow around the Onera M6 wing, and is around 10% faster in the Embraer benchmark case. In both cases, the mesh used by flow is larger, but as `Tranair` needs to generate several grids during the grid adaptation procedure, the total runtime remains comparable. Similarly to the two-dimensional cases, a reduction of 50% in wall-clock time is obtained by running `Flow` on four threads.

Code	n. cells	n. threads	wall-clock time	cpu time
`Flow`	600 000	1	245 s	245 s
`Flow`	600 000	4	100 s	400 s
`Tranair`	500 000	1	240 s	240 s

(a) Onera M6: $\alpha = 3.06°$, $M = 0.839$.

Code	n. cells	n. threads	wall-clock time	cpu time
`Flow`	700 000	1	290 s	290 s
`Flow`	700 000	4	120 s	480 s
`Tranair`	500 000	1	320 s	320 s

(b) Embraer benchmark wing: $C_L = 0.60$, $M = 0.78$.

Table 5.3.6: Computational performance of three-dimensional transonic flow computations preformed using `Flow` and compared to `Tranair`.

5.4 Sensitivity analysis

As `Flow` is intended to be used in the preliminary aircraft design stage, the code has been designed such that a computation can be easily configured. Consequently, the number of parameters chosen arbitrarily by the user is kept to a minimum. However, two parameters of paramount importance still need to be fixed: the wake inclination and the mesh density. The present section illustrates the effect of these two parameters on the accuracy of transonic flow computations performed within `Flow`.

5.4.1 Wake inclination

As explained earlier, the wake must be aligned with the trailing edge bisector for three-dimensional configurations. However, the bisector line is not well defined for highly curved trailing edges. The present study attempts to quantify the effect of the wake inclination on three-dimensional flow solutions.

The Embraer wing is chosen as a benchmark and is simulated at an angle of attack $\alpha = -1.2$ and a Mach number $M = 0.78$. This flight condition corresponds approximately to the high-speed maneuver described in chapter 2. Three wake inclinations are chosen: the first is based on a trailing edge defined as 1% of the chord, the second uses 15% of the chord, and the third wake is horizontal. The strategy to define the wake inclination is depicted in Figure 5.4.1. The results, given as integrated aerodynamic coefficients and pressure coefficients along the chord are compared to the predictions obtained from `Tranair`, and summarized in Table 5.4.1. Note that `Tranair` uses a horizontal wake. Compared to `Tranair`, the closest lift and moment coefficients are obtained using the second configuration, for which the wake is tilted following a bisector based on a trailing edge defined as 15% of the chord. The corresponding drag coefficient differs by 17 counts from `Tranair`, which is quite large compared to the first configuration, which only differs by 3 counts. This result is in line with those presented in Table 5.3.4b which show that, in the Embraer benchmark wing case, `Flow` underestimates the drag but yields lift and moment coefficients, and an angle of attack similar to `Tranair`.

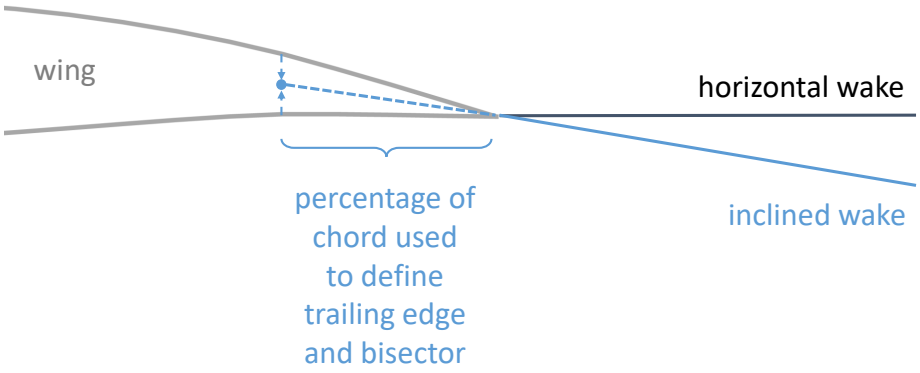
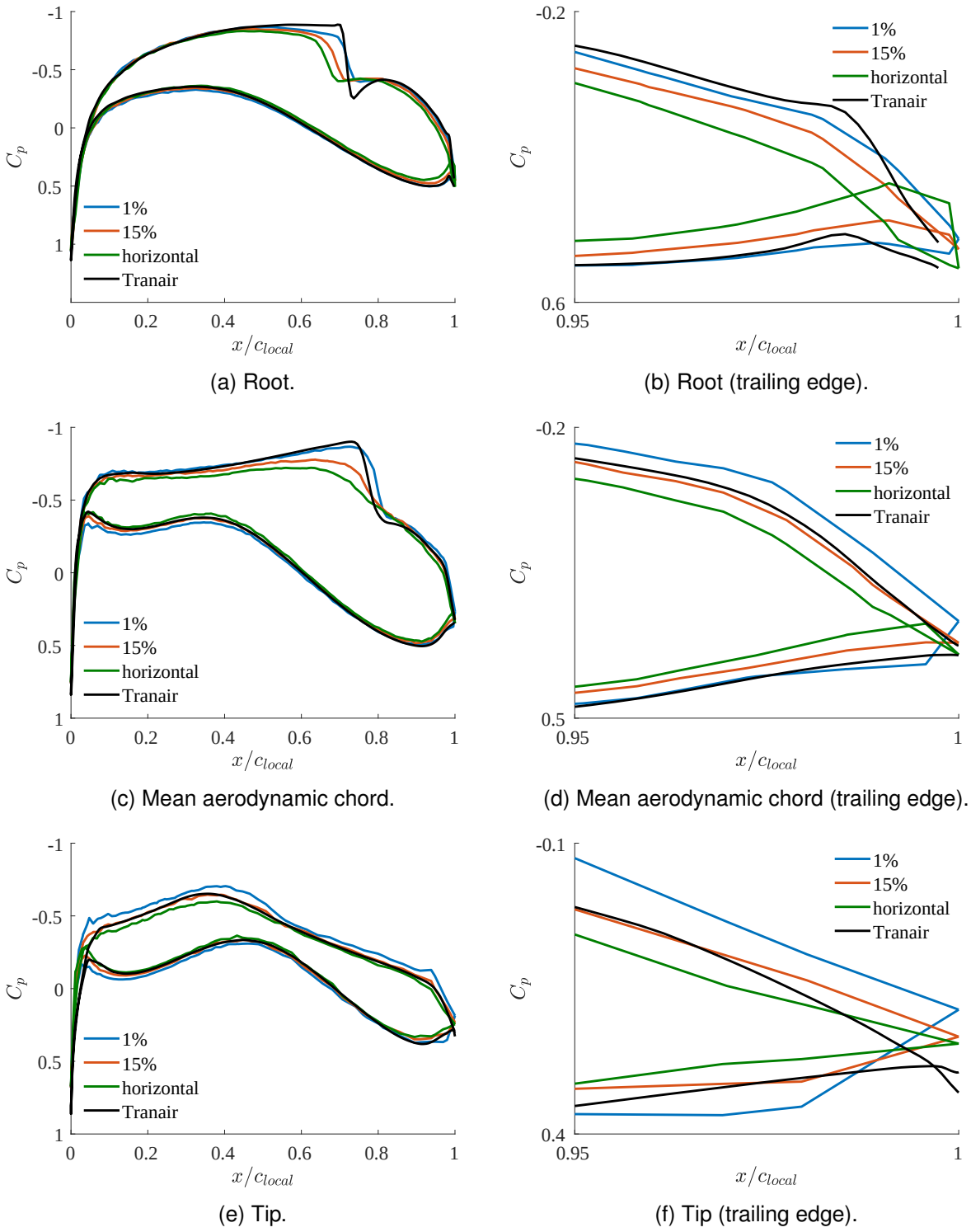


Figure 5.4.1: Strategy to define the trailing edge bisector and the wake inclination.

Code	Inclination	C_L	C_D	C_M
`Flow`	1%	0.60	0.0142	-0.877
`Flow`	15%	0.54	0.0122	-0.775
`Flow`	horizontal	0.48	0.0104	-0.685
`Tranair`	horizontal	0.56	0.0139	-0.806

Table 5.4.1: Evolution of the aerodynamic coefficients of the Embraer wing at $\alpha = -1.2°$ and $M = 0.78$ for different wake inclination compared to `Tranair`'s results.

Figure 5.4.2 shows the pressure coefficient along the root section, mean aerodynamic and tip section chords of the Embraer benchmark wing simulated at $\alpha = -1.2°$ and $M = 0.78$ for three wake inclinations. A zoom on the last 5% of the local chord is also provided to judge the accuracy of the Kutta condition. The results obtained using `Tranair` are also given for reference. The configuration yielding the most similar pressure distribution compared to `Tranair` varies depending on the considered section. At the root and the mean aerodynamic chord sections, the first configuration, based on a trailing edge defined as 1% of the chord, best matches `Tranair`'s results, while at the tip section, the second configuration is the closest one. The local behavior of the solution at the trailing edge confirms this result. Moreover, it indicates that the Kutta condition implemented in the two solvers does not ensure the exact equality of the pressure on the suction and pressure sides. Furthermore, the Kutta condition is not fulfilled on each section for both `Flow` and `Tranair`. For a given wake configuration, the difference between the pressure on the suction and pressure sides at the trailing edge obtained using `Flow` decreases along the span, while the opposite trend can be observed for `Tranair`.



(a) Root.

(b) Root (trailing edge).

(c) Mean aerodynamic chord.

(d) Mean aerodynamic chord (trailing edge).

(e) Tip.

(f) Tip (trailing edge).

Figure 5.4.2: Pressure distribution along the chord of several sections along the span of the Embraer wing at $\alpha = -1.2°$ and $M = 0.78$ for different wake inclination compared to `Tranair`'s predictions.

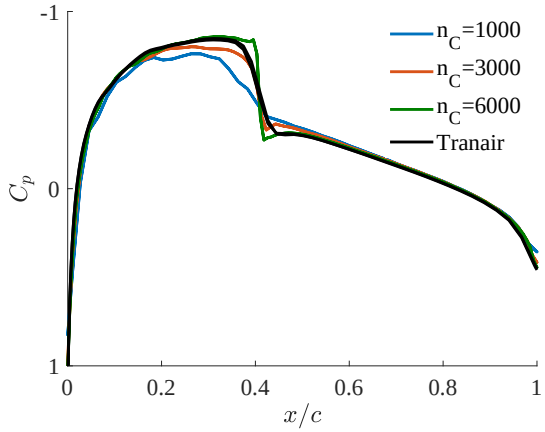
Both `Tranair` and `Flow` solutions depend on the Kutta condition. Even if the implementation in `Tranair` does not enforce the exact equality of the pressure on the suction and pressure sides along the whole span, the wake is always horizontal and its inclination is not controlled

by the user. This ensures that the results will be consistent from one lifting configuration to another. This is not the case with `Flow`, since the wake inclination must be provided by the user. Furthermore, a suitable wake inclination is not trivial to determine and will be different for each wing. Several tentative solutions will be discussed in chapter 7 in order to decrease the influence of the wake inclination on the solution.

5.4.2 Grid density

The two-dimensional flow over a NACA 0012 airfoil is computed using `Flow` at zero angle of attack and several freestream Mach numbers to illustrate the impact of grid refinement on shock capturing. The pressure along the chord of the airfoil, as well as the drag coefficients are compared to `Tranair`, and given in Figures 5.4.3 to 5.4.5. It should be recalled that `Tranair` makes use of a robust solution adaptive grid feature, which ensures that the local grid size always fits the solution.

For a freestream Mach number $M = 0.78$, the flow exhibits a very weak shock, with an associated maximum local Mach number of 1.1. For this case, Figure 5.4.3 shows that a grid counting less than $n_C = 6\,000$ cells is not dense enough to accurately capture the shock and yield an accurate drag coefficient. For a grid of $6\,000$ cells, the captured shock is slightly sharper, and the drag coefficient differs by less than one drag count compared to `Tranair`.



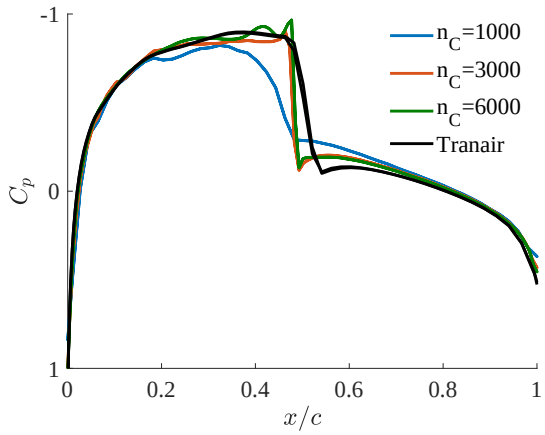
(a) Pressure coefficient.

n_C	C_D
$1\,000$	0.00161
$3\,000$	0.00135
$6\,000$	0.00172
`Tranair`	0.00171

(b) Drag coefficient.

Figure 5.4.3: Pressure along the chord and drag coefficient of the NACA 0012 at $\alpha = 0°$ and $M = 0.78$ for several mesh sizes compared to `Tranair`'s results.

For a freestream Mach number $M = 0.80$, the shock is stronger and the maximum local Mach number is close to 1.3, which is the theoretical limit of validity of the full potential equation. For this case, Figure 5.4.4 illustrates that a grid counting $n_C = 3\,000$ elements is sufficient to capture a sharp shock, even though it is displaced slightly upstream compared to `Tranair`'s prediction. The difference between the drag coefficients predicted by the two solvers increases to $4 - 5$ drag counts. Further refining the grid leads to an oscillatory behavior, which prevents convergence.



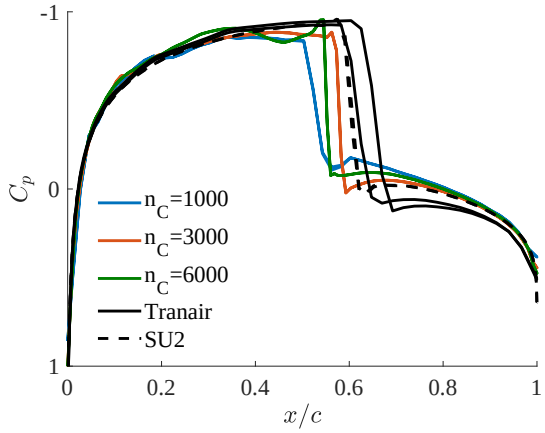
(a) Pressure coefficient.

n_C	C_D
1 000	0.00331
3 000	0.00457
6 000	0.00488
`Tranair`	0.00413

(b) Drag coefficient.

Figure 5.4.4: Pressure along the chord and drag coefficient of the NACA 0012 at $\alpha = 0°$ and $M = 0.80$ for several mesh sizes compared to `Tranair`'s results.

For a freestream Mach number $M = 0.82$, a strong shock is exhibited and the maximum local Mach number is close to 1.5. For this case, the full potential equation is no longer valid and an Euler solution computed using `SU2` is given as well in Figure 5.4.5. Although `Tranair` still converges, the asymmetry of the pressure between the suction and pressure sides, as well as the prediction of a stronger shock and a lower drag coefficient than `SU2`, illustrate that the limit of the methodology is reached. While using a grid of $n_C = 3\,000$ cells with `Flow` inconsistently yields a weaker shock displaced upstream and a lower drag coefficient than `Tranair` and `SU2`, further refining the grid leads to an oscillatory and unconverged solution. Moreover, as the grid is refined, the shock strength and location do not converge. This is to be compared with the results at $M = 0.78$, for which smooth convergence can be attained, even for fine grids.



(a) Pressure coefficient.

n_C	C_D
1 000	0.00754
3 000	0.01153
6 000	0.01086
`Tranair`	0.01840
`SU2`	0.01944

(b) Drag coefficient.

Figure 5.4.5: Pressure along the chord and drag coefficient of the NACA 0012 at $\alpha = 0°$ and $M = 0.82$ for several mesh sizes compared to `Tranair`'s and `SU2`'s results.

The grid sensitivity study presented in the present section showed that, although transonic

flow computations can reliably be performed using Flow, the solver still lacks robustness when high local Mach numbers are involved. As such, the consistency and stability properties of the upwinding formulation, which are the requirements for convergence, should be further investigated. Additionally, the implementation of techniques limiting the velocity near high flow gradients, or of a solution adaptive grid, is highly desirable. These aspects will be discussed in chapter 7.

5.5 Discussion

Flow, an unstructured grid, finite element method solving the full potential equation was developed and implemented. The code is written in C++ wrapped in Python, and designed to perform transonic aerodynamic and aeroelastic flow computations in the context of preliminary aircraft design. The solver has been tested on various two and three-dimensional configurations, and the results have been compared to Tranair. Overall, there is a good agreement between the predictions of the two solvers. The aerodynamic load coefficients predicted by Flow and Tranair usually differ by less than 10 counts or 5%, whichever the higher. The pressure distributions computed along the mean aerodynamic chord, as well as at several spanwise stations of the wings, also closely matched, though Flow tends to underpredict the velocity upstream of shockwaves. This has the effect of moving the shocks slightly upstream compared to Tranair's predictions. The accuracy of Flow under transonic flow conditions is much higher than that of Aero, the field panel method presented in chapter 4. For two-dimensional cases, Flow is found to be significantly faster than Tranair. For three-dimensional computations, the adaptive grid technology implemented in Tranair increases the solver computational efficiency and yields similar computational time as Flow. It should also be noted that the time required to generate a suitable three-dimensional unstructured grid is non negligible. As such, an adaptive grid technique could also be implemented in Flow. Moreover, such a procedure could also increase the solver's robustness and remove the responsibility of the user to specify a grid size prior to the computation. Development perspectives of adaptive mesh refinement will be presented in chapter 7. Flow is currently restricted to steady inviscid flows. Several enhancements, such as unsteady modeling, viscous corrections, and adjoint computations will be presented in chapter 7 as well.

One of the goals of the present work is to investigate the accuracy and computational efficiency of the different aerodynamic levels of fidelity for transonic aeroelastic computations in the context of preliminary aircraft design. The Python wrapping used in Flow makes the solver modular and easy to integrate into an aeroelasticity framework, as opposed to other software, such as Tranair. Such computations will be presented in the next chapter.

Chapter 6

Static aeroelastic computations

The impact of the different aerodynamic levels of fidelity on static aeroelastic computations is assessed in the present chapter. Previous comparisons performed by various authors are first briefly reviewed. The finite element method developed in chapter 5, `Flow`, is then validated in the context of fluid-structure interaction computations, and the levels of fidelity presented in chapter 1 are compared using the Embraer benchmark wing.

6.1 State-of-the-art

Few comparative studies are available for static aeroelastic computations, even though engineers commonly use multi-fidelity [179] or high-fidelity [180, 181] aerodynamic modeling. The most extensive study is probably the first Aeroelastic Prediction Workshop organized by the American Institute of Aeronautics and Astronautics in 2012 [182, 183]. In this workshop, some authors, such as Romanelli et al. [184] and Acar and Nikbay [185], compared their results obtained with linear potential or Euler equations to experimental data. Particularly, Romanelli et al. observed noticeable discrepancies between wing deflections obtained using the doublet lattice method and the Euler formulation. However, the workshop placed the emphasis on obtaining representative data rather than on comparing the models. Navier-Stokes solvers were mainly used and computational costs were not always reported. An insight into the tradeoff between accuracy and computational time is given by Edwards and Malone [186] for some models in the context of aeroelastic computations, and extensive comparisons are made by Schuster [187] and Henshaw et al. [188]. However, these works focus on unsteady aerodynamics and dynamic aeroelasticity. A systematic study of the effect of the major transonic aerodynamic modeling methods on static aeroelastic predictions was performed by Crovato et al. [40].

6.2 Methodology

In the present chapter, the aerodynamic solvers are coupled to structural solvers using a partitioned approach: the fluid and solid physics are solved using different numerical methods implemented in different solvers, as described in chapter 1.

The Euler equations are solved using `SU2` [41, 42, 43], and the linear potential equations are solved using `Panair` [48, 49] and `NASTRAN` [50, 51]. Furthermore, the Euler solution obtained

with SU2 on the undeformed wing shape, *i.e.* the wing shape in cruise condition, is also used to correct the doublet-lattice solution of NASTRAN. The numerical setup of these solvers is the same as in chapter 2. Finally, the full potential equation is solved using Flow [14, 171]. Reynolds-Averaged Naver-Stokes computations have been performed, but the mesh deformation procedure implemented in SU2 failed to produce a proper new grid, hence causing the solver to diverge. Computations using Tranair were not performed due to the complexity of coupling it to a structural solver. Viscous SU2 and Tranair solutions are therefore not presented in the present chapter.

The equilibrium equations of a solid 1.2.22 are solved using the linear finite element method implemented in NASTRAN [53], while Equation 1.2.25 is solved by modali, an in-house modal solver [189].

Fluid-structure coupling is performed either using NASTRAN, PanFsi, or CUPyDO [190, 191, 192, 193], depending on the software used to calculate the aerodynamic loads. The NASTRAN computations are performed by means of the elastic trim analysis, also known as *SOL 144*, which projects the loads directly on the structural model in physical space. Fluid-structure computations are also carried out using PanFsi, an in-house MATLAB code, such that the displacements are obtained using a MATLAB version of modali, while the loads are computed by Panair. MATLAB interpolation functions are used to transfer data between the fluid and structural meshes. The loads and the displacements are updated until the difference in the loads between two consecutive iterations falls below a prescribed tolerance. Finally, Flow and SU2 are coupled to modali through CUPyDO, a python suite designed to couple staggered solvers. The code is developed at the University of Liège and offers different coupling algorithms and different methods to interpolate the loads on the structural mesh and the displacements on the fluid nodes. In the present work, the simulations are performed using the Block-Gauss-Seidel algorithm described in chapter 1 and the variables are interpolated using radial basis functions. Convergence is reached when the difference in the magnitude of the displacements between two consecutive iterations drops below a prescribed tolerance. The normalized tolerance[1] is set to 10^{-4} for both PanFsi and CUPyDO.

The combinations of aerodynamic and structural solvers, and fluid-structure couplers, as well as the naming convention used in the present chapter, are summarized in Table 6.2.1.

Name	Aerodynamic solver	Structural solver	Coupler
PAN	Panair	modali	PanFsi
NAS	NASTRAN	NASTRAN	NASTRAN
NASC	NASTRAN	NASTRAN	NASTRAN
FLO	Flow	modali	CUPyDO
SU2	SU2	modali	CUPyDO

Table 6.2.1: Naming convention used in the present chapter.

[1] The normalized tolerance is obtained by normalizing the criterion (the loads or the displacements) by its maximum expected value.

`Flow`'s capability to perform fluid-structure interaction computations is first assessed on the Agard wing [194]. The full potential solver is compared to various literature results obtained by solving the Euler equations. Computations are then performed using the flexible model of the Embraer benchmark wing [40, 55] in two simulated maneuver conditions in order to assess the impact of aerodynamic modeling on static aeroelastic computations. The wing is considered to be clamped at its root section to represent the attachment to the fuselage, and the angle of attack is adjusted to produce a prescribed static load factor. This boundary condition is not fully realistic, as a real fuselage is not rigid. However, this setup allows to easily compare the different solvers. The gravity load is also neglected in the computations, which is not fully realistic. Possible improvements to these two aspects will be discussed in chapter 7. Both the Agard and the Embraer wings are modeled using orthotropic material properties.

6.3 Agard 445.6

The Agard 445.6 wing is a low aspect ratio, swept and tapered wing, which is depicted in Figure 6.3.1 and whose geometrical and structural parameters are given in Table 6.3.1. The wing was designed for wind tunnel flutter testing, and is widely used as a standard validation case for transonic flutter calculations. Altough experimental data have been collected by Yates [194] for dynamic aeroelastic cases, no measurements are available for static cases, which are the focus of the present work. However, various authors [190, 195, 196] performed static computations to validate their aeroelastic solvers. They simulated the wing at an angle of attack $\alpha = 1^\circ$ and a Mach number $M = 0.80$, yielding a freestream dynamic pressure of $2\,867$ Pa. The same condition is used in the present section.

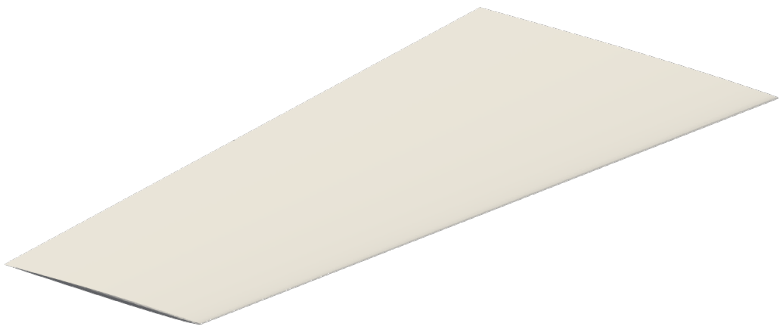


Figure 6.3.1: Agard 445.6 wing model.

Parameter	Value
Aspect ratio	3.3
Taper ratio	0.66
Sweep angle	$45°$
Root chord	559 mm
Semi-span	762 mm

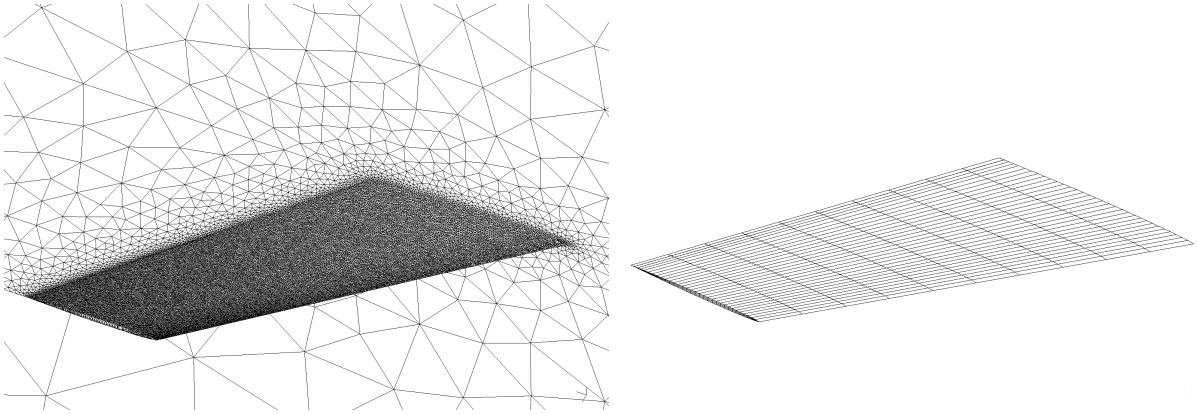
(a) Geometrical properties.

Parameter	Value
Longitudinal Young modulus	3.15 GPa
Transverse Young modulus	0.42 GPa
Shear modulus	0.44 GPa
Poisson's ratio	0.31
Density	382 kg m^{-3}

(b) Structural properties.

Table 6.3.1: Geometrical and structural properties of the Agard 445.6 wing.

The mesh used for FLO is built in the same way as in chapter 5, and the unstructured grid counts $250\,000$ tetrahedra, with a characteristic size of $1/200$ and $1/100$ of the local chord at the leading and trailing edges, respectively. The associated structural model is built in `Metafor` [197] and is based on the weakened model 3 of the wing [190]. The mesh is built using `gmsh` [57] and consists of 31, 2 and 17 hexahedral cells in the chordwise, normal and spanwise directions respectively. A modal analysis is then performed in `Metafor` to obtain the first four mode shapes which are subsequently used in `modali`. Note that the modes used in the present study are identical to those obtained by Güner et al. [198]. The computational grids used by `Flow` and `Metafor` are illustrated in Figure 6.3.2. The coupling of `Flow` and `modali` is performed using `CUPyDO` and the tolerance on the displacements is set to 10^{-3} mm, which correspond to 10^{-4} times the expected maximum displacement.



(a) Grid used by `Flow`.

(b) Grid used by `Metafor`.

Figure 6.3.2: Fluid and structural computational grids used for the Agard 445.6 wing.

6.3.1 Aerodynamic loads

Table 6.3.2 gives the aerodynamic coefficients of the deformed wingshape. The results are also compared to the Euler solution obtained by Thomas [190] using `SU2` and `modali` coupled through `CUPyDO`. There is good overall agreement between `Flow` and `SU2` for all aerodynamic coefficients, which differ by one count or less. The relative difference in the drag coefficients predicted by the solvers is about 20%, but this remains acceptable since the magnitude of the coefficient is quite small.

Model	C_L	C_D	C_M
FLO	0.053	0.00044	-0.055
Euler (Thomas [190])	0.054	0.00035	-0.056

Table 6.3.2: Aerodynamic coefficients for the flexible Agard wing obtained using `Flow` at $\alpha = 1^\circ$ and $M = 0.80$, and compared to those obtained by Thomas [190].

6.3.2 Wing deflection

Table 6.3.3 gives the vertical deflection Δz at the wingtip averaged between the leading and trailing edges, as well as the pitch rotation angle $\Delta\varepsilon$ of the wingtip of the Agard wing. $\Delta\varepsilon$ is computed about the quarter-chord of the wingtip section and is positive nose up. The results are also compared to those obtained by Thomas [190], Melville et al. [195] and Goura [196] using various Euler solvers. Again, there is good overall agreement between the different solvers and results previously reported in the literature. `Flow` tends to predict slightly smaller displacements, but a similar wingtip's rotation, than Euler solvers.

Model	Δz (mm)	$\Delta\varepsilon$ ($^\circ$)
FLO	11.3	-0.2
Euler (Thomas [190])	12.4	-0.2
Euler (Melville [195])	11.7	-0.2
Euler (Goura [196])	12.0	-0.2

Table 6.3.3: Mean vertical displacement and pitch-up rotation of the wingtip of the flexible Agard wing obtained using `Flow` at $\alpha = 1^\circ$ and $M = 0.80$, and compared to Thomas [190], Melville's [195] and Goura's [196] results.

6.4 Embraer benchmark wing

The Embraer wing, described in chapter 2, is considered to be flexible and is used as a benchmark for static aeroelastic computations. The wing skin and the wingbox structure are illustrated in Figure 6.4.1. Note that the engine's nacelle is also represented but not taken into account in the computations.

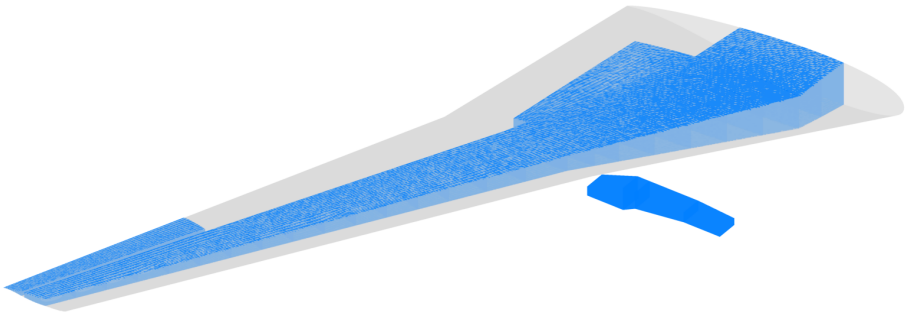
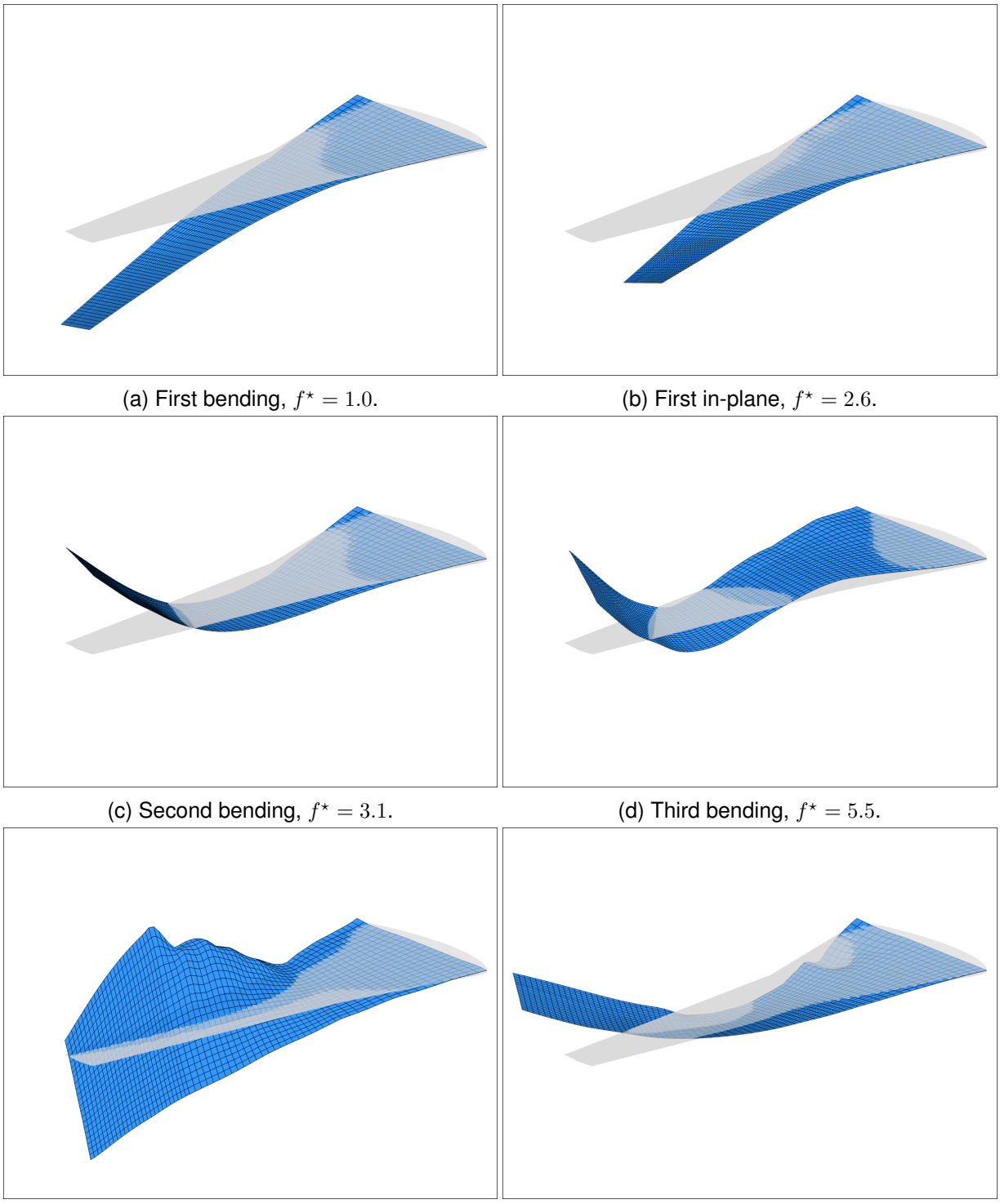


Figure 6.4.1: Wingbox structure of Embraer benchmark wing.

The aerodynamic meshes used for models PAN, NAS, NASC and SU2 are described in chapter 2, while the mesh used for FLO is described in chapter 5. The structural model is discretized in NASTRAN using $50,000$ shell elements, and the fluid-structure computations performed using NASTRAN are carried out directly using this structural model. The computations performed through MATLAB and CUPyDO are carried out using the first six mode shapes obtained by a modal analysis performed in NASTRAN. The modes and their associated frequencies, normalized by a reference value, are depicted in Figure 6.4.2. Since the geometry of the wingbox does not match with that of the wing skin, the modes are first interpolated on the mean chord plane of the wing using the infinite plane spline methodology, and then projected from this mean plane to the wing skin using a closest point strategy. The methodology is described in detail by Güner [199]. The grid used by `modali` to represent the mode shapes consists of $2\,100$ points on the wing surface. The fluid-structure computations are performed at low-altitude/low-speed and high-altitude/high-speed maneuver conditions. For the low-speed maneuver, the Mach number is $M = 0.50$, the lift coefficient is $C_L = 0.80$ and the altitude is $6\,500$ ft, resulting in a load factor $n = 2.5$. For the high-speed maneuver, $M = 0.78$, $C_L = 0.60$, the altitude is $27\,000$ ft, and $n = 2$.



(a) First bending, $f^\star = 1.0$.

(b) First in-plane, $f^\star = 2.6$.

(c) Second bending, $f^\star = 3.1$.

(d) Third bending, $f^\star = 5.5$.

(e) First torsion, $f^\star = 6.9$.

(f) Second in-plane, $f^\star = 7.4$.

Figure 6.4.2: First six mode shapes and normalized frequencies of the Embraer benchmark wing.

6.4.1 Aerodynamic loads

Table 6.4.1 summarizes the predicted angle of attack and aerodynamic coefficients of the benchmark wing in its deformed configuration for low and high-speed maneuvers. At low-speed, the linear model PAN yields similar results compared to the nonlinear model SU2. However, it

slightly overpredicts the angle of attack needed to achieve the target lift coefficient and underpredicts the drag coefficient at high-speed. On the other hand, the lattice model NAS neglects the camber of the wing, and strongly overperdicts the angle of attack and underpredicts the moment coefficient, for both flight conditions. Using the Euler correction with NASC significantly improves the predictions, even though the angle of attack is still overestimated by about one degree. Finally, FLO is found to underestimate the drag compared to SU2, at both low and high speeds. Overall, the results are similar to those presented in chapter 2. Comparison of the results of Table 6.4.1 to those of Table 2.4.2 illustrates the impact of wing deformation on the angle of attack and aerodynamic coefficients: the angle of attack needed to reach a prescribed lift increases, as well as the drag and the magnitude of the moment.

Model	α	C_L	C_D	C_M
PAN	$+3.7$	0.80	0.0256	-1.073
NAS	$+9.8$	0.80	-	-0.937
NASC	$+5.1$	0.80	-	-0.988
FLO	$+3.8$	0.80	0.0217	-1.049
SU2	$+3.7$	0.80	0.0257	-1.060

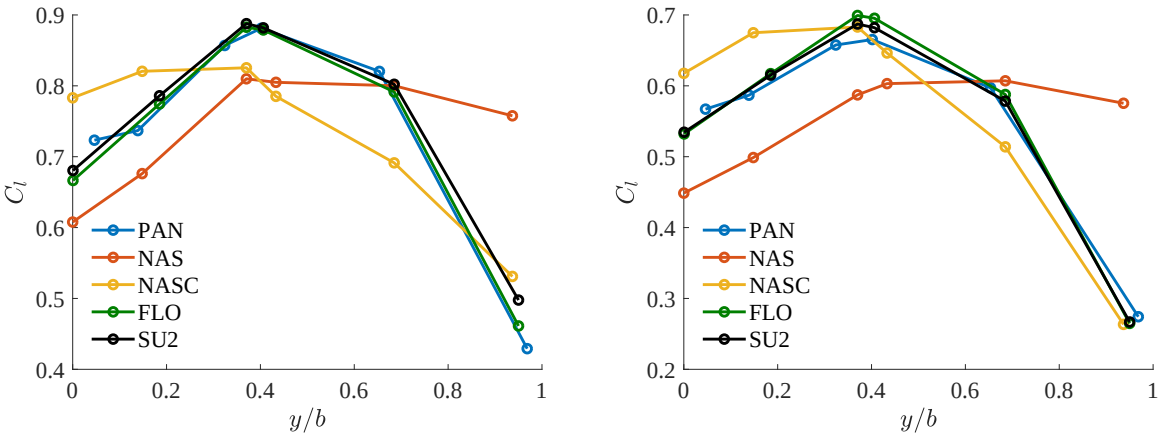
(a) $M = 0.50$ and $C_L = 0.80$.

Model	α	C_L	C_D	C_M
PAN	$+0.5$	0.60	0.0148	-0.885
NAS	$+6.1$	0.60	-	-0.705
NASC	$+0.9$	0.60	-	-0.812
FLO	$+0.1$	0.60	0.0141	-0.867
SU2	-0.1	0.60	0.0186	-0.890

(b) $M = 0.78$ and $C_L = 0.60$.

Table 6.4.1: Angles of attack and aerodynamic coefficients for the flexible Embraer wing obtained from different levels of fidelity for low and high-speed maneuvers.

Figure 6.4.3 shows the lift coefficient distribution along the span of the deformed wing for low and high-speed maneuvers. PAN and FLO predict similar distributions than SU2 at both low and high speeds, contrary to NAS, which is completely inaccurate as it ignores the camber of the wing. Using the correction method implemented in NASTRAN allows to improve the lift distribution, even though NASC results still noticeably differ from the other models, especially at low-speed.

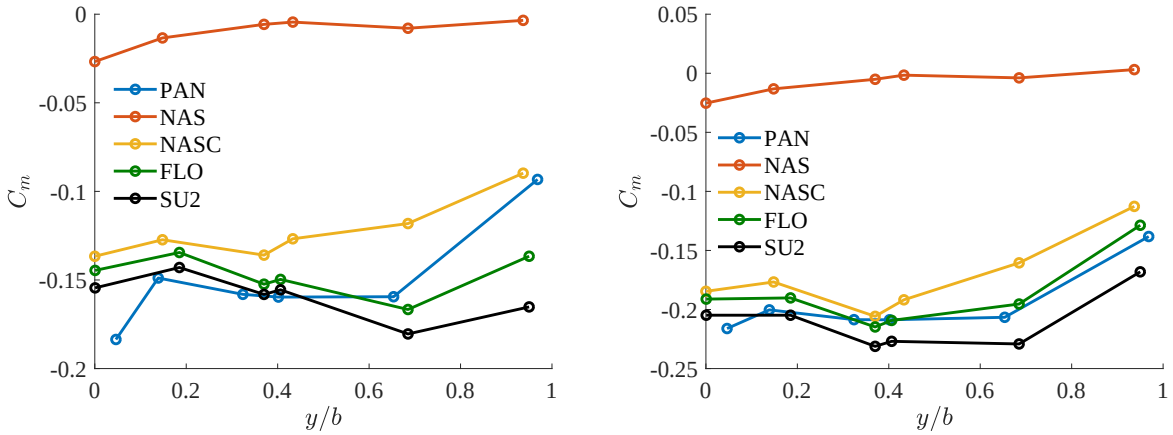


(a) $M = 0.50$ and $C_L = 0.80$.

(b) $M = 0.78$ and $C_L = 0.60$.

Figure 6.4.3: Sectional lift coefficient distribution along the span of the deformed Embraer wing in low and high-speed maneuver conditions, obtained from different levels of fidelity.

Figure 6.4.4 shows the quarter-chord moment coefficient distribution along the span of the deformed wing for low and high-speed maneuvers. The nonlinear models FLO and SU2 yield similar results at both low and high speeds. At high-speed, PAN predicts a moment distribution similar to the nonlinear models. On the other hand, at low-speed, it underestimates the magnitude of the moment at the root and overestimates it at the tip. Again, NAS is completely inaccurate as it ignores the camber of the wing. Using the Euler pressure correction improves the solution, even though NASC predictions still differ from those of the nonlinear models, particularly at low speed.



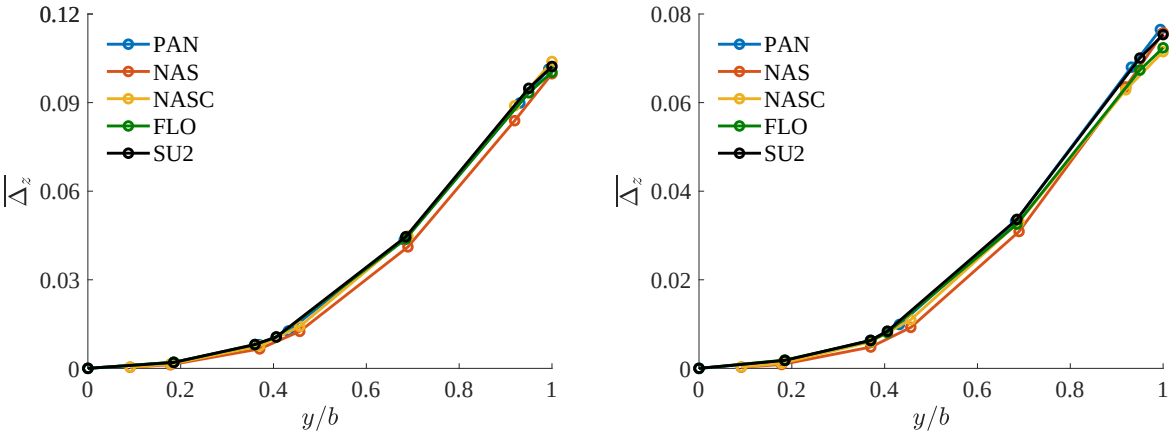
(a) $M = 0.50$ and $C_L = 0.80$.

(b) $M = 0.78$ and $C_L = 0.60$.

Figure 6.4.4: Sectional moment coefficient distribution along the span of the deformed Embraer wing in low and high-speed maneuver conditions, obtained from different levels of fidelity.

6.4.2 Wing deflection

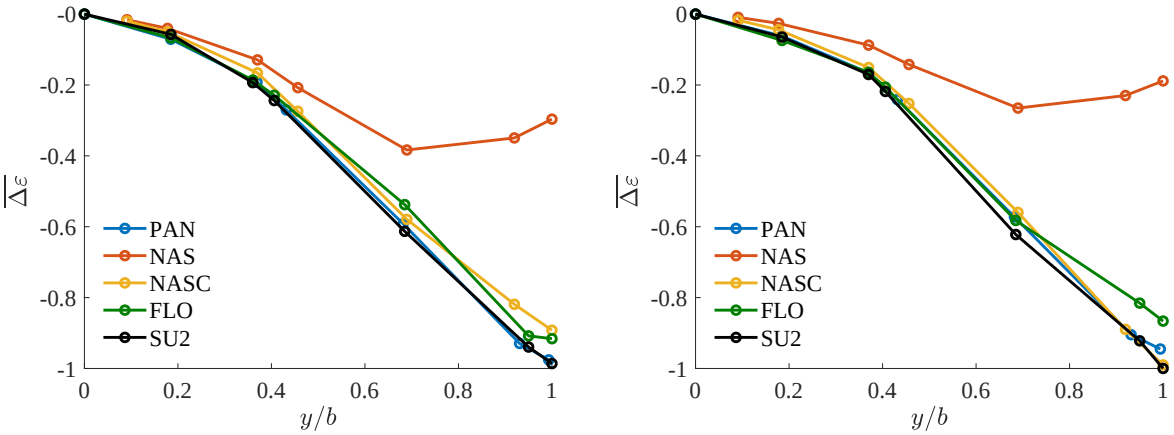
Figure 6.4.5 shows the vertical displacement, averaged between the leading and trailing edges, of the deformed wing for low and high-speed maneuvers. Note that the displacement is normalized with respect to the half-span of the wing. All models predict a similar displacement curve for both maneuver conditions. This is also true for NAS, even though its lift distribution differs from the other models.



(a) $M = 0.50$ and $C_L = 0.80$. (b) $M = 0.78$ and $C_L = 0.60$.

Figure 6.4.5: Mean vertical displacement along the span of the deformed Embraer wing in low and high-speed maneuver conditions, obtained from different levels of fidelity.

Figure 6.4.6 shows the nose-up rotation in pitch along the span of the deformed wing at low and high speeds. The rotation angle is computed along the quarter-chord and is normalized by the maximum value of the rotation obtained from the SU2 solution. All models predict a similar rotation, except for NAS, which strongly underestimates the rotation at the outboard section of the wing. For the others models, the differences in the moment distributions observed in Figure 6.4.4 have little impact on the deformation of the wing.



(a) $M = 0.50$ and $C_L = 0.80$. (b) $M = 0.78$ and $C_L = 0.60$.

Figure 6.4.6: Nose-up rotation along the span of the deformed Embraer wing in low and high-speed maneuver conditions, obtained from different levels of fidelity.

6.4.3 Computational performance

The mesh size and the time required to run the computations presented in the present chapter are given in Table 6.4.2. The calculations were performed in serial on a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz). In this case, the computational cost of the Euler correction required by NASC has been included. PAN converged in 8 and 9 fluid-structure iterations at

low and high speed, respectively. FLO and SU2 converged respectively in 7 and 9 iterations for both low and high speeds. Finally, NAS and NASC converged in 3 iterations at both speeds. PAN and NAS are about fifteen times faster than FLO, which is itself around fifty times faster than SU2.

Model	n. cells	n. threads	wall-clock time
PAN	$1\,400$	1	80 s
NAS	800	1	75 s
NASC	800	1	5 h + 75 s
FLO	$700\,000$	1	20 min
SU2	$1\,300\,000$	1	13 h

(a) $M = 0.50$ and $C_L = 0.80$.

Model	n. cells	n. threads	wall-clock time
PAN	$1\,400$	1	90 s
NAS	800	1	75 s
NASC	800	1	6 h + 75 s
FLO	$700\,000$	1	25 min
SU2	$1\,300\,000$	1	12 h

(b) $M = 0.78$ and $C_L = 0.60$.

Table 6.4.2: Mesh size and computational time required by the different models for the Embraer benchmark case.

6.5 Discussion

The static fluid-structure interaction computations first performed on the Agard wing demonstrated that `Flow`, coupled with `modali` through `CUPyDO`, yielded results comparable to various Euler solutions gathered from the literature, hence validating the aeroelastic capability implemented in `Flow`.

Static aeroelastic computations were then performed on the flexible Embraer benchmark wing to compare the different levels of fidelity. Overall, the results predicted by the linear and nonlinear models are quite similar, at both low and high speeds. However, the camber must be modeled, otherwise wildly inaccurate results will be obtained. Representing the geometry by a flat lattice and correcting the results with a solution obtained using a higher fidelity calculation on the undeformed wing shape is also more or less accurate, but the computational cost of the method then becomes similar to the cost of the higher fidelity computation. For the present case at least, static wing deflections are not sensitive to shock modeling. The aerodynamic coefficients and the angle of attack obtained using the different solvers are also similar, except for the drag coefficient at high-speed. Consequently, flight shape calculations can be carried out using linear methods with reasonable confidence in the predictions. Once the deformed wing shape is known, a single nonlinear rigid aerodynamic calculation can be performed on this shape in order to estimate the aerodynamic loads more accurately.

The analyses performed on the Embraer wing also showed the effect of taking into account the

static deformation on the aerodynamic coefficients. Overall, a higher angle of attack is needed to reach the same lift coefficient, and the drag coefficient is slightly higher, as is the magnitude of the moment coefficient.

Chapter 7

Conclusion

7.1 Summary and conclusions

Modern aircraft wings usually have high aspect ratios and are made of composite materials. Since such wings are light and flexible, hence prone to large deformations, their aeroelastic behavior must be considered in the early stages of the aircraft design process, where low-fidelity linear aerodynamic modeling is traditionally favored because of its low computational cost. However, modern aircraft fly in the transonic regime, in which the flow is nonlinear. The main goals of the present work were to assess the impact of aerodynamic modeling methodology on transonic aerodynamic and aeroelastic computations performed on both rigid and flexible wings during preliminary aircraft design, and to develop an aerodynamic modeling tool able to perform such computations efficiently.

The comparison of the different levels of fidelity commonly used for transonic aerodynamic modeling in preliminary aircraft design showed that the model should take the viscosity of the fluid into account, or at least include a boundary layer modeling technique, as the boundary layer has a non-negligible effect on the pressure distribution. Using an inviscid model will result in an overestimation of the aerodynamic loads, even though they remain representative in the context of preliminary aircraft design. Moreover, in the general case, a nonlinear model is mandatory to predict transonic flows, as shocks might be present. Neglecting shock waves could result in choosing a wing shape not suited for the prescribed flight condition. Full potential based models are appropriate to carry out such calculations, as they offer a good trade-off between accuracy and computational time, and usually achieve an accuracy similar to that of higher fidelity models for a fraction of their cost. For wing shapes already optimized for the transonic regime, the shock, if any is present, will be weak and a linear model can be used, as it will yield aerodynamic loads similar to those predicted by a nonlinear model, except for the drag. Since aeroelastic computations are usually performed using an optimized wing shape and that wing displacements mainly consist of out-of-plane motion driven by the lift and moment distributions, using a linear model also allows to recover static deflections similar to those predicted by nonlinear models. It should however be stressed that the geometry used with the linear model must include the camber of the wing, otherwise highly inaccurate results will be obtained. Consequently, static aeroelastic computations could be performed efficiently by using a multi-fidelity approach. Several multi-fidelity strategies will be discussed in the next section.

In the context of this work, two different techniques for solving the full potential equation were investigated and implemented in order to perform transonic aerodynamic and static aeroelastic computations: a field panel method and a finite element code. Both codes are open-source and available on GitHub and GitLab, respectively. The field panel code, `Aero` [160], yields reliable results for compressible, but not transonic flows, at a moderate computational cost. However, when the flow is supercritical, the predicted shock is heavily smeared and displaced upstream when compared to other full potential solutions. Moreover, the computational requirements quickly become excessive, both in time and memory. Techniques, such as the fast multipole method, could be implemented to reduce the computational cost, but would not guarantee better results. Although the iterative procedure currently implemented in `Aero` does not yield reliable predictions for transonic computations, the code could be used to correct a linear solution by using field sources obtained through another nonlinear solver. This multi-fidelity technique will be detailed in the next section. The finite element code, `Flow` [171], yields results that closely match those predicted by `Tranair`, a widely used commercial full potential software, for both two and three-dimensional benchmark cases. However, in some cases, `Flow` tends to slightly underestimate the maximum local Mach number upstream of shockwaves, which results in slight shock smearing and drag underestimation. The computational cost of `Flow` is smaller than that of `Tranair` for two-dimensional cases, and similar for three-dimensional flows. If multiple threads are used, the cost can be decreased by at least one half of the ideal speed-up factor. `Flow` is also interfaced to `CUPyDO` [193], an in-house fluid-structure interaction coupling code, and can be used to perform static aeroelastic computations. Although `Flow` yields reliable and consistent results for most transonic cases, it has some drawbacks. Firstly, since the density upwinding formulation depends on the grid size, the accuracy of the shock capturing and the stability of the solution also strongly depend on it. Secondly, the results depend on the wake inclination, which must be aligned with the trailing edge bisector. However, the bisector is not well defined for highly curved geometries, such as supercritical airfoils. Ideas and techniques to alleviate these two issues will be presented in the next section. Thirdly, the vortex at the wingtip trailing edge might cause a local singular behavior in the solution, slowing down the convergence rate of the method. Consequently, the grid should be kept coarse in this area, which might affect the solution accuracy locally.

7.2 Suggestions for future work

Although the objectives of the present thesis have been fulfilled, the research work opened new questions. This section suggests future work to improve the static aeroelastic computations performed in chapter 6 and to add new features to `Flow`.

7.2.1 Improvements of aeroelastic computations

The static aeroelastic computations performed in chapter 6 only gave a first insight into actual aero-structural computations performed during the preliminary aircraft design stage. Several ways to improve these computations will be discussed in the present section.

Inclusion of the gravity load

The aeroelastic computations were performed by only considering the aerodynamic loads acting on the Embraer benchmark wing. Such computations should also include the gravity load to yield realistic and consistent results. The gravity load could be directly included in the structural modal model, or implemented as an additional constant load in `CUPyDO`. The inclusion of the gravity should be further investigated before actual aeroelastic computations can be carried out.

Investigation on the fuselage-wing boundary condition

The boundary condition used at the wing root assumes that the wing is rigidly clamped. In a real situation, the wing is attached to a fuselage that is not rigid. This boundary condition could have an influence on the results and further investigation is necessary. The computations performed in `CUPyDO` could be improved either by modeling the fuselage, or by replacing the boundary condition. On the one hand, the first approach requires to model the internal structure of the fuselage and to obtain its mode shapes. On the other hand, the second approach requires to model the effect of the fuselage on the wing, and to use reaction forces as boundary conditions instead of clamping the wing root. Note that such a boundary condition is not readily available in `CUPyDO` and additional implementation work would be required as well.

Integration in aero-structural optimization

The present work focused only on isolated fluid-structure interaction computations. However, these computations are usually performed during structural or aero-structural optimization processes, such as aeroelastic tailoring. During such computations, the deformation of the wing could become large and strongly affect the behavior of the flow, which could impair the fluid solver's stability. Benchmark studies should therefore be performed to assess the robustness of the solver and of the process. Moreover, in the case of aero-structural computations, aerodynamic sensitivities should be computed to drive the optimization process. While obtaining the sensitivities for a linear model is computational cheap, it might require a high computational time for a nonlinear model. The way aerodynamic sensitivities are computed should therefore also be investigated.

Multi-fidelity aeroelastic computations

The results obtained so far suggest that linear models, provided that they model wing camber, yield accurate static wing deflections at a very low computational cost. However, a nonlinear model is still needed to obtain accurate load distributions, especially for the drag. Consequently, the aeroelastic computation could be performed by using a linear model corrected by a nonlinear model. For a simple case, the nonlinear model could be used only once during the last fluid-structure iteration. For more complex cases, the nonlinear model could be used several times during the computation, as suggested by Jovanov and De Breuker [179]. In both cases, the nonlinear computation should be fast and it should be used infrequently. Another

alternative consists in using a field panel approach, whereby the field source terms would first be generated from an already available flow solution, and then used to correct a linear computation. As a nonlinear computation is first performed on the rigid wing model to optimize its shape for the cruise condition, that computation could be used to compute the field sources. The fluid-structure computation could then be performed with a panel method corrected using these field sources. A similar procedure is already used in aircraft design to scale the aerodynamic influence coefficient matrices in the context of dynamic aeroelastic computations [4]. Using such multi-fidelity computation strategies would allow to minimize the cost while ensuring results accuracy. Note that multi-fidelity modeling is an active research topic.

7.2.2 Development of new features for `Flow`

`Flow`, the finite element solver developed and presented in chapter 5 is currently restricted to steady inviscid flow computations. Future developments, such as adjoint computations for optimization, and extension of the solver to compute unsteady and viscous flows, as well improvements based on sensitivity studies, are presented in this section.

Wake modeling

The current implementation of `Flow` requires to generate a wake embedded in the volume grid. Although this operation is supported by `gmsh`, it is not practical and requires special care during the pre-processing. Using techniques based on the automatic embedment of the wake or on the modification of the numerical formulation on elements intersected by the wake, such as those described by Parrinello et al. [200], would help alleviate this issue.

Additionally, the parametric study performed in chapter 5 showed that the three-dimensional Kutta condition formulation used in `Flow` makes the results sensitive to the wake inclination angle. Removing the dependency or, at least characterizing it, is necessary to obtain more reliable results. The best solution would be to use a flat and horizontal wake. For such a solution to work, additional terms should be assembled on the trailing edge nodes, as in two-dimensional cases. This has already been attempted, but oscillations were obtained in the solution. For geometries with highly curved trailing edges, such as supercritical airfoils, the solution procedure did not even converge. Several attempts have been made to stabilize the computation; the most promising was proposed by Galbraith et al. [13] and consists in normalizing the Kutta contribution by the trailing edge cell area. The technique was found to damp the oscillations for wings made of symmetric airfoils, but proved ineffective for supercritical airfoils. Further research is thus needed. If the wake cannot be flat, it could be iteratively deformed to follow the local flow direction. This method presents two drawbacks. The mesh would have to be regenerated or deformed at each iteration, and the iterative procedure would increase the computational cost. However, note that embedding the wake in the mesh as proposed by Davari et al. [137] or Parrinello et al. [200] would avoid regenerating or deforming the volume mesh. If the dependency of the results on the wake inclination cannot be removed, several three-dimensional benchmark cases should be defined and used to find suitable inclination angles.

More specifically, the benchmark would allow to define which portion of the trailing edge region should be considered when defining the trailing edge bisector.

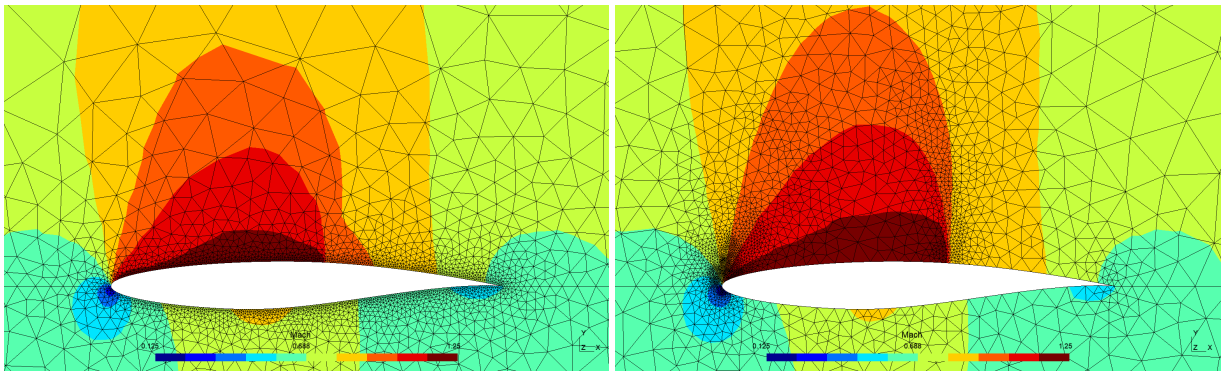
Density upwinding and solution adaptive mesh refinement

The sensitivity study performed in chapter 5 showed that the transonic flow stabilization implemented in `Flow` as well as the shock capturing properties of the solver greatly depends on the grid size. Particularly, the numerical scheme does not converge when the mesh is too fine and that high local Mach numbers are involved. Consequently, the consistency and stability properties of the density upwinding procedure implemented in the code, which are the requirements for convergence, should be further investigated. As the upwinding formulation considered in the present work is similar to that already used by various authors, particular attention should be devoted to the numerical implementation. More specifically, the way the upwind element is chosen should be further investigated.

Additionally, developing a feature such as solution based mesh refinement would allow to remove the difficult task of generating a suitable grid from the user, thus enabling accurate, yet stable computations. Three main aspects must be chosen when implementing a mesh refinement strategy. First, the metric for mesh refinement. Second, the technique used to refine the mesh. Third, the termination criterion. In aerospace applications, both rapid expansions, such as the flow acceleration in the leading edge region, and compression phenomena, such as shockwaves, must be captured accurately. For shock capturing, the metric is usually based on the gradient of the density [130], which is physically consistent. In order to capture expansion phenomena, a metric based on the Laplacian of the density can be used. Recently Schmidmayer et al. [201] proposed several criteria for adaptive mesh refinement for compressible flow computations. The technique used for refining the mesh is usually based on cell splitting. Unstructured triangular and tetrahedral meshes are particularly well suited for this technique, as each cell can be split without creating any hanging node. The termination criterion can be based on the local mesh size or a normalized quantity [130].

`gmsh` features a remeshing capability and the software was used to generate a sequence of solution adapted grids for transonic computations in `Flow`. The procedure can be summarized as follows. Firstly, a transonic flow computation is performed using a coarse grid. Secondly, the cells having a high density gradient are marked for refinement and the cells having a low density gradient are marked for coarsening, while the other cells are left unchanged. Thirdly, an updated cell size is computed for each cell by multiplying the current cell size by a user-defined factor. Finally, the new cell sizes are passed to `gmsh` which generates a new mesh. The process is repeated until the minimum cell size reaches a user defined size. The grid adaption procedure was tested on the RAE 2822 airfoil at an angle of attack $\alpha = 2^\circ$ and a freestream Mach number $M = 0.715$. Figure 7.2.1 illustrates the Mach number around the airfoil as well as the computational grid, and Figure 7.2.2 depicts the pressure coefficient along the chord of the airfoil. The pre-defined grid is obtained by using a uniform cell size of 0.01 of the chord on the airfoil surface and counts $6\,000$ cells and $3\,000$ nodes, while the adapted mesh has a minimal

cell size of 0.005 of the chord near the leading edge and counts $5\,400$ elements and $2\,700$ nodes. Note that the cell size near the shock is similar for both meshes. The final grid required 5 intermediate flow solutions to be generated. Although Figures 7.2.1 and 7.2.2 show that using a grid adapted to the solution yields a smoother solution, and that `Flow`'s predictions move closer to `Tranair`'s, the grid adaptation procedure has two major drawbacks. Firstly, the accuracy of the solution and the robustness of the process both depend on several user-defined parameters, namely: the two thresholds to identify high and low density gradients, the two factors used to increase or decrease the cell size and the termination criterion. Numerical experiments performed on various airfoils at various freestream conditions showed that all the parameters were dependent on both the geometry and the flow conditions, and did not allow to find a set or a range of parameters that produced satisfactory results in all cases. Secondly, `gmsh` generates a completely new grid which might move the shock too much between two successive iterations and lead to robustness issues. Furthermore, the remeshing is costly in three-dimensional cases. Instead of resorting to `gmsh` to generate a new grid, the mesh data structure could be directly manipulated inside `Flow` and the cells could be split or coalesced. Such a procedure would be more robust and computationally efficient, but also requires extensive implementation efforts. As a conclusion, more research is needed on this topic.



(a) Pre-defined grid.

(b) Solution adapted grid.

Figure 7.2.1: Mach number around the RAE 2822 airfoil at $\alpha = 2°$ and $M = 0.715$ obtained from `Flow` by using a pre-defined grid and a solution adaptive grid procedure.

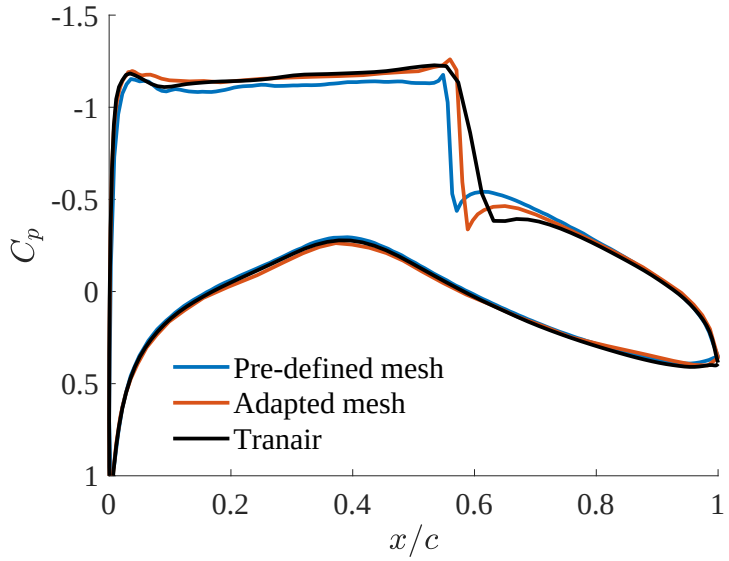


Figure 7.2.2: Pressure coefficient along the chord of the RAE 2822 airfoil at $\alpha = 2°$ and $M = 0.715$ obtained from `Flow` by using a pre-defined grid and a solution adaptive grid procedure, and compared to `Tranair`'s solution.

Adjoint solution

In order to perform aerodynamic or aero-structural optimization calculations, `Flow` must be able to compute the sensitivities of some objective function J, *e.g.* the drag, with respect to some design variables v, *e.g.* the wing shape. The optimization can be formulated as a constrained minimization problem,

$$\begin{aligned} &\min J(\phi, v), \\ &s.t.\ R(\phi, v) = 0, \end{aligned} \tag{7.2.1}$$

where ϕ is the velocity potential and R is the full potential equation. The problem can be solved by constructing the augmented Lagrangian,

$$\mathcal{L} = J - \lambda R, \tag{7.2.2}$$

and requiring that its total derivative vanishes, that is,

$$\delta\mathcal{L} = 0 \Rightarrow \begin{cases} \frac{\partial J}{\partial \phi} - \lambda \frac{\partial R}{\partial \phi} = 0 \\ \frac{\partial J}{\partial v} - \lambda \frac{\partial R}{\partial v} = 0 \\ R = 0 \end{cases} . \tag{7.2.3}$$

The third equation in 7.2.3 expresses that the potential always satisfies the full potential equation, *i.e.* the forward problem. Choosing the Lagrange multipliers λ to satisfy the first equation allows to compute the total derivative of the objective function with respect to the design variables using the second equation, and to pass this derivative to an optimizer. This so-called

adjoint approach allows to compute the total gradient for any number of design variables by solving one linear equation per objective function only. This is advantageous as aerodynamic and aero-structural optimization usually involves a large number of design variables.

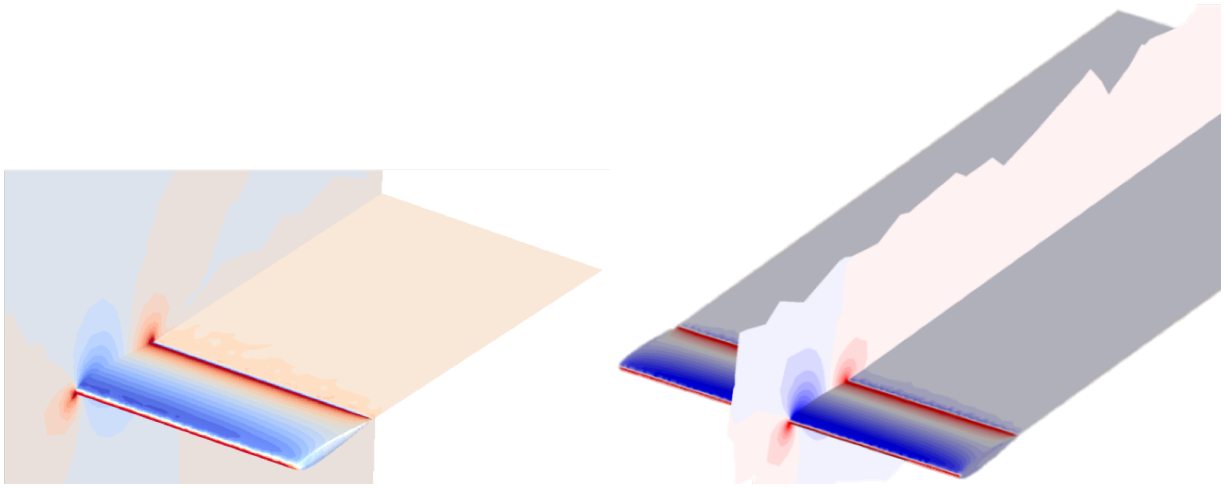
An adjoint feature has been partly implemented in `Flow`. The solver is currently capable of computing the adjoint variables λ_L and λ_D representing the sensitivities with respect to the lift and drag coefficients. The derivative of the potential equation $\frac{\partial R}{\partial \phi}$ appearing in 7.2.3 is computed by re-using the tangent matrix of the forward problem, without taking the contribution of the Kutta condition. The derivative of the objective function is computed analytically as

$$\frac{\partial J}{\partial \phi} = \frac{-2}{S_{\text{ref}}} \int_{\Gamma_{\text{b}}} \rho \mathbf{n} \cdot \hat{\mathbf{d}} \nabla \phi \cdot \nabla \delta\phi \, dS, \tag{7.2.4}$$

where S_{ref} is a reference area, Γ_{b} denotes the boundary of the body, ρ is the isentropic density, $\mathbf{n}$ is the unit normal vector to Γ_{b}, and $\hat{\mathbf{d}}$ is the lift or drag normalized direction. Additionally, a Dirichlet boundary condition requiring the adjoint sensitivities to vanish in the farfield is prescribed on the outer boundaries. Finally, the Kutta condition is enforced by adding the following terms on the lower and upper wake nodes, respectively,

$$\begin{aligned} \int_{\Gamma_{\text{w,l}}} \nabla \delta\phi_{\text{u}} \cdot \mathbf{n} - \nabla \delta\phi_{\text{l}} \cdot \mathbf{n} \, dS = 0, \\ \int_{\Gamma_{\text{w,l}}} \nabla \delta\phi_{\text{u}}^2 - \nabla \delta\phi_{\text{l}}^2 \, dS = 0. \end{aligned} \tag{7.2.5}$$

Figure 7.2.3 shows the Lagrange multiplier associated with the drag for the flow over a rectangular NACA 0012 at angle of attack $\alpha = 5°$ and Mach $M = 0$ compared to the results obtained by Galbraith et al. [13]. Overall, there is a qualitative good agreement although quantitative comparison is not possible. Few mathematical developments exist for the adjoint full potential equation. This is particularly true for the Kutta condition, whose implementation could not be verified. Today, the preferred approach is to use automatic differentiation, for which analytic developments are not necessary. However, using such an approach is not suited for `Flow`, as the tangent matrix is readily available, and recomputing it would unnecessarily increase the computational cost. Further research is thus needed on this topic.



(a) `Flow` solution. (b) Galbraith et al. [13] solution.

Figure 7.2.3: Drag sensitivity for the flow over a NACA 0012 at $\alpha = 5°$ and $M = 0$ obtained using `Flow` and compared to Galbraith et al. [13] solution.

Viscous-inviscid coupling

The results obtained in chapter 2 showed that solving the full potential equation coupled with an integral formulation of the boundary layer equations offered an accuracy similar to solving the Reynolds-Averaged Navier-Stokes equations for only a fraction of the computational cost. The implementation of a viscous-inviscid coupling procedure in `Flow` is thus highly desirable.

During his Master thesis, Bilocq [202] implemented a two-dimensional viscous solver and a viscous-inviscid interaction procedure in `Flow`. The solver is based on the two equations integral formulation of the boundary layer equations developed by Drela [15], and the quasi-simultaneous method developed by Veldman [19] is used as coupling scheme. Both the equations and the coupling scheme are briefly described in chapter 1. Two-dimensional viscous computations performed on the NACA 0012 airfoil at various angles of attack and Mach numbers were compared to `Xfoil` [203, 204] and `SU2`, and overall good agreement was observed. As an example, the pressure and the friction coefficients along the chord of the airfoil at an angle of attack of attack $\alpha = 5°$ and a Mach number $M = 0.5$ are illustrated in Figures 7.2.4 and 7.2.5. Although the viscous-inviscid coupling available in `Flow` yields overall satisfactory results, spurious oscillations are present in the predictions. Moreover, the implementation is restricted to two-dimensional and subcritical flows. Extensions to three-dimensional cases and to treat transonic flows is possible but requires extensive implementation effort.

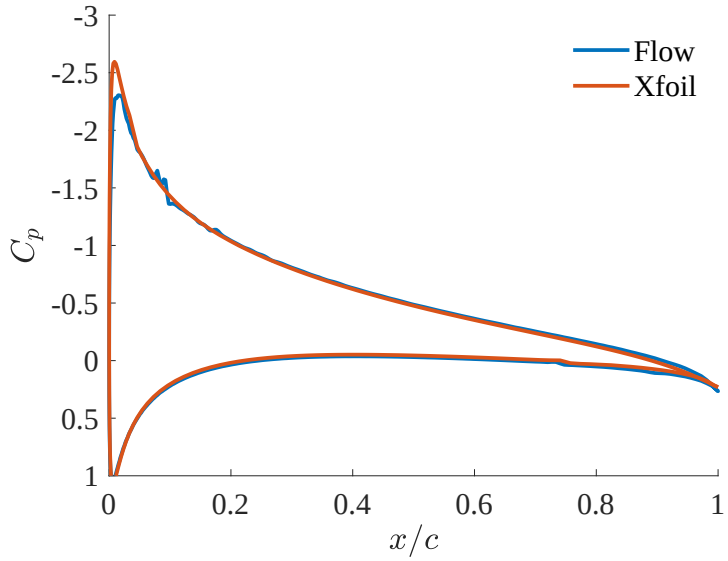
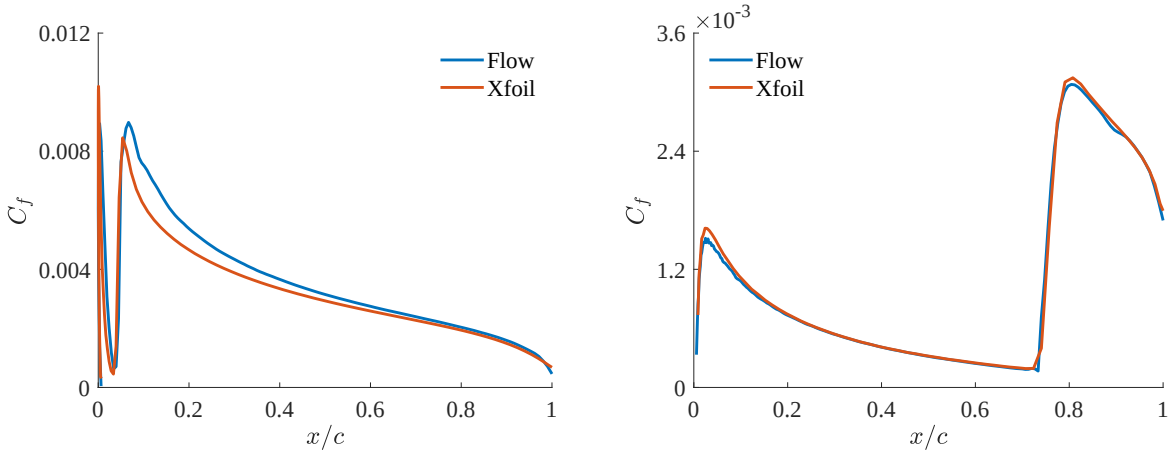


Figure 7.2.4: Pressure coefficient along the chord of the NACA 0012 airfoil at $\alpha = 5°$ and $M = 0.5$ obtained from `Flow` and compared to `Xfoil` (reproduced from Bilocq [202]).



(a) Suction side.

(b) Pressure side.

Figure 7.2.5: Friction coefficient along the chord of the NACA 0012 airfoil at $\alpha = 5°$ and $M = 0.5$ obtained from `Flow` and compared to `Xfoil` (reproduced from Bilocq [202]).

Unsteady flow modeling

The present work only dealt with steady flow computations. However, unsteady flow modeling is also required in aircraft design, especially for flight dynamics or dynamic aeroelastic computations, such as flutter prediction. Several methods can be used to model unsteady nonlinear potential flows: either perform time-accurate computations, or decompose the flow into multiple harmonic solutions.

The unsteady full potential equation is written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \nabla \phi = 0, \tag{7.2.6}$$

where the isentropic density is now time-dependent and given by

$$\rho = \rho_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \left(1 - \frac{2}{M_\infty} \frac{\partial \phi}{\partial t} - |\nabla \phi|^2 \right) \right]^{\frac{1}{\gamma - 1}}. \tag{7.2.7}$$

Equation 7.2.6 can be considered as a function of the potential only. However, it becomes second-order in time and cannot be discretized with unsteady schemes usually used in computational fluid dynamics. Sankar et al. [205], Malone and Sankar [206], and Shankar et al. [207] proposed an adaptation of the steady Newton scheme to deal with the unsteady equation in the context of the finite volume method. However, their formulation is complex and should be investigated further. An alternative solution for solving Equation 7.2.6 consists in considering the density as a second variable, alongside the potential. In that case, the set of equations remains first-order in time and can be treated more easily. Nevertheless, there are two main drawbacks. Firstly, the steady solver cannot be directly re-used and has to be heavily adapted. Secondly, two equations must be solved instead of one, hence increasing the computational cost. Although the finite element formulation proposed by Galbraith et al. [13] is based on such an idea, the authors did not develop their solver for an unsteady framework.

In aerospace applications, a significant part of unsteady computations involves flows that are periodic in time. In such cases, the flow can be decomposed into several harmonics and the potential can be written

$$\phi = \phi_0 + \sum_k \phi_k \exp(\mathrm{i}\omega_k t), \tag{7.2.8}$$

where ϕ_0 is the steady potential, and ϕ_k and ω_k are the amplitude and the frequency of the k^{th} harmonic, respectively. By analogy with structural dynamics, ϕ_k is also referred to as the k^{th} flow mode. This decomposition allows a potential flow to be solved using an harmonic method. The harmonic balance method was initially developed to perform electrical circuit analysis [208], and has gained popularity in computational fluid dynamics over the past years. In the harmonic balance method, the set of equations can be seen as several steady equations coupled together through source terms. If few harmonics are retained, the periodic state can be reached quite quickly and the technique can be inexpensive compared to a time-accurate computation. Although, the harmonic balance method has been successively used to solve the Navier-Stokes and Euler equations, it has never been applied to unsteady potential theory, to the best of the author's knowledge. As such, it could be a new research topic to investigate.

A further simplification to the harmonic decomposition consists in linearizing the unsteady flow about a known steady solution by considering only small perturbations. Equation 7.2.8 can then be simplified by retaining only one flow mode, and the unsteady potential is expanded as

$$\phi = \Phi + \varphi \exp(\mathrm{i}\omega t), \tag{7.2.9}$$

where Φ is the steady potential, and where φ denotes the amplitude of the perturbation and ω, its frequency. Substituting Equation 7.2.9 into Equation 7.2.6 yields,

$$\mathrm{i}\omega\varrho + \nabla \cdot (\rho\nabla\varphi + \varrho\nabla\Phi) = 0, \tag{7.2.10}$$

where the perturbation density is given by,

$$\varrho = -\frac{\rho M^2}{|\nabla\Phi|^2}\left[\mathrm{i}\omega\varphi + \nabla\phi\cdot\nabla\varphi\right]. \tag{7.2.11}$$

Note that Equations 7.2.10 and 7.2.11 are linear in φ. Consequently, for a given set of n perturbation frequencies, one nonlinear steady computation must be performed and n uncoupled sets of linear equations must be solved, which is inexpensive if n is small. The unsteady flow over a wing can be obtained using the following procedure. Firstly, a modal analysis is performed to obtain the structural modes of vibration of the wing. Then, a nonlinear steady computation is performed at a given flight condition. Finally, each structural mode motion is used as a boundary condition, and ω is set to the corresponding natural frequency, in order to solve Equation 7.2.10 and to recover the corresponding flow mode. The flow modes can subsequently be used for transonic flutter computations. Several authors, such as Whitehead [209], Hall [210] and Florea [211], already investigated this procedure. Moreover, `Tranair` and `blwf` implement solution techniques to obtain the flow modes and the flutter speed. Alternatively, the new methodology based on dynamic mode decomposition and interpolation, recently developed by Güner [199], could be used.

In the three unsteady modeling methodologies described in the present section, the most challenging development is the implementation of the Kutta condition. As for steady methods, few authors give details about the Kutta condition. If only small amplitude motions are considered, as it is the case in flutter computations, the modeling of the wake and the formulation of the Kutta condition could be kept similar to the steady case. In such a case, only the expression of the mass flux and the pressure should be adapted to take into account the unsteadiness of the flow.

Bibliography

[1] Statista: number of scheduled passengers boarded by the global airline industry from 2004 to 2020. https://www.statista.com/statistics/564717/airline-industry-passenger-traffic-globally/, Accessed September 2020.

[2] Flightpath 2050 Europe's Vision for Aviation. https://ec.europa.eu/transport/sites/transport/files/modes/air/doc/flightpath2050.pdf, Accessed September 2020.

[3] International Energy Agency: data and reports on CO2 emisions. https://www.iea.org/data-and-statistics/?country=WORLD&fuel=CO2+emissions, https://www.iea.org/reports/tracking-transport-2019, Accessed September 2020.

[4] J.R. Wright and J.E. Cooper. *Static Aeroelasticity and Flutter*, chapter 22, pages 475–480. John Wiley and Sons, 2015.

[5] M.H. Shirk and T.J. Hertz. Aeroelastic Tailoring - Theory, Practise and Promise. *Journal of Aircraft*, 23(1):6–18, January 1986.

[6] P.R. Spalart and S.R. Allmaras. A One-Equation Turbulence Model for Aerodynamic Flows. *Recherche Aerospatiale*, 1:5–21, 1994.

[7] F.R. Menter. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. *AIAA Journal*, 32(8), August 1994.

[8] J.L. Steger and B.S. Baldwin. Shock waves and drag in the numerical calculation of isentropic transonic flows. Technical report, NASA, 1972.

[9] R. Neel. *Advances in Computational Fluid Dynamics: turbulent separated flows and transonic potential flows*. PhD thesis, Virginia Polytechnic Institute, August 1995.

[10] R.M. Lieg. A Full Potential Solver for Lifting Flows on Unstructured Tetrahedral Meshes. Master's thesis, Concordia University, 2005.

[11] F. Lyu, T. Xiao, and X. Yu. A Fast and Automatic Full Potential Finite Volume Solver on Cartesian Grids for Unconventional Configurations. *Chinese Journal of Aeronautics*, 2017.

[12] B. Nishida. *Fully Simultaneous Coupling of the Full Potential Equation and the Integral Boundary Layer Equations in Three Dimensions*. PhD thesis, Massachussets Institute of Technology, February 1996.

[13] M.C. Galbraith, S.R. Allmaras, and R. Haimes. Full Potential Revisited: A Medium Fidelty Aerodynamic Analysis Tool. In *55th AIAA Aerospace Sciences Meeting, SciTech Forum*. AIAA, January 2017.

[14] A. Crovato, R. Boman, H. Güner, V.E. Terrapon, G. Dimitriadis, H.S. Almeida, A.P. Prado, C. Breviglieri, P.H. Cabral, and G.H. Silva. A Full Potential Static Aeroelastic Solver for Preliminary Aircraft Design. In *18th International Forum on Aeroelasticity and Structural Dynamics*. International Forum on Aeroelasticity and Structural Dynamics, June 2019.

[15] M. Drela. *Two-Dimensional Transonic Aerodynamic Design and Analysis Using The Euler Equations*. PhD thesis, Massachussets Institute of Technology, 1985.

[16] M. Drela, M.B. Giles, and W. Thompkins. Newton Solution of Coupled Euler and Boundary-Layer Equations. *Numerical and Physical Aspects of Aerodynamics Flows III*, 1986.

[17] M. Drela and M.B. Giles. Viscous-Inviscid Analysis of Transonic and Low Reynolds Number Airfoils. *AIAA Journal*, 25(10), 1987.

[18] B. Mughal and M. Drela. A calculation method for the three-dimensonal boundary-layer equations in integral form. In *31st Aerospace Sciences Meeting*. AIAA, 1993.

[19] Veldman A.E.P. New, Quasi-simultaneous Method to Calculate Interacting Boundary Layers. *AIAA Journal*, 19(1), 1981.

[20] R.C. Lock and B.C. Williams. Vicous-inviscid Interactions in External Arodynamics. *Progress in Aerospace Science*, 24:51–171, 1987.

[21] R. Kamakoti and W. Shyy. Fluid–structure interaction for aeroelastic applications. *Progress in Aerospace Sciences*, 40(8):535–558, November 2004.

[22] G. Hou, J. Wang, and A. Layton. Numerical Methods for Fluid-Structure Interaction — A Review. *Communications in Computational Physics*, 12(2):337—377, August 2012.

[23] U. Kuttler and W.A. Wall. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, February 2008.

[24] C. Wood, A.J. Gil, O. Hassan, and J. Bonet. Partitioned block-Gauss–Seidel coupling for dynamic fluid–structure interaction. *Computers and Structures*, 88(23):1367–1382, December 2010.

[25] J. Degroote, K.J. Bathe, and J. Vierendeels. Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. *Computers and Structures*, 87(11):793–801, November 2009.

[26] I. Bhateley and R. Cox. *Application of Computational Methods to Transonic Wing Design*, volume 81 of *Progress in Astronautics and Aeronautics*, chapter 8, pages 405–431. American Institute of Aeronautics and Astronautics, Reston, VA, USA, 1981.

[27] A. Verhoff and P.J. O'Neil. *Extension of FLO codes to transonic flow prediction for fighter configurations*, volume 81 of *Progress in Astronautics and Aeronautics*, chapter 11, pages 467–487. American Institute of Aeronautics and Astronautics, Reston, VA, USA, 1981.

[28] P. Rubbert and G. Saaris. Review and evaluation of a three-dimensional lifting potential flow analysis method for arbitrary configurations. In *10th AIAA Aerospace Science Meeting*, San Diego, CA, USA, January 1972.

[29] J. Flores, T.L. Holst, D. Kwak, and D.M. Batiste. Comparison of the Full-Potential and Euler Formulations for Computing Transonic Airfoil Flows. Technical report, NASA, Washington, DC, USA, 1984.

[30] G.H. Klopfer and D. Nixon. Nonisentropic potential formulation for transonic flows. *AIAA Journal*, 22:770–776, 1984.

[31] J. Le Balleur. Strong matching methods for computing transonic viscous flow including wakes and seprations - lifting airfoils. *La Recherche Aerospatiale*, 3:161–185, 1981.

[32] R. Melnik, R. Chow, H. Mead, and A. Jameson. A Multigrid Method for the Computation of Viscid/Inviscid Interaction on Airfoils. Technical report, Grumman Aerospace Corporation, Bethpage, NY, USA, 1983.

[33] J. Van Muijden, A. Broekhuizen, A. van der Wees, and J. van der Vooren. Flow analysis and drag prediction for transonic transport wing/body configurations using a viscous-inviscid interaction type method. In *19th ICAS Congress*, Anaheim, CA, USA, September 1994.

[34] T.L. Holst. Transonic flow computations using nonlinear potential methods. *Progress in Aerospace Sciences*, 36:1–61, 2000.

[35] M. Drela, M. Giles, and W. Thompkins. *Newton Solution of Coupled Euler and Boundary Layer Equations*, volume 3 of *Numerical and Physical Aspects of Aerodynamic Flows*, chapter 7, pages 143–154. Spinger, Berlin/Heidelberg, Germany, 1986.

[36] M. Potsdam. An Unstructured Mesh Euler and Interactive Boundary Layer Method for Complex Configurations. In *12th Applied Aerodynamic Conference*, Colorado Springs, CO, USA, June 1994.

[37] M. Aftosmis, M. Berger, and J. Alonso. Applications of a Cartesian Mesh Boundary-Layer Approach for Complex Configurations. In *44th AIAA Aerospace Science Meeting and Exhibit*, Reno, NV, USA, January 2006.

[38] A. Jameson. The Evolution of Computational Methods in Aerodynamics. *Journal of Applied Mechanics*, 50:1052–1070, 1983.

[39] F. Johnson, E. Tinoco, and N. Yu. Thirty Years of development and applications of CFD at Boeing Commercial Airplanes. *Computational Fluids*, 34:1115–1151, 2005.

[40] A. Crovato, H.S. Almeida, G. Vio, G.H. Silva, A.P. Prado, C. Breviglieri, H. Güner, P.H. Cabral, R. Boman, V.E. Terrapon, and G. Dimitriadis. Effect of Levels of Fidelity on Steady Aerodynamic and Static Aeroelastic Computations. *Aerospace*, 7(4):42, April 2020.

[41] F. Palacios, M.R. Colonno, A.C. Aranake, A. Campos, S.R. Copeland, T.D. Economon, A.K. Lonkar, T.W. Lukaczyk, T.W.R. Taylor, and J.J. Alonso. Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design. *AIAA Journal*, 2013.

[42] T.D. Economon, F. Palacios, S.R. Copeland, T.W. Lukaczyk, and J.J. Alonso. Stanford University Unstructured (SU2): An open-source suite for multi-physics simulation and design. *AIAA Journal*, 2016.

[43] Stanford University Unstructured - SU2 v6.2. https://su2code.github.io/, Accessed September 2020.

[44] A. Jameson. Origins and Further Development of the Jameson–Schmidt–Turkel Scheme. *AIAA Journal*, 55(5), May 2017.

[45] F.T. Johnson, Samant S.S., M.B. Bieterman, R.G. Melvin, D.P. Young, J.E. Bussoletti, and C.L. Hilmes. Tranair: A Full-Potential, Solution-Adaptative, Rectangular Grid-Code for Predicting Subsonic, Transonic, and Supersonic Flows About Arbitrary Configurations. Technical report, NASA, 1992.

[46] M.B. Bieterman, R.G. Melvin, F.T. Johnson, J.E. Bussoletti, D.P. Young, W.P. Huffman, C.L. Hilmes, and M. Drela. Boundary Layer Coupling in a General Configuration Full Potential Code. Technical report, The Boeing Company, Seattle, WA, 1994.

[47] R.E. Bank and D.J. Rose. Global Approximate Newton Method. *Numerische Mathematik*, 27:179–295, 1981.

[48] R.L. Carmichael and L.L. Erickson. Panair: A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. *AIAA Journal*, 7(2), 1981.

[49] Panair. https://pdas.com/panair.html, Accessed September 2020.

[50] E. Albano and W.P. Rodden. A Doublet-Lattice Method for calulation lift distributions on oscillating surfaces in subsonic flows. *AIAA Journal*, 1969.

[51] NASTRAN. https://mscsoftware.com/products/msc-nastran, Accessed September 2020.

[52] C. Reschke and T. Kier. An Integrated Model for Aeroelastic Simulation of large flexible Aircraft using MSC. NASTRAN. Technical report, DLR German Aerospace Center - Institute of Robotics and Mechatronics, Wessling, Germany, September 2004.

[53] W.P. Rodden and E.H. Johnson. *NASTRAN Aeroelastic Analysis User's Guide*. MSC Software, 1994.

[54] V. Schmitt and F. Charpin. Pressure distributions on the ONERA-M6-wing at transonic Mach numbers. *Experimental data base for computer program assessment*, 4, 1979.

[55] G.H.C. Silva, A.P. Prado, P.H. Cabral, R. De Breuker, and J.K.S. Dillinger. Tailoring of a Composite Regional Jet Wing Using the Slice and Swap Method. *Journal of Aircraft*, 56(3):990–1004, January 2019.

[56] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79:1309–1331, 2009.

[57] Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. http://gmsh.info, Accessed September 2020.

[58] ANSYS ICEM CFD. https://www.ansys.com/products/fluids, Accessed September 2020.

[59] E. Murman and J. Cole. Calculation of plane steady transonic flows. *AIAA Journal*, 1971.

[60] J.A. Krupp and E.M. Murman. The numerical calculation of steady transonic flows past thin lifting airfoils and slender bodies. *AIAA Journal*, 1972.

[61] W.F. Jr Ballhaus and F.R. Bailey. Numerical calculations of transonic flow about swept wings. *AIAA paper*, 1972.

[62] P.A. Newman and F.B. Klunker. Computation of transonic flow about finite lifting wings. *AIAA Journal*, 1972.

[63] S. Rolfs and R. Vanino. A steady relaxation method for two- and three-dimensional transonic flows. In *Euromech 40*, Sweden, 1973.

[64] W. Schmidt and S. Hedman. Recent explorations in relaxation methods for three-dimensional transonic potential flow. *ICAS paper*, 1976.

[65] W. Mason, D.A. Mackenzie, M.A. Stern, and Johnson J.K. A numerical three-dimensional viscous transonic wing-body analysis and design tool. *AIAA paper*, 1978.

[66] C.M. Albone, M.G. Hall, and G. Joyce. Numerical solutions for transonic flows past wing-body configurations. In Springer, editor, *Symposium Transonicum II*, Berlin, 1975.

[67] M.C.P. Firman. Calculations of transonic flow over wing/body combinations with an allowance for viscous effects. Technical report, AGARD, 1981.

[68] V. Shankar and N.D. Malmuth. Computational treatment of three-dimensional canard-wing interactions. *Journal of Aircraft*, 1983.

[69] W.J. Rae. Calculation of three-dimensional transonic compressor flow fields by a relaxation method. *Journal of energy*, 1977.

[70] W.J. Rae and J.A. Lordi. A study of inlet conditions for three-dimensional transonic compressor flows. Technical report, Calspan, 1978.

[71] F.R. Bailey and W.F. Jr Ballhaus. Comparison of computed and experimental pressures for transonic flows about isolated wings and wing-fuselage configurations. Technical report, NASA, 1975.

[72] J.L. Steger and H. Lomax. Numerical calculation of transonic flow bout two-dimensional airfoils by relaxation procedures. *AIAA Journal*, 1972.

[73] P.R. Garabedian and D.G. Korn. Analysis of transonic airfoils. *Communications on Pure and Applied Mathematics*, 1971.

[74] D.M. Young. *Iterative Solution of Large Linear Systems*. Elsevier, 1 edition, July 1971.

[75] F. Bauer, P. Garabedian, D. Korn, and A. Jameson. Supercritical Wing Section II. *Lecture Notes in Economics and Mathematical Systems*, 1975.

[76] J.C. Jr South and A. Jameson. Relaxation solutions for inviscid axisymmetric transonic flow over blunt or pointed bodies. In *First AIAA CFD Conference*, Palm Springs, CA, 1973. AIAA.

[77] J.D Keller and J.C. Jr South. *RAXBOD: A fortran program for inviscid transonic flow over axisymmetric bodies*. NASA, 1976.

[78] J.L. Steger and H. Lomax. Calculation of transonic flow around axisymmetric inlets. *AIAA Journal*, 1975.

[79] T.J. Baker. A numerical method to compute inviscid transonic flow around axisymmetric ducted bodies. In Springer, editor, *Symposium Transonicum II*, Berlin, 1975.

[80] D.A. Caughey and A. Jameson. Accelerated iterative calculation of transonic nacelle flow fields. *AIAA Journal*, 1977.

[81] Reyhner T. Cartesian mesh solution for axisymmetric transonic potential flow around inlets. *AIAA Journal*, 1977.

[82] D.C. Ives and Liutermoza J.F. Analysis of transonic cascade flow using conformal mapping and relaxation techniques. *AIAA Journal*, 1977.

[83] D.C. Ives and Liutermoza J.F. Second-order-accurate calculation of transonic flow over turbomachinery cascades. *AIAA Journal*, 1979.

[84] L.A. Carlson. Transonic airfoil analysis and design using Cartesian coordinates. *Journal of Aircraft*, 1976.

[85] A. Jameson. Iterative solution of transonic flows over airfoils and wings, including flows at mach 1. *Communications on Pure and Applied Mathematics*, 1974.

[86] A. Jameson, D.A. Caughey, P.A. Newman, and R.M. Davis. *A brief description of the Jameson-Caughey NYU transonic swept-wing computer program FLO-22*. NASA, 1976.

[87] P.A. Henne and R.M. Hicks. *Wing analysis using a transonic potential flow computational method*. NASA, 1978.

[88] A. Jameson. Transonic potential flow calculation using conservative form. In *Second AIIA CFD Conference*, pages 145–155, 1975.

[89] A. Jameson and D.A. Caughey. A finite-volume method for transonic potential flow calculations. In *Third AIIA CFD Conference*, pages 35–54, 1977.

[90] A. Jameson and D.A. Caughey. Numerical calculation of transonic potential flow about wing-body combinations. *AIAA Journal*, 1979.

[91] D.A. Caughey and A. Jameson. Progress in finite-volume calculations for wing-fuselage combinations. *AIAA Journal*, 1980.

[92] L.T. Chen, K.C. Yu, and T.Q. Dang. Transonic computational method for an aft-mounted nacelle/pylon with power effect. *Journal of Aircraft*, 27, 1990.

[93] J.F. Thompson, F.C. Thames, and C.W. Mastin. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 15, 1974.

[94] N.J. Yu. Transonic flow simulations for complex configurations with surface fitted grids. *AIAA paper*, 1981.

[95] Street C. Viscous-inviscid interaction for transonic wing-body configurations including wake effects. *AIAA Journal*, 20, 1982.

[96] S.H. Woodson, J.F. Campbell, and F.R. De Jarnette. Interactive three-dimensional boundary-layer method for transonic flow over swept wings. *AIAA Journal*, 29, 1991.

[97] G H. Klopfer and D. Nixon. Nonisentropic Potential Formulation for Transonic Flows. In *21st AIAA Aerospace Sciences Meeting*, Reno, Nevada, January 1983. AIAA.

[98] NACA. Equations, tables, and charts for compressible flow. Technical report, NACA, 1953.

[99] A. Parrinello and P. Mantegazza. Improvements and extensions to a full-potential formulation based on independent fields. *AIAA Journal*, 50(3), March 2012.

[100] A. Eberle. *A finite volume method for calculating transonic potential flow around wings from the pressure minimum integral*. NASA, 1978.

[101] T.L. Holst and W.F. Jr Ballhaus. Fast conservative schemes for the full potential equation applied to transonic flows. *AIAA Journal*, 1979.

[102] M.M. Hafez, E.M. Murman, and J.C. Jr South. Artificial compressibility methods for numerical solution of transonic full potential equation. *AIAA Journal*, 1979.

[103] B. Engquist and S. Osher. Stable and entropy satisfying approximations for transonic flow calculations. *Mathematics of Computation*, 1980.

[104] P.M. Goorjian and R. Van Buskirk. Implicit calculations of transonic flows using monotone methods. *AIAA paper*, 1981.

[105] P.M. Goorjian, M.C. Meagher, and R. Van Buskirk. Monotone implicit algorithms for the small-disturbance and full potential equations applied to transonic flows. *AIAA paper*, 1983.

[106] J.W. Boerstoel. A multigrid algorithm for steady transonic potential flows around aerofoils using newton iteration. Technical report, NASA, 1981.

[107] J. Slooff. Some new developments in exact integral equation formulations for sub or transonic compressible potential flow. *Computational Mechanics Publications*, 1982.

[108] S. Osher, M. Hafez, and W. Jr Whitlow. Entropy condition satisfying approximations for the full potential equations of transonic flow. *Mathematics of Computation*, 1985.

[109] M.M. Hafez, W. Jr Whitlow, and S. Osher. Improved finite difference schemes for transonic potential fow calculations. *AIAA Journal*, 1987.

[110] W.G. Habashi and M.M. Hafez. Finite element solutions of transonic flow problems. *AIAA Journal*, 1982.

[111] G. Volpe and A. Jameson. Transonic potential flow calculations by two artificial density methodss. *AIAA Journal*, 1988.

[112] G.S. Dulikravich. Analysis of artificial dissipation models for the transonic full potential equation. *AIAA Journal*, 1988.

[113] M.O. Bristeau, O. Pironneau, R. Glowinsky, J. Periaux, P. Perrier, and G. Poirier. On the numerical solution of nonlinear problems in fluid dynamics by least square and finite element methods. Application to transonic flow simulations. *Computer methods in applied mechanics and engineering*, 1985.

[114] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 1986.

[115] M. Drela and M.B. Giles. ISES: A two dimensional viscous aerodynamic design and analysis code. *AIAA paper*, 1987.

[116] W.P. Huffman, R.G. Melvin, D.P. Young, F.T. Johnson, and J.E. Bussoletti. Practical Design and Optimization in Computational Fluid Dynamics. *AIAA paper*, July 1993.

[117] D.P. Young, W.P. Huffman, R.G. Melvin, M.B. Bieterman, C.L. Hilmes, and F.T. Johnson. Inexactness and Global Convergence in Design Optimization. *AIAA paper*, September 1994.

[118] R.G. Melvin, W.P. Huffman, D.P. Young, F.T. Johnson, C.L. Hilmes, and M.B. Bieterman. Recent Progress in Aerodynamic Design and Optimization. *Internatonal Journal for Numerical Methods in Fluids*, 30:205–216, 1999.

[119] D.P. Young, W.P. Huffman, R.G. Melvin, C.L. Hilmes, and F.T. Johnson. Nonlinear Elimination in Aerodynamic analysis and Design Optimization. In Springer Verlag, editor, *Large-Scale PDE-constrained Optimization*, Lecture Notes in Computational Science and Engineering, pages 17–44. Biegler, L., October 2004.

[120] The Boeing Company. *Tranair manual*, August 2009.

[121] O.V. Karas and V.E. Kovalev. BLWF56 presentation. blwf-aero.ru/BLWF_code/index_en.html, Accessed September 2020.

[122] T. Holst and S. Thomas. Numerical solution of transonic wing flow fields. *AIAA Journal*, 21, 1983.

[123] B. Engquist and S. Osher. One sided difference approximations for nonlinear conservation laws. *Mathematics of Computation*, 36:45–75, 1980.

[124] T. Holst. Multizone chimera algorithm for solving the full-potential equation. *Journal of Aircraft*, 35, 1998.

[125] Viscous full-potential (VFP) method for three-dimensional wings and wing-body combinations. Part 1: Validation of VFP results with experiment and comparisons with other methods, Accessed September 2020.

[126] S. Prince, D. Di Pasquale, K. Garry, and Nuzzo C. A Rapid Aerodynamic Prediction Method for Unconventional Transonic Aircraft Configurations. In *31st Congress of the International Council of the Aeronautical Sciences*, Belo-Horizonte, Brazil, September 2018. International Council of the Aeronautical Sciences.

[127] A. Parrinello and P. Mantegazza. Independent two-fields solution for full-potential unsteady transonic flows. *AIAA Journal*, 48(7), July 2010.

[128] D.J. Kinney. Finite Element Simulations of Compressible Inviscid and Viscous Flows. Master's thesis, Department of Mechanical Engineering, U.C. Davis, 1992.

[129] D.J. Kinney. *Finite Element Solution of the Full Potential Equation Over Aircraft Configurations Using Unstructured Tetrahedrai Grids*. PhD thesis, Department of Mechanical and Aeronautical Engineering, U.C. Davis, 1994.

[130] D.J. Kinney, M.M. Hafez, and P.A. Gelhausen. Validation of a new unstructured full potential formulation. *AIAA paper*, 1995.

[131] D.J. Kinney, M.M. Hafez, and P.A. Gelhausen. An unstructured full potential boundary layer formulation. *AIAA paper*, 1997.

[132] D.J. Kinney, J.R. Gloudemans, and M.M. Hafez. The finite element solution of the full potential equation over aircraft configurations using unstructured tetrahedral and prismatic grids. *AIAA paper*, 1994.

[133] D.J. Kinney and Hafez. Finite element computations of transonic potential flow. In *International Conference on Finite Elements in Fluids*, Venezia, Italy, 1995.

[134] D.J. Kinney, A.S. Hahn, and P.A. Gelhausen. Comparison of low and high nacelle subsonic transport configurations. *AIAA paper*, 1997.

[135] D. Eller. Fast, unstructured-mesh finite-element method for nonlinear subsonic flows. *Journal of Aircraft*, 2012.

[136] G. Redeker, R. Muller, D. Isaacs, , and R. Hirdes. A Selection of Experimental Test Cases for the Validation of CFD Codes. Technical Report 2, AGARD, August 1994.

[137] M. Davari, R. Rossi, P. Dadvand, I. Lopez, and R. Wuchner. A cut finite element method for the solution of the full potential equation with an embedded wake. *Computational Mechanics*, 2018.

[138] A. Rottgermann. *Eine Methode zur Berücksichtigung kompressibler und transsonischer Effekte in Randelementverfahren*. PhD thesis, Institut fur Aerodynamik und Gasdynamik der Universit at Stuttgart, 1995.

[139] W.J. Piers and J.W. Slooff. Calculation of Transonic Flow by means of a Shock Capturing Field Panel Method. *AIAA Journal*, 1979.

[140] B. Sanderse. Cartesian grid methods for preliminary aircraft design. Master's thesis, T.U. Delft, 2008.

[141] F.T. Johnson, R.M. James, J.E. Bussoletti, A.C. Woo, and D.P. Young. A Transonic Rectangular Grid Embedded Panel Method. In *Third Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference*. AIAA/ASME, 1982.

[142] D.P. Young, A.C. Woo, J.E. Bussoletti, and F.T. Johnson. An Exterior Poisson Solver using Fast Direct Methods and Boundary Integral Equations with Applications to Nonlinear Potential Flow. *SIAM Journal on Applied Mathematics*, 7(3), July 1986.

[143] L. Erickson and S. Strande. A theoretical basis for extending surface-paneling methods to transonic flow. *AIAA Journal*, 23(12), December 1985.

[144] FLO codes. http://aero-comlab.stanford.edu/jameson/flo_codes.html, Accessed Septmeber 2020.

[145] J. Flores, T.L. Holst, D. Kwak, and D.M. Batiste. A New Consistent Spatial Differencing Scheme for the Transonic Full Potential Equation. *AIAA paper*, 1983.

[146] T.H. Pulliam, D.C. Jespersen, and R.E. Childs. An Enhanced Version of an Implicit Code for the Euler Equations. *AIAA paper*, 1983.

[147] P.M. Sinclair. An exact integral (field panel) method for the calculation of two-dimensional transonic potential flow around complex configurations. *The Aeronautical Journal*, 1986.

[148] P.M. Sinclair. A three-dimensional field-integral method for the calculation of transonic flow on complex configurations — theory and preliminary results. *The Aeronautical Journal*, 1988.

[149] A. Rottegermann and S. Wagner. Cost Efficient Calculation of Compressible Potential Flow Around a Helicopter Rotor Including Free Vortex Sheet. In *AGARD FDP Symposium on Aerodynamics and Aeroacoustics of Rotorcraft*. AGARD, October 2017.

[150] A. Rottgermann, R. Behr, Ch. Schottl, and S. Wagner. *Calculation of Blade-Vortex Interaction of Rotary Wings in Incompressible Flow by an Unsteady Vortex-Lattice Method Including Free Wake Analysis*, volume 33-7 of *Numerical Techniques for Boundary Element Methods, Notes on Numerical Fluid Mechanics*, chapter 15, pages 153–166. Vieweg Verlag, 1992.

[151] F.X. Caradonna and C. Tung. Experimental and Analytical Studies of a Model Helicopter Rotor in Hover. In *Sixth European Rotorcraft and Powered Lift Aircraft Forum*, September 1980.

[152] L. Gebhardt, D. Fokin, T. Lutz, and S. Wagner. An Implicit-Explicit Dirichlet-Based Field Panel Method for Transonic Aircraft Design. *AIAA Journal*, 2002.

[153] P. Dadvand, R. Rossi, and E. Onate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of Computational Methods in Engineering*, 17(3):253–297, 2010.

[154] Kratos Multiphysics. https://github.com/KratosMultiphysics/Kratos, Accessed September 2020.

[155] J. Katz and A. Plotkin. *Low-Speed Aerodynamics*. Cambridge Aerospace Series. Cambridge University Press, 2001.

[156] L-C. Chu, E. Yates, and O. Kandil. Integral Equation Solution of the Full Potential Equation for Transonic Flows. In *27th Aerospace Sciences Meeting*, 1989.

[157] J.L. Hess and A.M.O. Smith. Calculation of potential flow about arbitrary bodies. *Progress in Aerospace Sciences*, 8:1 – 138, 1967.

[158] Z.F. Seidov and P.I. Skvirsky. Gravitational potential and energy of homogeneous rectangular parallelepiped. *arXiv:astro-ph/0002496*, February 2000.

[159] A. Crovato, G. Dimitriadis, and V. Terrapon. Higher Fidelity Transonic Aerodynamic Modeling for Preliminary Aircraft Design. In *31st Congress of the International Council of the Aeronautical Sciences*. International Council of the Aeronautical Sciences, September 2018.

[160] Aero v1.0.1 - a Field Panel Method for Aerodynamic Loads Computation in Preliminary Aircraft Design. https://github.com/acrovato/aero, Accessed September 2020.

[161] I.E. Sutherland, R.F. Sproull, and R.A. Schumacker. A Characterization of Ten Hidden-Surface Algorithms. *ACM Comput. Surv.*, 6(1):1–55, March 1974.

[162] M. Shimrat. Algorithm 112: Position of Point Relative to Polygon. *Commun. ACM*, 5(8), August 1962.

[163] B. Maskew. Subvortex Technique for the Close Approach to a Discretized Vortex Sheet. *Journal of Aircraft*, 4(2), February 1977.

[164] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[165] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.

[166] Y.J. Liu and N. Nishimura. The fast multipole boundary element method for potential problems: A tutorial. *Engineering Analysis with Boundary Elements*, 30(5):371–381, 2006.

[167] D.J. Willis, J. Peraire, and J.K. White. A combined pFFT-multipole tree code, unsteady panel method with vortex particle wakes. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, January 2005.

[168] J. Mooren. A Fast, Unstructured Panel Solver. Fall 2012.

[169] R.P. Dwight. Robust Mesh Deformation using the Linear Elasticity Equations. *Journal of Computational Fluid Dynamics*, 12:401–406, 2009.

[170] R. Pelz and A. Jameson. Transonic flow calculations using triangular finite elements. *AIAA Journal*, 23:569–576, April 1985.

[171] Flow v1.9 - a open-source, unstructured finite elements, full potential solver. https://gitlab.uliege.be/am-dept/waves/tree/master/flow, Accessed September 2020.

[172] waves. https://gitlab.uliege.be/am-dept/waves, Accessed September 2020.

[173] D. M. Beazley. SWIG: An easy to use tool for integrating scripting languages with C and C++. In *Proceedings of the 4th conference on USENIX Tcl/Tk Workshop*, volume 4, page 15. USENIX Association, 1996.

[174] Intel threading building blocks. https://software.intel.com/en-us/tbb, Accessed September 2020.

[175] Intel threading building blocks. https://github.com/intel/tbb, Accessed September 2020.

[176] Pardiso: Parallel Direct Sparse Solver. https://software.intel.com/en-us/node/470282, Accessed September 2020.

[177] MUMPS: MUltifrontal Massively Parallel sparse direct Solver. http://mumps.enseeiht.fr/, Accessed September 2020.

[178] W. Sun and Y.X. Yuan. *Optimization Theory and Methods - Nonlinear Programming*. Springer, 2006.

[179] K. Jovanov and R. De Breuker. Accelerated convergence of high-fidelity aeroelasticity using low-fidelity aerodynamics. In *16th International Forum on Aeroelasticity and Structural Dynamics*, St. Petersburg, Russia, July 2015.

[180] G. Kenway and J. Martins. Multipoint High-fidelity Aerostructural Optimization of a Transport Aircraft Configuration. *Journal of Aircraft*, 51:144–160, 2014.

[181] T. Brooks, G. Kenway, and J. Martins. Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings. *AIAA Journal*, 56:2840–2855, 2018.

[182] J. Heeg, P. Chwalowski, and D. Schuster. Overview and lessons learned from the aeroelastic prediction workshop. In *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Boston, MA, USA, April 2013.

[183] D. Schuster, J. Heeg, C. Wieseman, and P. Chwalowski. Analysis of Test Case Computations and Experiments for the Aeroelastic Prediction Workshop. In *51st AIAA Aerospace Sciences Meeting*, Grapevine, TX, USA, January 2013.

[184] G. Romanelli, M. Castellani, P. Mantegazza, and S. Ricci. Coupled CSD/CFD non-linear aeroelastic trim of free-flying flexible aircraft. In *53rd AIAA/SME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Honolulu, HI, USA, April 2012.

[185] P. Acar and M. Nikbay. Steady and Unsteady Aeroelastic Computations of HIRENASD Wing for Low and High Reynolds Numbers. In *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Boston, MA, USA, April 2013.

[186] J. Edwards and J. Malone. Current status of computational methods for transonic unsteady aerodynamics and aeroelastic applications. *Computational System Engineering*, 3:545–569, 1992.

[187] D. Schuster, D. Liu, and L. Huttshell. Computational Aeroelasticity: Success, Progress, Challenge. *Journal of Aircraft*, 40:843–856, 2003.

[188] M.D.C. Henshaw, K. Badcock, G. Vio, C. Allen, J. Chamberlain, I. Kaynes, G. Dimitriadis, J. Cooper, M. Woodgate, A. Rampurawala, Jones, Fenwick, Gaitonde, Taylor, Amor, Eccles, and Denley. Non-linear aeroelastic prediction for aircraft applications. *Progress in Aerospace Science*, 43:65–137, 2007.

[189] modali v1.0.1 - a modal solver for FSI computations. https://github.com/ulgltas/modali, Accessed September 2020.

[190] D. Thomas, A. Variyar, R. Boman, T. Economon, J. Alonso, G. Dimitriadis, and V.E. Terrapon. Staggered strong coupling between existing fluid and solid solvers through a python interface for fluid-structure interaction problems. In *VII International Conference on Computational Methods for Coupled Problems in Science and Engineering*, pages 645–660, 2017.

[191] D. Thomas, M.L. Cerquaglia, R. Boman, T. Economon, J. Alonso, G. Dimitriadis, and V.E. Terrapon. CUPyDO: An integrated Python environment for coupled fluid-structure problems. *Advances in Engineering Software*, 2019.

[192] M.L. Cerquaglia, D. Thomas, R. Boman, V.E. Terrapon, and J.-P. Ponthot. A fully partitioned Lagrangian framework for FSI problems characterized by free surfaces, large solid deformations and displacements, and strong added-mass effects. *Computer Methods in Applied Mechanics and Engineering*, In press, 2019.

[193] CUPyDO v1.2.3 - Python tools for partitioned fluid-structure coupling. http://github.com/ulgltas/cupydo, Accessed September 2020.

[194] E.C. Yates. AGARD standard aeroelastic configurations for dynamic response I: Wing 445.6. Technical report, AGARD, 1988.

[195] R.B. Melville, S.A. Mortan, and D.P. Rizzetta. Implementation of a fully-implicit, aeroelastic navier-stokes solver. In *13th Computational Fluid Dynamics Conference*. AIAA, June-July 1997.

[196] G. Goura. *Time marching analysis of flutter using computational fluid dynamics*. PhD thesis, University of Glasgow, 2001.

[197] METAFOR, A nonlinear finite element code, University of Liege. http://metafor.ltas.ulg.ac.be/, Accessed September 2020.

[198] H. Güner, D. Thomas, G. Dimitriadis, and V.E. Terrapon. Unsteady aerodynamic modeling methodology based on dynamic mode interpolation for transonic flutter calculations. *Journal of Fluids and Structures*, 84:218–232, January 2019.

[199] H. Güner. *Unsteady Aerodynamic Modeling Methodology for Transonic Flutter Calculations*. PhD thesis, University of Liège, May 2020.

[200] A. Parrinello, M. Morandini, and P. Mantegazza. Automatic Embbeding of Potential Flow Wake Surfaces in Generic Monolithic Unstructured Meshes. *Journal of Aircraft*, 50(4), July-August 2013.

[201] K. Schmidmayer, F. Petitpas, and E. Daniel. Adaptive mesh refinement algorithm based on dual trees for cells and faces for multiphase compressible flows. *Journal of Computational Physics*, 388, March 2019.

[202] A. Bilocq. Implementation of a viscous-inviscid interaction scheme in a finite element full potential solver. Master's thesis, University of Liège, June 2020.

[203] M. Drela. XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils. In *Low Reynolds Number Aerodynamics*, pages 1–12. Springer Berlin Heidelberg, 1989.

[204] M. Drela. Xfoil. https://web.mit.edu/drela/Public/web/xfoil/, Accessed September 2020.

[205] N.L. Sankar, J.B. Malone, and Y. Tassa. An implicit conservative algorithm for steady and unsteady three-dimensional transonic potential flows. Technical report, Lockheed Georgia company, 1981.

[206] J.B. Malone and N.L. Sankar. Numerical Simulation of Two-Dimensional Unsteady Transonic Flows using the Full-Potential Equation. *AIAA Journal*, 22(8), August 1984.

[207] V. Shankar, H. Ide, J. Gorski, and S. Osher. A fast, time-acurate unsteady full potential scheme. Technical report, Rockwell International science center, 1985.

[208] R.J. Gilmore and M.B. Steer. Nonlinear circuit analysis using the method of harmonic balance - a review of the art. part i. introductory concepts. *International Journal of Microwave and Millimeter-Wave Computer-Aided Engineering*, 1(1):22–37, 1991.

[209] D.S. Whitehead. A finite element solution of unsteady two-dimensional flow in cascades. *International Journal for Numerical Methods in Fluids*, 10:13–34, 1990.

[210] K.C. Hall. A deforming grid variational principle and finite element method for computing small disturbances flows in cascades. In *30th Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 1992. AIAA.

[211] R. Florea and K.C. Hall. Eigenmode analysis of Unsteady Flows about Airfoils. *Journal of Computational Physics*, 147:568–593, July 1998.

[212] D. Van Heesch. Doxygen. http://www.doxygen.nl/, Accessed September 2020.

[213] geoGen v1.0.1 - a gmsh geometry generator for CFD solvers. https://github.com/acrovato/geoGen, Accessed September 2020.

[214] VTK: The Visualization ToolKit. https://vtk.org, Accessed September 2020.

[215] Paraview, an open-source, multi-platform data analysis and visualization application. https://www.paraview.org, Accessed September 2020.

[216] SciPy. https://docs.scipy.org/doc/scipy/reference/index.html, Accessed September 2020.

[217] PyOpt. http://www.pyopt1.org, Accessed September 2020.

Appendix A

Additional computations for the Embraer benchmark wing

The present appendix provides additional computations performed on the Embraer benchmark wing and complements the results given in chapter 2.

A.1 Effect of the turbulence model on viscous computations

This section shows the effect of the turbulence model on the computations performed on the Embraer benchmark wing at maneuvering speeds using the Reynolds-Averaged Navier-Stokes level of fidelity. The results obtained using the Spalart-Allmaras (SA) [6] and the Menter's $k-\omega$ Shear Stress Transport (SST) [7] models implemented in `SU2` are compared to those obtained using the viscous-inviscid interaction capability implemented in `Tranair`. Overall, there is a good agreement between the results obtained using the two turbulence models. Furthermore, the difference between the results obtained using the SA and SST models is slightly smaller than that between the results obtained using `SU2` and `Tranair`, except for the difference in the moment coefficient.

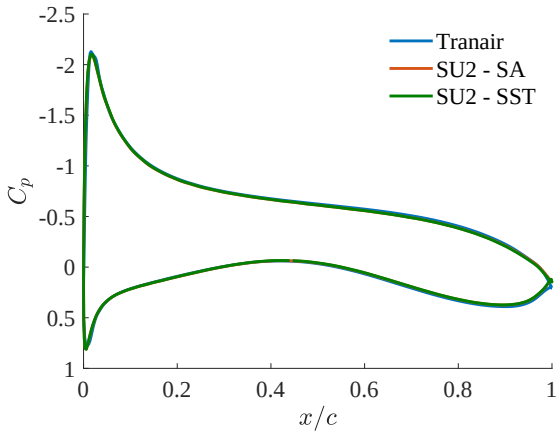
Model	α	C_L	C_D	C_M
`Tranair`	$+3.6$	0.80	0.0310	-1.025
`SU2` SA	$+3.8$	0.80	0.0317	-1.018
`SU2` SST	$+3.9$	0.80	0.0312	-1.015

(a) $M = 0.50$ and $C_L = 0.80$.

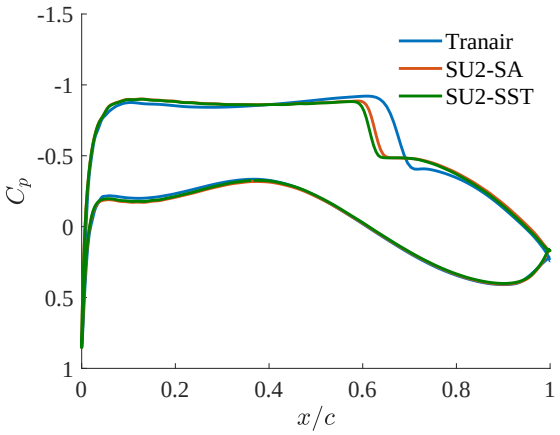
Model	α	C_L	C_D	C_M
`Tranair`	$+0.2$	0.60	0.0241	-0.815
`SU2` SA	$+0.4$	0.60	0.0244	-0.819
`SU2` SST	$+0.5$	0.60	0.0242	-0.821

(b) $M = 0.78$ and $C_L = 0.60$.

Table A.1.1: Aerodynamic coefficients of the Embraer benchmark wing for low and high-speed maneuvers obtained from the `SU2`'s Spalart-Allmaras and $k-\omega$ Shear Stress Transport turbulence models and compared to `Tranair`'s results.

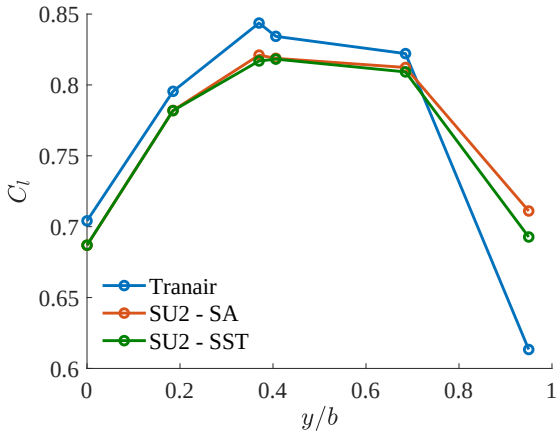


(a) $M = 0.50$, $C_L = 0.80$.

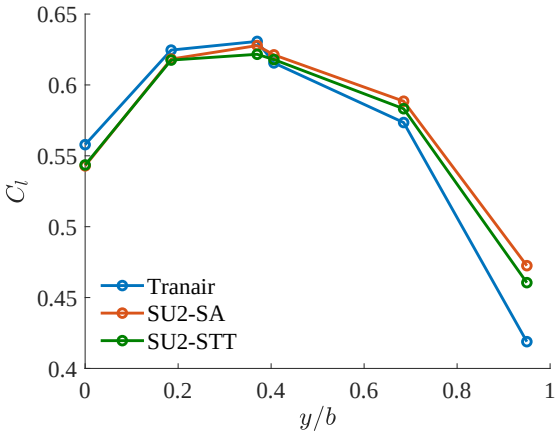


(b) $M = 0.78$, $C_L = 0.60$.

Figure A.1.1: Pressure distribution along the mean aerodynamic chord of the Embraer benchmark wing for low and high-speed maneuvers obtained from the `SU2`'s Spalart-Allmaras and $k - \omega$ Shear Stress Transport turbulence models and compared to `Tranair`'s results.

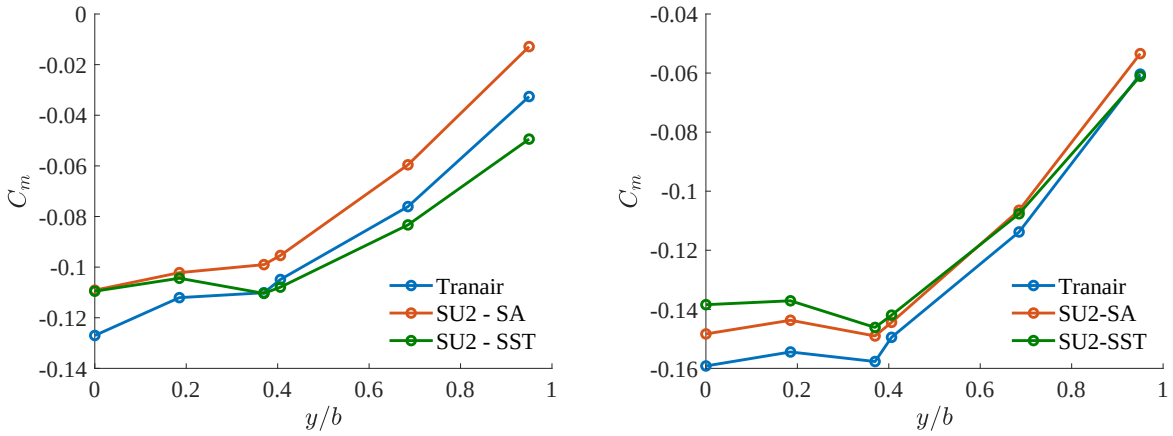


(a) $M = 0.50$, $C_L = 0.80$.



(b) $M = 0.78$, $C_L = 0.60$.

Figure A.1.2: Sectional lift distribution along the span of the Embraer benchmark wing for low and high-speed maneuvers obtained from the `SU2`'s Spalart-Allmaras and $k - \omega$ Shear Stress Transport turbulence models and compared to `Tranair`'s results.



(a) $M = 0.50$, $C_L = 0.80$.

(b) $M = 0.78$, $C_L = 0.60$.

Figure A.1.3: Sectional moment distribution along the span of the Embraer benchmark wing for low and high-speed maneuvers obtained from the `SU2`'s Spalart-Allmaras and $k - \omega$ Shear Stress Transport turbulence models and compared to `Tranair`'s results.

A.2 Additional flight points

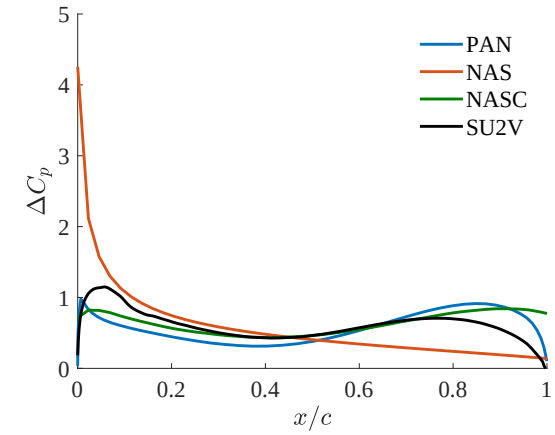
This section presents the results for the Embraer benchmark wing at several cruise speeds.

A.2.1 Low-speed cruise

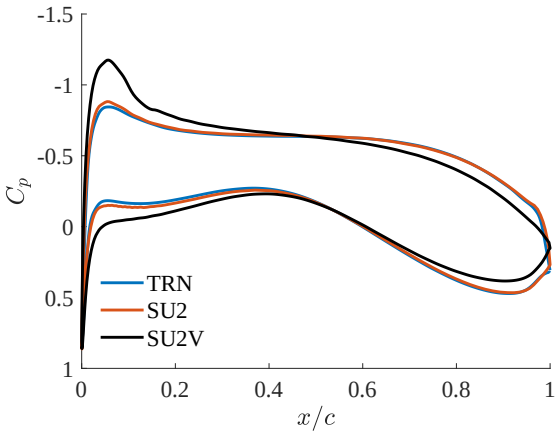
The wing has been simulated at Mach number $M = 0.70$, a lift coefficient $C_L = 0.58$, and altitude of $36\,000$ft.

Model	α	C_L	C_D	C_M
PAN	-0.2	0.58	0.0128	-0.817
NAS	$+5.5$	0.58	-	-0.705
NASC	-0.2	0.58	-	-0.797
TRN	-0.4	0.58	0.0141	-0.810
SU2	-0.2	0.58	0.0162	-0.810
TRNV	$+1.0$	0.58	0.0230	-0.764
SU2V	$+1.2$	0.58	0.0247	-0.765

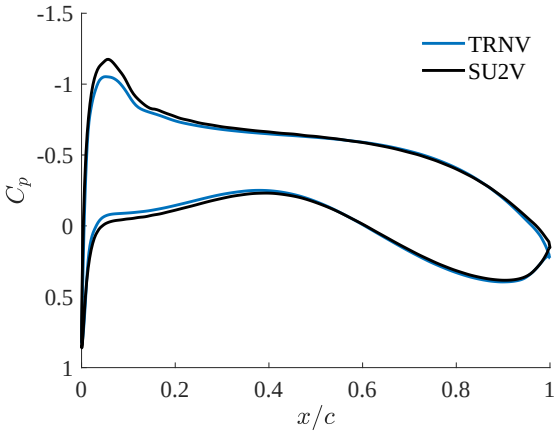
Table A.2.1: Aerodynamic coefficients obtained from different levels of fidelity for the Embraer benchmark wing at $M = 0.70$ and $C_L = 0.58$.



(a) Linear models

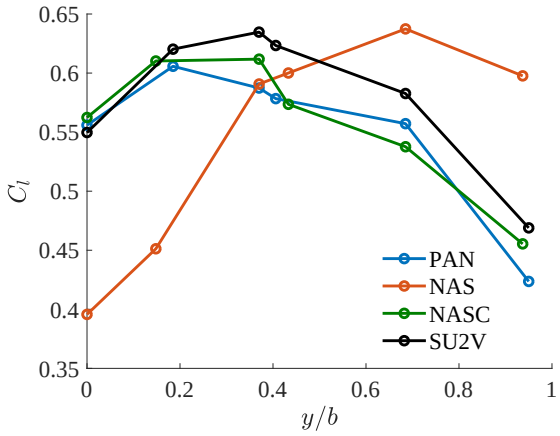


(b) Inviscid models

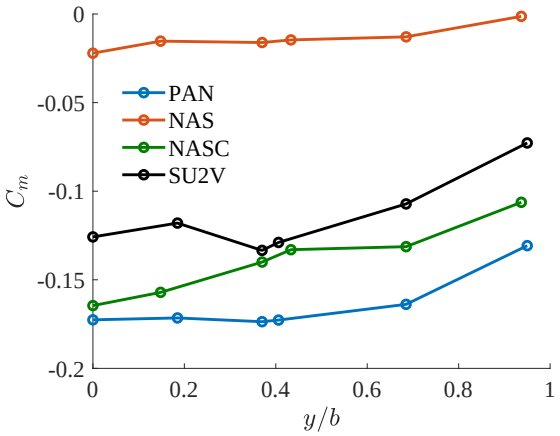


(c) Viscous models

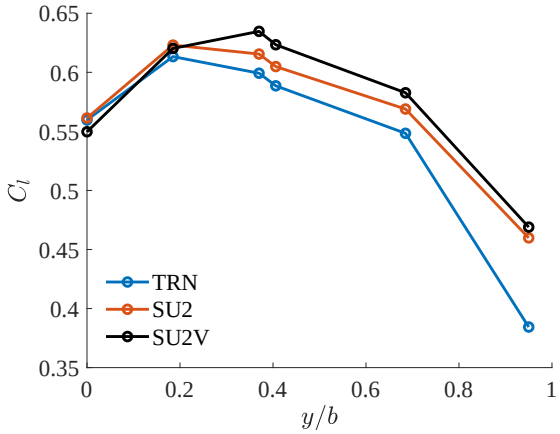
Figure A.2.1: Pressure distribution along the mean aerodynamic chord of the Embraer benchmark wing at $M = 0.70$ and $C_L = 0.58$ obtained from different levels of fidelity.



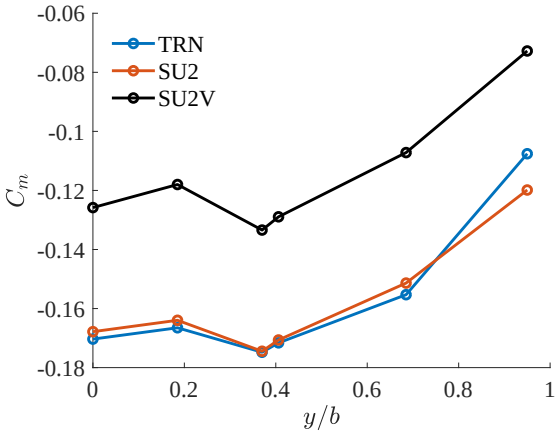
(a) Linear models



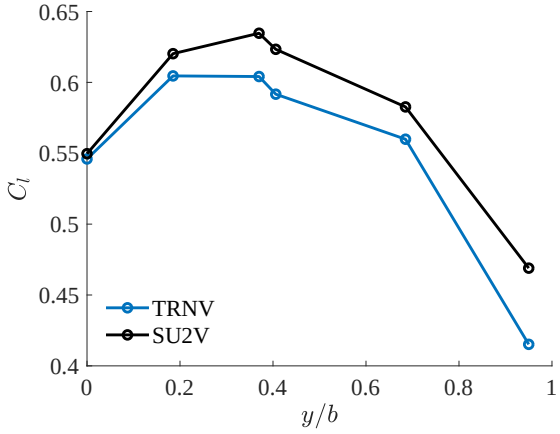
(b) Linear models



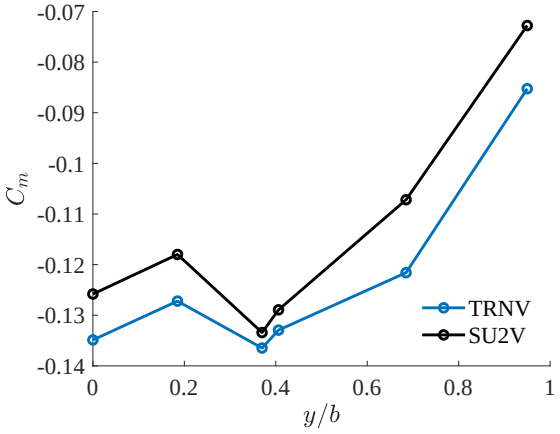
(c) Inviscid models



(d) Inviscid models



(e) Viscous models



(f) Viscous models

Figure A.2.2: Sectional aerodynamic loads along the span of the Embraer benchmark wing at $M = 0.70$ and $C_L = 0.58$ obtained from different levels of fidelity.

Model	n. cells	n. threads	wall-clock time	cpu time
PAN	$1\,000$	1	10 s	10 s
NAS	500	1	20 s	20 s
NASC	500	1	20 s	20 s
TRN	$500\,000$	1	4 min	4 min
SU2	$1\,300\,000$	12	10 min	2 h
TRNV	$500\,000$	1	10 min	10 min
SU2V	$1\,500\,000$	36	24 h	36 d

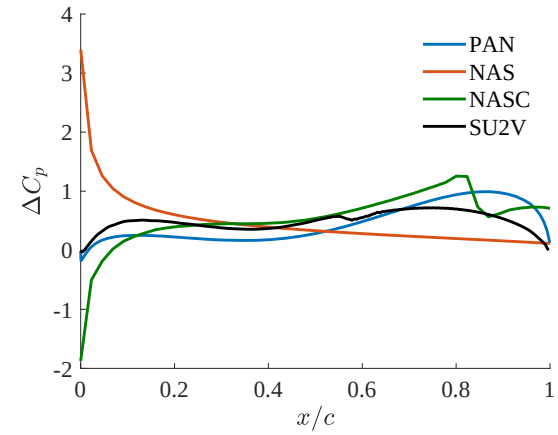
Table A.2.2: Mesh size and computational time required by the different models for the Embraer benchmark case at $M = 0.70$ and $C_L = 0.58$.

A.2.2 Nominal cruise

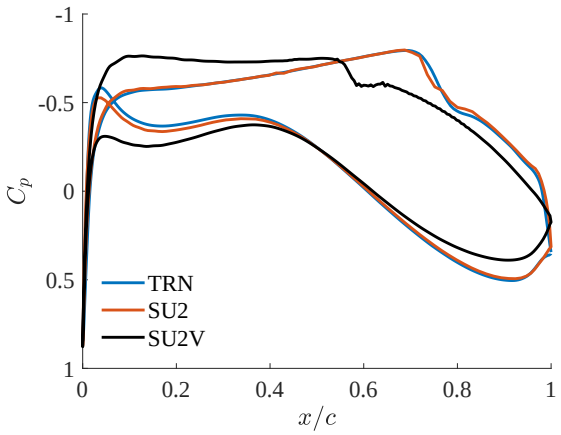
The wing has been simulated at Mach number $M = 0.78$, a lift coefficient $C_L = 0.47$, and altitude of $36\,000$ft.

Model	α	C_L	C_D	C_M
PAN	-1.6	0.47	0.0090	-0.706
NAS	$+4.1$	0.47	-	-0.576
NASC	-2.3	0.47	-	-0.708
TRN	-1.8	0.47	0.0105	-0.703
SU2	-1.7	0.47	0.0125	-0.701
TRNV	-0.7	0.47	0.0184	-0.656
SU2V	-0.3	0.47	0.0209	-0.648

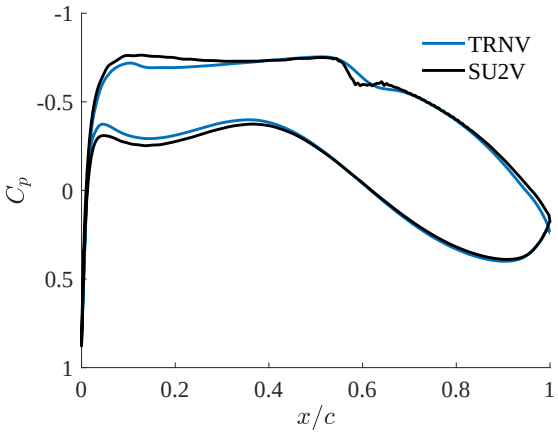
Table A.2.3: Aerodynamic coefficients obtained from different levels of fidelity for the Embraer benchmark wing at $M = 0.78$ and $C_L = 0.47$.



(a) Linear models

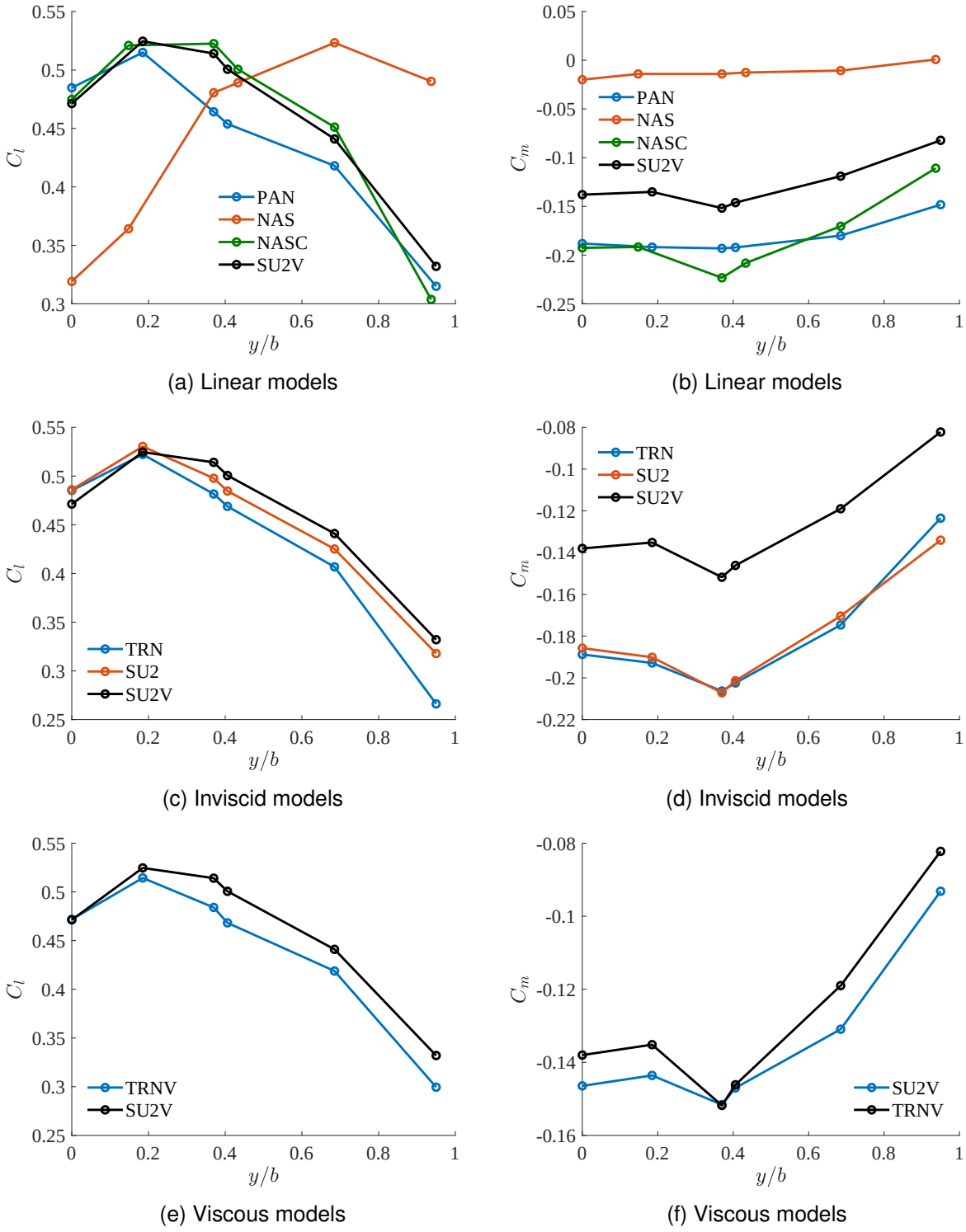


(b) Inviscid models



(c) Viscous models

Figure A.2.3: Pressure distribution along the mean aerodynamic chord of the Embraer benchmark wing at $M = 0.78$ and $C_L = 0.47$ obtained from different levels of fidelity.



(a) Linear models

(b) Linear models

(c) Inviscid models

(d) Inviscid models

(e) Viscous models

(f) Viscous models

Figure A.2.4: Sectional aerodynamic loads along the span of the Embraer benchmark wing at $M = 0.78$ and $C_L = 0.47$ obtained from different levels of fidelity.

Model	n. cells	n. threads	wall-clock time	cpu time
PAN	$1\,000$	1	10 s	10 s
NAS	500	1	20 s	20 s
NASC	500	1	20 s	20 s
TRN	$500\,000$	1	5 min	5 min
SU2	$1\,300\,000$	12	15 min	3 h
TRNV	$500\,000$	1	11 min	11 min
SU2V	$1\,500\,000$	36	24 h	36 d

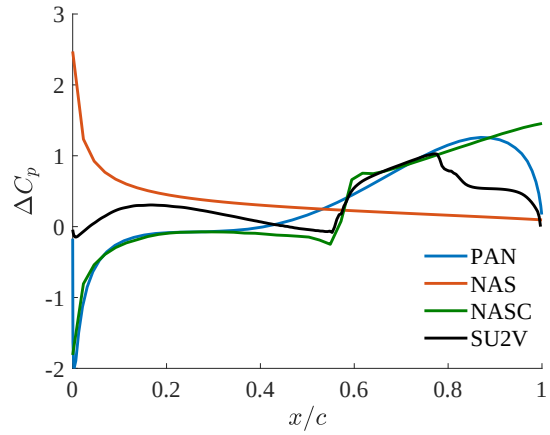
Table A.2.4: Mesh size and computational time required by the different models for the Embraer benchmark case at $M = 0.78$ and $C_L = 0.47$.

A.2.3 High-speed cruise

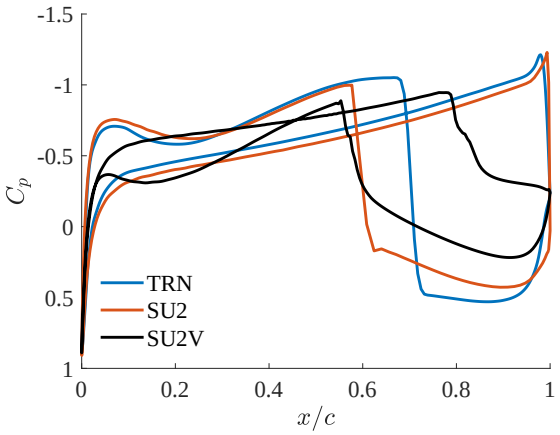
The wing has been simulated at Mach number $M = 0.89$, a lift coefficient $C_L = 0.36$, and altitude of $36\,000$ft.

Model	α	C_L	C_D	C_M
PAN	-3.1	0.36	0.0059	-0.626
NAS	$+2.7$	0.36	-	-0.448
NASC	$+2.1$	0.36	-	-0.676
TRN	-1.5	0.36	0.0680	-0.663
SU2	-2.2	0.36	0.0589	-0.673
TRNV	$+1.5$	0.36	0.0684	-0.491
SU2V	$+1.0$	0.36	0.0650	-0.521

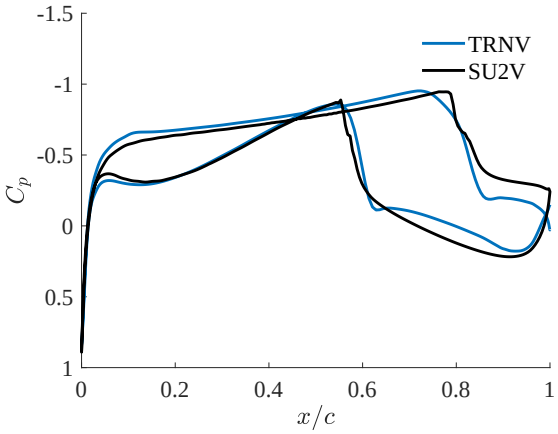
Table A.2.5: Aerodynamic coefficients obtained from different levels of fidelity for the Embraer benchmark wing at $M = 0.89$ and $C_L 0.36$.



(a) Linear models

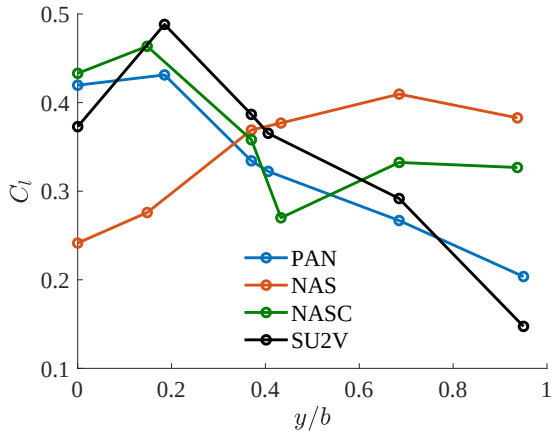


(b) Inviscid models

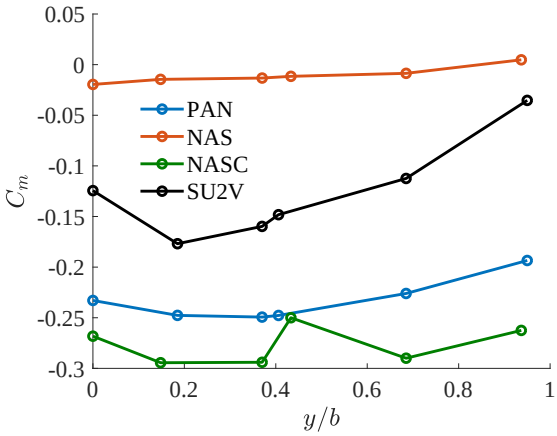


(c) Viscous models

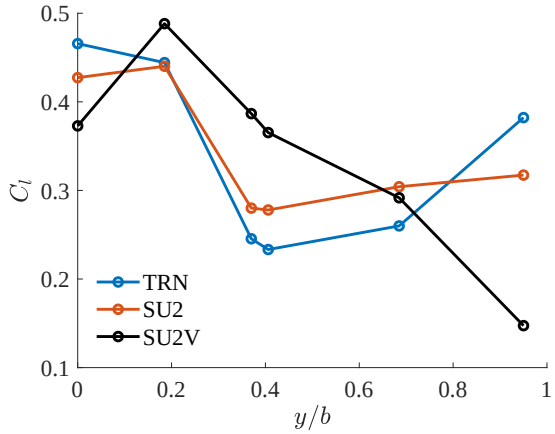
Figure A.2.5: Pressure distribution along the mean aerodynamic chord of the Embraer benchmark wing at $M = 0.89$ and $C_L 0.36$ obtained from different levels of fidelity.



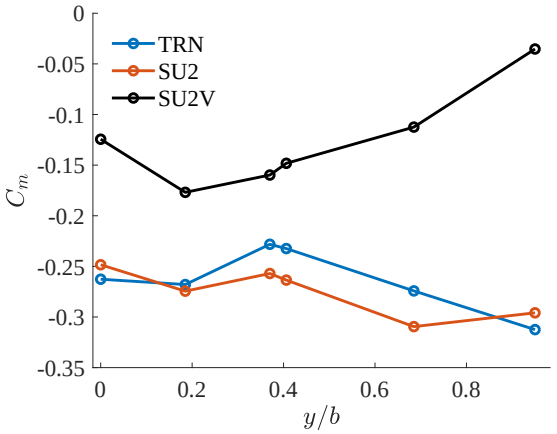
(a) Linear models



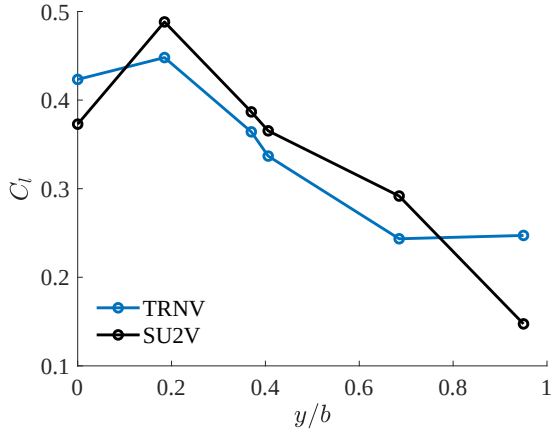
(b) Linear models



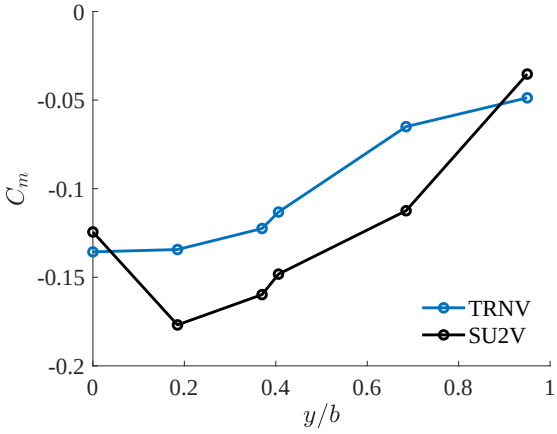
(c) Inviscid models



(d) Inviscid models



(e) Viscous models



(f) Viscous models

Figure A.2.6: Sectional aerodynamic loads along the span of the Embraer benchmark wing at $M = 0.89$ and $C_L 0.36$ obtained from different levels of fidelity.

Model	n. cells	n. threads	wall-clock time	cpu time
PAN	$1\,000$	1	10 s	10 s
NAS	500	1	20 s	20 s
NASC	500	1	20 s	20 s
TRN	$500\,000$	1	11 min	11 min
SU2	$1\,300\,000$	12	25 min	5 h
TRNV	$500\,000$	1	3 h	3 h
SU2V	$1\,500\,000$	36	48 h	72 d

Table A.2.6: Mesh size and computational time required by the different models for the Embraer benchmark case at $M = 0.89$ and $C_L 0.36$.

Appendix B

`Aero` - field panel code

This appendix gives more details about the implementation of the field panel method presented in chapter 4.

B.1 Organization of the code

The code is split into three main blocks: a pre-processor, a solver and a post-processor. The pre-processor reads the user provided data, such as the grid points and the flow conditions, and stores them into matrices. These matrices are further grouped into structures and passed to the solver. The solver first assembles the aerodynamic influence coefficients matrices and iteratively calls the panel method and the field module to solve the flow. When convergence has been reached, the solver computes the flow variables on the wing surface. Finally, the post-processor outputs the computed flow variables into readable ASCII files.

Figure B.1.1 illustrates the main blocks of the code and the different functions. They will be further described in the next sections.

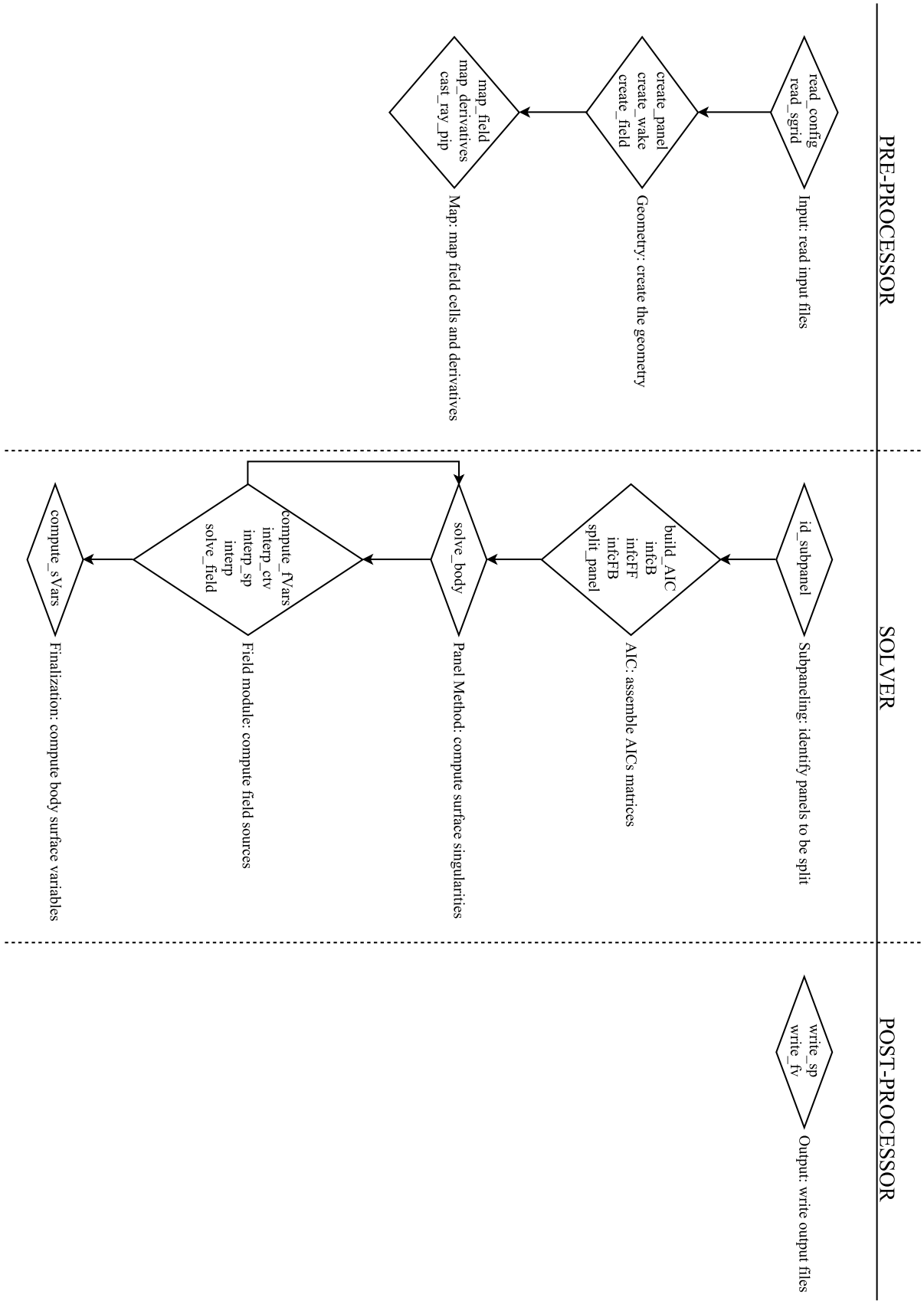


Figure B.1.1: Code flowchart.

B.1.1 Input and output files

The code relies on two ASCII input files. The first is a configuration (.CFG) file containing the freestream flow variables, *i.e.* Mach number and angle of attack, as well as the domain and the Cartesian grid definition parameters. The second file (.PTS) contains the points defining the geometry to be analyzed. These points need to be provided in the Selig format. The geometry is divided into stations along the span, and the points defining these stations are written in counter-clockwise order, starting from the trailing edge. A MATLAB code is provided to easily generate a grid in the required format for any wing shape consisting at most of nine planforms.

The results produced by the field panel method are written in two ASCII formatted .DAT files. The first file contains the coordinates of the center of all body panels as well as the corresponding pressure coefficient. The second file contains the coordinates of the center of all field cells as well as the corresponding field variables. Additionally, the code generates two ASCII .POS files, containing the Mach number in the field and the pressure coefficient on the wing surface. These files can be viewed and post-processed using `gmsh` [57].

B.1.2 Structures

The structures used in the code belong to two categories: the first is used to store the data and the solution on geometric entities, and the second is used to store the aerodynamic influence coefficients.

Geometry:

- *Network*: contains the geometry and the surface singularities, velocity and pressure of each surface panel
- *Field*: contains the geometry and the field singularities, velocity and density and Mach number of each field panel
- *Subpanel*: contains the geometry and the surface singularities of each sub-panel

Aerodynamic influence coefficient:

- *Body_AIC*: contains the body-to-body and body-to-field coefficients
- *Field2field_AIC*: contains the field-to-field coefficients
- *Field2body_AIC*: contains the field-to-body coefficients
- *Subpanel_AIC*: contains the sub-panel-to-field coefficients

B.1.3 Functions

As described in Figure B.1.1, the code is split into three main blocks, each composed of functions.

Pre-processor:

- *read_config*: reads the user-provided data file containing the freestream flow variables, the domain parameters, etc.
- *read_sgrid*: reads the user-provided file containing the surface grid points
- *create_panel*: creates the surface panels from the points and stores the data into matrices
- *create_wake*: creates the wake panels and stores the data into matrices
- *create_field*: creates the field panels from the domain and Cartesian grid parameters and stores the data into matrices
- *map_field*: identifies which field panels lie inside or outside the body
- *map_derivatives*: for each exterior field cell, identifies in which direction a finite difference may be performed, without crossing the wake or the body surface
- *cast_ray_pip*: ray casting and three-dimensional point-in-polyhedron algorithm, used by the previous two functions

Solver:

- *id_subpanel*: identifies surface panels lying too close to field cells and marks them
- *build_AIC*: builds the aerodynamic coefficient matrices and stores them into structures
- *infcB*: computes the body-to-body and body-to-field influence coefficients of a pair of panels
- *infcFF*: computes the field-to-field influence coefficients of a pair of field panels
- *infcFB*: computes the field-to-body influence coefficients of a pair of panels
- *split_panel*: splits the surface panels previously identified into sub-panel and computes the new body-to-field influence coefficients
- *solve_body*: computes the surface sources singularities and solves for the doublet singularities
- *compute_fVars*: computes the flow variables (density and Mach number) in the field
- *interp_ctv*: interpolates the surface singularities from panel centers to panel vertices

- *interp_sp*: interpolates the surface singularities from panel vertices to sub-panel centers
- *interp*: bilinear interpolation algorithm, used by the previous two functions
- *solve_field*: computes the field sources and the upwind bias
- *compute_sVars*: computes the flow variables (velocity and pressure coefficient) on the body surface

Post-processor:

- *write_sp*: writes the surface pressure coefficient in .DAT and .POS files
- *write_fv*: writes the field Mach number in .DAT and .POS files

B.2 Minigrid

In order to compute the derivatives in the field, the simplest approach is to use regular finite differences in the Cartesian grid. However, to increase the accuracy and to prevent the derivatives from passing through the surface of the body or the wake, a minigrid technique, proposed by Gebhardt et al. [152], can be used.

The minigrid technique consists in calculating the potential at several points inside a cell and computing the derivatives from these points. In practice, the size of the minigrid is small compared to the cell ($\Delta x_{\mathrm{MG}} = 0.001 \times \Delta x$) so that the derivatives do not intersect the body or wake surfaces. Using a minigrid with such small size yield identical results to computing the velocity directly the analytic aerodynamic influence coefficients rather than using finite differences on a regular grid. Figure B.2.1 depicts the two types of grid.

(a) Regular grid. (b) Minigrid.

Figure B.2.1: Grid types to compute derivatives.

On a minigrid (Figure B.2.1b), a typical derivative is computed as

$$\frac{\partial \varphi_{\mathrm{f}}}{\partial x}\Big|_{i,j} \simeq \frac{\varphi_{\mathrm{f}}|_{i+\Delta x_{\mathrm{MG}},j} - \varphi_{\mathrm{f}}|_{i-\Delta x_{\mathrm{MG}},j}}{2\Delta x_{\mathrm{MG}}}, \tag{B.2.1}$$

while on a regular grid (Figure B.2.1a), it would be computed as

$$\frac{\partial \varphi_{\mathrm{f}}}{\partial x}|_{i,j} \simeq \frac{\varphi_{\mathrm{f}}|_{i+1,j} - \varphi_{\mathrm{f}}|_{i-1,j}}{2\Delta x}. \tag{B.2.2}$$

Despite its advantages, the minigrid has two major drawbacks. The first is that the potential has to be computed at several points inside the cell, hence requiring more than one influence coefficient per cell. In three dimensions, seven coefficients are required instead of one. The second drawback is that the oscillations in the velocity close to the body caused by the discontinuity between the surface singularities are amplified by the minigrid. To completely remove these oscillations and ensure stability of the iterative process, a large number of sub-panels needs to be used, leading to prohibitive computational cost. On the other hand, a regular grid is coarser than a minigrid and tends to smooth the derivatives. Moreover, the accuracy of the results obtained on a coarse minigrid and a fine regular grid, both chosen to have the same computational cost, is almost the same. To avoid the oscillations in the solution and to simplify the implementation, the regular grid approach has been retained in the present work.

Appendix C

`Flow` - finite element code

This appendix gives more details about the implementation of the finite element code presented in chapter 5. A wiki is available at `https://gitlab.uliege.be/am-dept/waves/wikis/flow`, and documentation specific to developers can be generated using `doxygen` [212].

C.1 Code architecture

Contrary to the field panel method presented in chapter 4 and appendix B, `Flow` heavily relies on the modularity provided by the object-oriented programming. The code is split into classes, each having a specific responsibility. The scientific parts of the code needing computing and storage efficiency, such as the data structure manager and the solver, are written in C++. Using `SWIG` [173], these classes are derived in Python, which provides an intuitive and easy-to-use interface. The end-product is a highly flexible and modular, but fast and efficient, code.

Figure C.1.1 illustrates the global arrangement of the high-level C++ classes and an example use in Python.

Python input file

```
# modules
import tbox
import flow

# freestream flow
a = 1*pi/180 # angle of attack
b = 0*pi/180 # angle of sideslip
m = 0.8 # mach number

# mesh
msh = tbox.MeshData("geofile") # load mesh from geometry
mshCrck = tbox.MshCrack(msh) # create wake sheet
mshCrck.setCrack('wake')
mshCrck.run()

# problem
pbl = flow.Problem(msh, a, b, m)
pbl.set(flow.Medium(msh, 'field', m, a, b)) # set constitutive laws
pbl.add(flow.Initial(msh, 'field', a, b)) # add initial condition
pbl.add(flow.Dirichlet(msh, 'upstream', a, b)) # add Dirichlet BC
pbl.add(flow.Freestream(msh, 'farfield', a, b)) # add Neumann BC
pbl.add(flow.Wake(msh, ['wake', 'wake_'])) # add wake BC
pbl.add(flow.Kutta(msh, ['te', 'te_'])) # add Kutta condition
pbl.add(flow.Boundary(msh, 'wing')) # mark wing boundary

# solver
lsol = tbox.Pardiso() # use Pardiso as linear solver
sol = flow.Newton(pbl, lsol) # define Newton solver
sol.maxIt = 25 # set numerical parameters
sol.relRes = 1e-6
sol.run() # run and save results
sol.save()
```

C++ classes

Figure C.1.1: Code architecture.

C.1.1 Input and output files

`Flow` relies on two input files. The first (.GEO/.MSH) contains the mesh written in the native `gmsh` [57] format. If the geometry is created using `gmsh`, this geometry file can be directly provided to `Flow` instead. Furthermore, if the considered geometry is a wing, a suitable input file can be easily generated using `geoGen` [213]. The second file (.PY) is written in Python and defines the problem to be solved. As briefly illustrated on the left side in Figure C.1.1, this script file instantiates all the relevant objects and manipulates them. Alternatively, the use of Python also allows to easily automate the computation procedures commonly used in aircraft design. For example, a polar calculation script, allowing to sweep a given range of angles of attack, and a trim calculation script, allowing to automatically adjust the angle of attack to reach a prescribed lift coefficient, have already been written. The user can then simply provide a python dictionary containing the different parameters to configure the solver. Examples of geometry, script and configuration files are given in the code repository [171].

The data generated by `Flow` can be post-processed in several ways. Firstly, all the flow variables computed by the solver are readily available through Python. Secondly, these variables are written to disk either in the gmsh (.POS) or VTK [214] (.VTU) format, and can be visualized and post-processed using `gmsh` or `Paraview` [215]. If VTK and `Paraview` are used, sample python classes are also provided to automatically generate slices along the span of a wing. Thirdly, the data on the boundary of interest, such as wing and fuselage, are saved to ASCII .DAT files. These data can then be exported to virtually any post-processing tool.

C.1.2 C++ classes

The following list describes the main C++ classes used in `Flow`:

- *Problem*: holds the problem to be solved (flow parameters, boundary conditions, ...)
- *Medium*: holds the linear or nonlinear constitutive laws of the air
- *Initial*: prescribes an initial condition
- *Dirichlet*: prescribes a Dirichlet boundary condition
- *Freestream*: prescribes a Neumann boundary condition on freestream boundaries
- *Wake*: prescribes the Kutta condition on the wake sheets
- *Kutta*: prescribes the Kutta condition on the trailing edges
- *Newton*: solves the set of equations using a quasi-Newton method
- *Picard*: solves the set of equations using the Picard (fixed-point) method

Additionally, the following main C++ classes, part of the `waves` framework, are required:

- *MeshData*: holds the data structure of the grid
- *MeshCrack*: creates a crack in the mesh, used to insert a wake sheet with duplicated degrees of freedom
- *MeshDeformation*: deforms the mesh using linear elasticity laws
- *Pardiso*: interfaces Intel `Pardiso` direct linear (inner) solver
- *Mumps*: interfaces `MUMPS` direct linear (inner) solver
- *Gmres*: interfaces the GMRES iterative linear (inner) solver

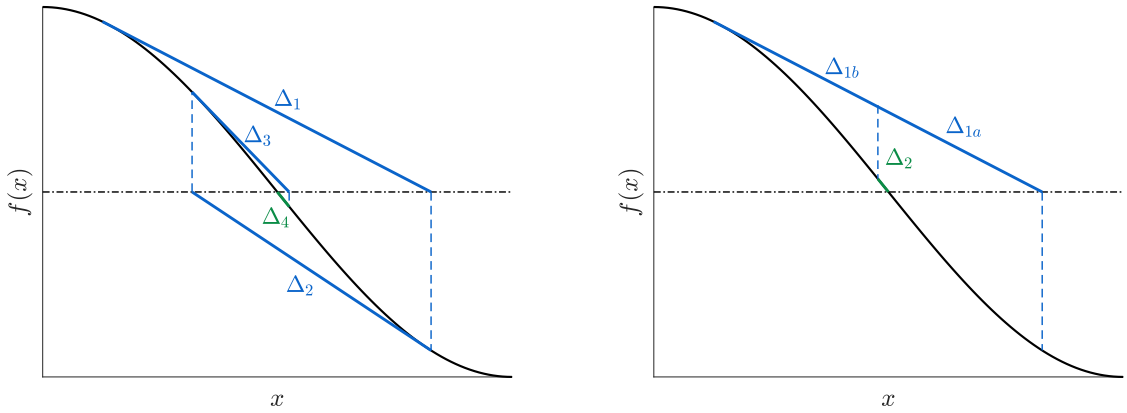
C.1.3 Python classes

Three python classes are currently available in `Flow`:

- *Config*: performs the basic configuration of the problem and solver
- *Polar*: sweeps a given range of angles of attack
- *Trim*: adjusts the angle of attack to reach a prescribed lift coefficient

C.2 Line search algorithms

A line search algorithm allows to adapt the magnitude of the change in the solution computed by a Newton method. The process usually relies on function evaluations performed at several intermediate solution points. The solution giving the lowest residual is retained as the new solution. A sample process is illustrated in Figure C.2.1. In Figure C.2.1a, a pure Newton method without line search is used, and four linear system solves alongside four function evaluations are required to find the root of the function. In Figure C.2.1b, the line search allows to find the root using two system solves and three function evaluations. In the present implementation, two line search methods have been implemented in order to increase the robustness of the Newton algorithm used to solve the nonlinear potential equation: the quadratic line search and the Bank and Rose line search.



(a) Newton method without line search.

(b) Newton method with line search.

Figure C.2.1: Effect of a line search technique on the convergence of a Newton method.

C.2.1 Quadratic line search

The quadratic line search algorithms approximates the equation for which a root is sought as a quadratic function. A bracketing phase is first performed to restrict the quadratic approximation range. Then, several function evaluations are performed with the goal to minimize the residual. As such, the procedure requires at least four function evaluations for each linear solution update. The block diagram of the line search is given in Figure C.2.2 and more details about the method can be found in chapter 2.4.1, page 93 of the book by Sun and Yuan [178].

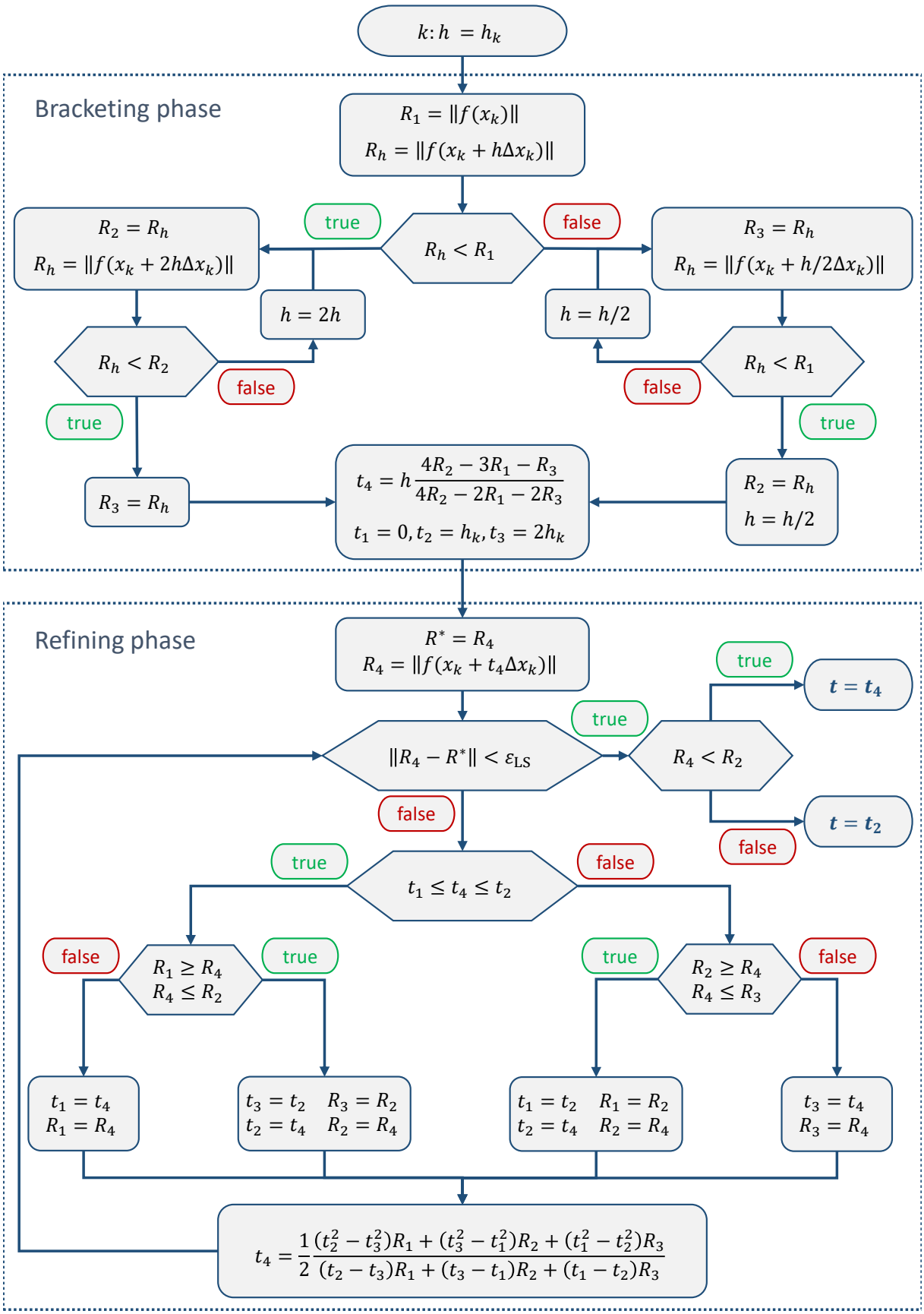


Figure C.2.2: Quadratic line search block diagram.

C.2.2 Bank and Rose line search

The line search developed by Bank and Rose [47] was initially implemented for semi-conductor physics modeling. In this procedure, the step length is computed as a function of the current residual vector and is iteratively decreased if needed. The damping factor K is then retained for the next Newton iteration. Two function evaluations are at least required by the algorithm. The block diagram of the line search wrapped into a Newton method is given in Figure C.2.3.

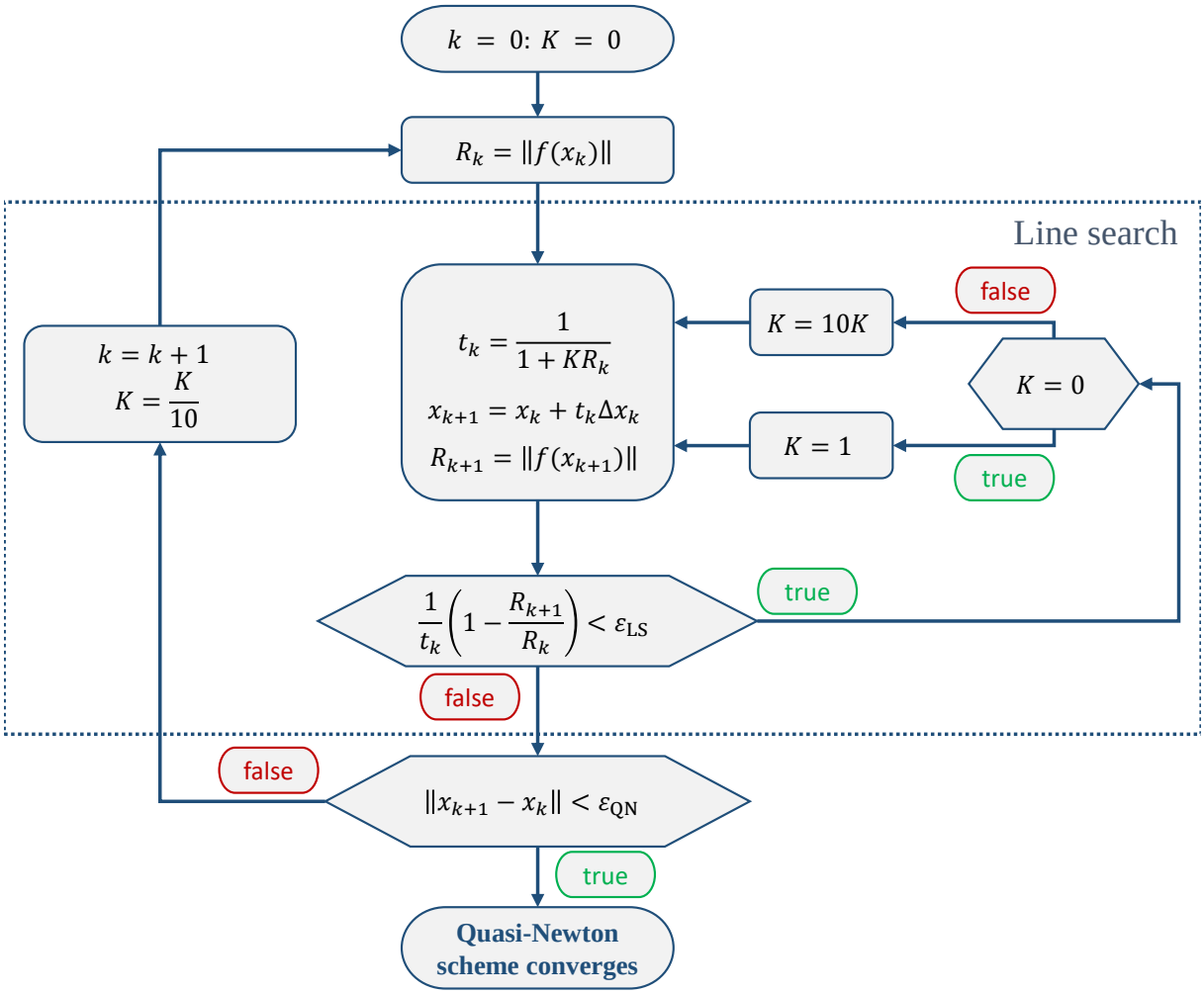


Figure C.2.3: Bank and Rose line search block diagram.

C.3 Integration

The wrapping generated using `SWIG` and used in `Waves` allows to easily integrate `Flow` into various environments that rely on Python. The relevant variables calculated by the solver are readily available in Python. More particularly, the geometry and the solution field, as well as the derived variables on the boundaries of interest, such as a wing, are hold by the *Boundary* class, which acts as an interface. These various data can then easily be accessed and manip-

ulated by external software. `Flow` has already been integrated within `CUPyDO` [193] to perform aeroelastic computations, as described in chapter 6. As another example, the code could be integrated in an optimizer, such as those provided by `scipy` [216] or even `pyOpt` [217].

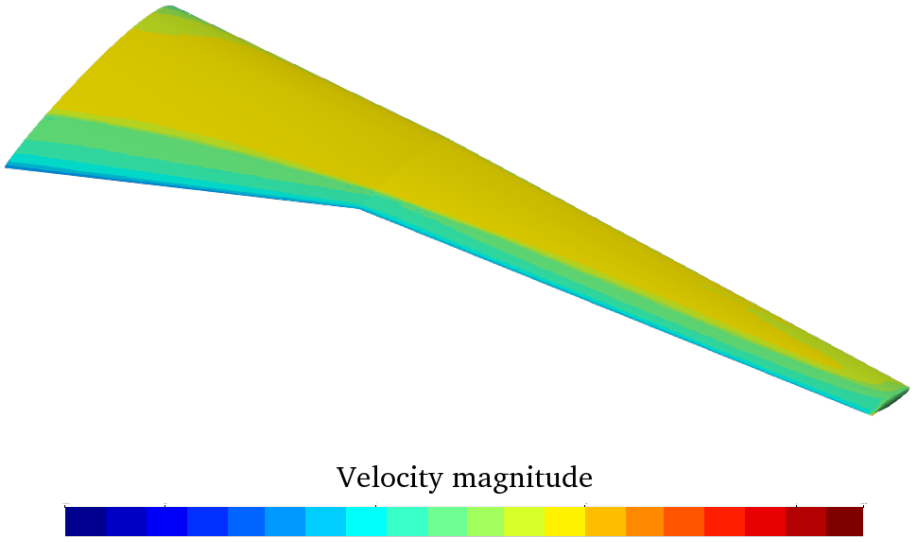
Appendix D

`Flow` sensitivity analysis

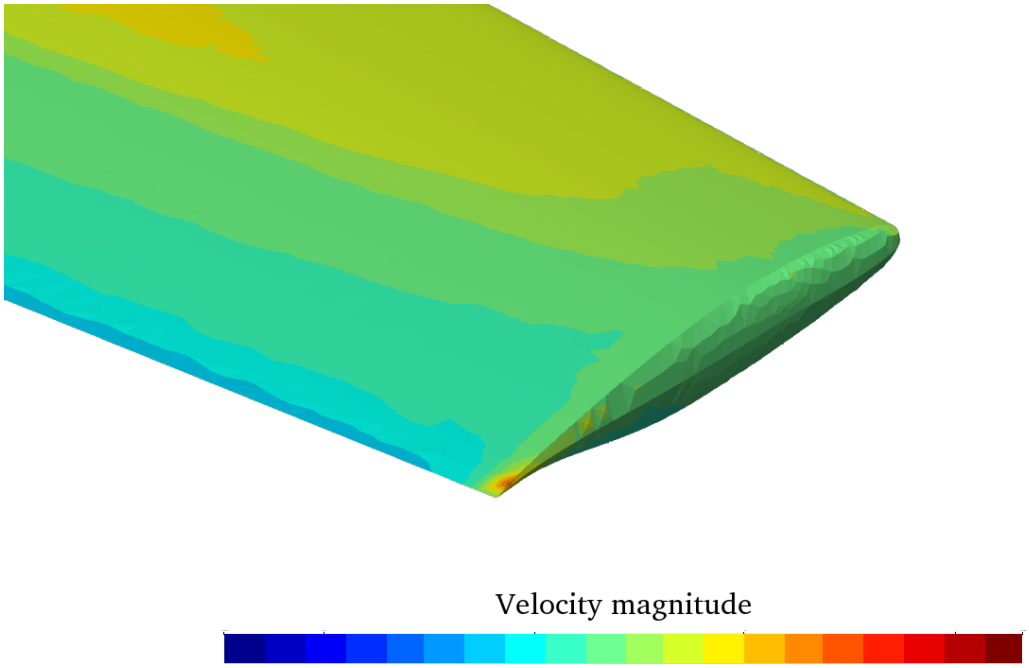
D.1 Tip vortex singularity

As explained in chapter 5, the vortex generated at the wingtip trailing edge of three-dimensional lifting configurations induces an infinite local velocity, which translates to a large local Mach number and a near vacuum density value at this location for high-speed freestream flows. The three-dimensional flow over the Embraer benchmark wing described in chapter 2 is computed at a high lift coefficient $C_L = 0.75$ and a high Mach number $M = 0.78$ to illustrate the solution behavior at the wingtip trailing edge. Figure D.1.1 shows the contour of the velocity magnitude with a focus on the wingtip trailing edge. Although the density is clamped using the Padé approximation described in chapter 5 and that the grid is kept coarse in the vicinity of the wingtip trailing edge, the velocity reaches very high values locally. The local Mach number is therefore also spuriously high. This may disrupt the convergence, or even cause the divergence of the method, although the solution is converged elsewhere in the domain.

Numerical experiments showed that the grid density should be decreased as much as possible near the wingtip trailing edge for high-speed and high-lift flows such as the one presented in Figure D.1.1. Although the cell size is usually set to $1/100$ of the local chord to get sufficient accuracy, good convergence characteristics can only be attained by using a cell size of $1/50$ to $1/30$ of the chord. As a result, the local behavior of the solution is slightly degraded. The implementation of techniques limiting the velocity near high flow gradients, or of a solution adaptive grid, is therefore highly desirable.



(a) Full wing.



(b) Wingtip region.

Figure D.1.1: Velocity magnitude surface contour of the Embraer benchmark wing at $C_L = 0.75$ and $M = 0.78$.

D.2 Linear solver

This section illustrates the computational performance of the different linear (inner) solver that can be used in `Flow`. The first solver is `Pardiso` [176], which is originally developed at the Institute of Computational Science in Switzerland, and shipped with the Intel *Math Kernel Li-*

brary. The second solver is MUMPS [177] developed by various French research groups and Universities. Both Pardiso and MUMPS are direct solvers. The third solver is the *Generalized Minimum RESidual* iterative algorithm, originally developed by Saad [114] and implemented in Eigen [164]. Since Pardiso and MUMPS are direct solvers, they do not require any configuration. On the other hand, GMRES is an iterative solver and different options need to be adjusted to obtain good performance. First, a tolerance below which the residual drops must be defined as a convergence criterion. Second, the restart parameter, specifying the number of iterations after which the algorithm is restarted, must be chosen. Third, GMRES is effective only if the set of linear equations is preconditioned. The preconditioning matrix is computed using an incomplete LU decomposition of the Jacobian matrix. Consequently, the fill-in factor, controlling how incomplete the preconditioner is, must be chosen as well. Additionally, the preconditioner can be computed such that the matrix entries below a given tolerance are dropped out. Numerical experiments showed that setting the number of restarts to 50 and using a drop tolerance of 10^{-6} for the preconditioner work usually well. The impact of the GMRES tolerance and the fill-in factor of the precondtioning matrix on the computational cost is quantified in Table D.2.1.

The three inner solvers are compared on the Onera M6 wing, simulated at an angle of attack $\alpha = 3.06°$ and a Mach number $M = 0.839$, and on the Embraer benchmark wing, simulated at a lift coefficient $C_L = 0.60$ and $M = 0.78$. Flow requires 15 nonlinear iterations to solve the Onera M6 case. The computational time needed using MUMPS is 280 s, while Pardiso requires 245 s. The Embraer benchmark case requires 16 iterations, and a computational time of 312 s using MUMPS, and of 290 s using Pardiso. The computational time required by the GMRES algorithm greatly depends on the tolerance and the fill-in factor, as illustrated in Table D.2.1. Note that using a zero fill-in factor, which is equivalent to not preconditioning the set of equations, is very cheap but prevents GMRES from converging. Similarly, using a tolerance higher than 10^{-3} prevents the Newton algorithm from converging. The results are therefore not included in the table. Several conclusions can be drawn. Firstly, GMRES is always faster than MUMPS, whatever the combination of tolerances and fill-in factors used. Secondly, GMRES can be faster than Pardiso if a favorable combination of tolerances and fill-in factors is used. Thirdly, for a given tolerance, the average number of inner iterations decreases as the fill-in factor increases, but the time required to compute the preconditioner increases. The fill-in factor for which the total computational time is the smallest depends on the tolerance and the case. Finally, the best performance are always achieved using a large tolerance and small fill-in factor. However, more benchmark cases need to be studied in order to confirm this behavior.

Tolerance / fill-in factor	1	2	3
10^{-3}	69 / 214 s	32 / 230 s	23 / 245 s
10^{-5}	129 / 240 s	55 / 235 s	35 / 249 s
10^{-8}	229 / 276 s	96 / 252 s	62 / 263 s

(a) Onera M6: $\alpha = 3.06°, M = 0.839$.

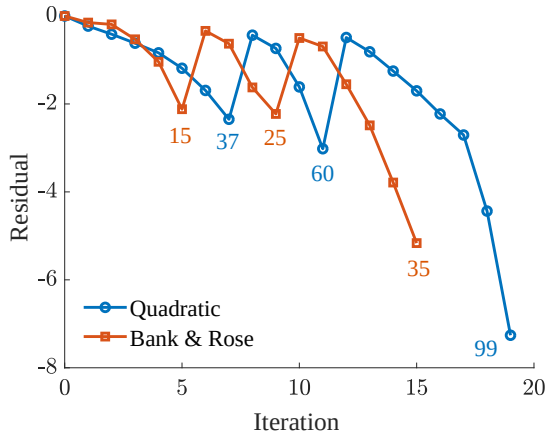
Tolerance / fill-in factor	1	2	3
10^{-3}	56 / 242 s	27 / 264 s	20 / 266 s
10^{-5}	107 / 270 s	49 / 274 s	38 / 269 s
10^{-8}	204 / 306 s	89 / 286 s	60 / 286 s

(b) Embraer benchmark wing: $C_L = 0.60, M = 0.78$.

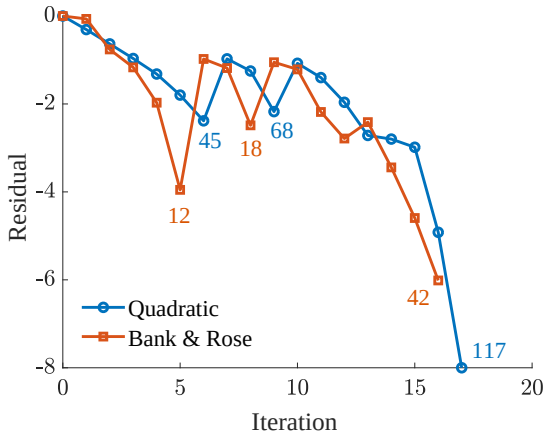
Table D.2.1: Number of inner iterations averaged by the number of nonlinear iterations and computational time required to solve the flow over the Onera M6 and the Embraer benchmark wings using GMRES, as a function of the tolerance and the fill-in factor of the preconditioner.

D.3 Line search

The present section illustrates the difference in convergence characteristics of the two line search algorithms described in appendix C that are available in `Flow`. Both methods are compared on the Onera wing, simulated at an angle of attack $\alpha = 3.06°$ and a Mach number $M = 0.839$, and on the Embraer benchmark wing, computed at a lift coefficient $C_L = 0.60$ and $M = 0.78$. The convergence plot, showing the evolution of the logarithm of the relative residual along the Newton iteration count, is given in Figure D.3.1. The accumulated number of function evaluations is also noted at each change in the upwinding parameters, *i.e.* when the residual drops below 10^{-2}. The Bank and Rose line search algorithm exhibits a better overall convergence rate than the quadratic line search for both test cases. More specifically, the number of Newton iterations is reduced by 4 and 1 for the Onera and the Embraer wing computation, respectively, and the total number of residual evaluations is reduced by roughly 65% in both cases. For the Onera computation, the computational time is reduced by 30%, while it is reduced by 25% in the Embraer case. In both cases, the final results differ by less than one count, in terms of aerodynamic coefficients.



(a) Onera M6: $\alpha = 3.06°, M = 0.839$.



(b) Embraer benchmark wing: $C_L = 0.60, M = 0.78$.

Figure D.3.1: Evolution of the relative residual as a function of the Newton iterations for the Onera and Embraer wings computations, with the accumulated number of function evaluations.