UNIVERSITY OF GENOA

Department of Mechanical, Energetics, Management and Transportation Engineering

Master's Thesis
in
Mechanical Engineering - Energy and Aeronautics

Surface waviness tolerance identification and transition delay of boundary layer flows over swept airfoils using an adjoint-based optimisation method

Supervisors:
Prof. Hanifi Ardeshir
Prof. Bottaro Alessandro

Co-Supervisor:
PhD. Moniripiri Mohammed

Candidate:
Cozzi Piero

Academic year 2022/2023

"Perfection is not attainable, but if we chase perfection we can catch excellence." - Vince Lombardi

Abstract

This research introduces an adjoint-based method for two types of studies aimed at maintaining laminar flow over swept aerodynamic surfaces: determining manufacturing tolerances subject to wavy excrescences and shape optimization to delay boundary layer transition. While focusing on a natural laminar flow airfoil, the methodology and algorithm are generalized to any case. The consideration of swept wings necessitated the implementation of a 2.5D transformation, with validation results provided. The growth of convective unstable disturbances is computed by solving Euler, boundary layer, and parabolized stability equations. The gradient of the kinetic energy of disturbances in the boundary layer with respect to surface grid points is calculated by solving adjoints of the governing equations. The utilization of sensitivity analysis information varies depending on the specific study under consideration.
For specific flight conditions, the steepest ascent method (GA) is employed to identify the waviness profile that induces a specific increase in the maximum value of the N-factor, ΔN. Numerical simulations utilize the NLF(2)-0415 airfoil across varying angles of attack, Reynolds numbers, and Mach numbers to examine their effects on computed tolerances. Angles of attack between $-1.00°$ and $1.75°$, Reynolds numbers between 9×10^6 and 15×10^6, and Mach numbers between 0.45 and 0.6 are considered for waviness profiles with different ranges of wavelengths.
In the transition delay study, the optimal airfoil profile shape is determined using the steepest descent method (GD). To parameterize the shape of the gradient and find the optimal perturbation of the surface, the projection onto the space of Hicks-Henne bump functions is proposed, along with some simple shape smoothing approaches. This section of the work focuses on the development of computational algorithms, presenting the codes. Results obtained are also discussed. The presented methodology demonstrates promising potential but further efforts are required to fully exploit its capabilities for the given problem.
For both studies, the use of the SLSQP optimization algorithm is also proposed to

perform constraint optimization with minimum L_2-norm of geometry deformation; however, results are not yet available.

Abstract

Questa ricerca introduce un metodo basato sulle equazioni aggiunte per due tipi di studi mirati a mantenere il flusso laminare su superfici aerodinamiche rastremate: determinazione delle tolleranze dimensionali di profili ondulati e del loro impatto sullo sviluppo dei disturbi, e ottimizzazione della forma della superficie al fine di ritardare la transizione dello strato limite. Pur concentrandosi su un profilo alare NLF, la metodologia e l'algoritmo sono generalizzabili a qualsiasi casistica. La considerazione di ali rastremate ha reso necessaria l'implementazione della trasformazione 2.5D, i cui risultati di validazione sono forniti. La crescita delle perturbazioni convettive instabili è calcolata risolvendo le equazioni di Eulero, dello strato limite (BLE) e di stabilità (PSE). Il gradiente dell'energia cinetica delle perturbazioni nello strato limite, calcolato rispetto ai punti di discretizzazione della superficie del profilo, è calcolato risolvendo le equazioni aggiunte. In base allo studio considerato, le informazioni provenienti dall'analisi di sensitività vengono impiegate di conseguenza.
Utilizzando il metodo della salita più ripida, cioè del gradiente (GA), viene identificato il profilo ondulato che provoca un aumento specifico del valore massimo del N-factor, ΔN. Le simulazioni numeriche utilizzano il profilo alare NLF(2)-0415 considerando angoli di attacco ($\in [-1.00°, 1.75°]$), numeri di Reynolds ($\in [9 \times 10^6, 15 \times 10^6]$) e numeri di Mach ($\in [0.45, 0.6]$) variabili; lo scopo è quello di esaminare, seguendo un approccio parametrico, l'effetto di ciascuno di essi nella determinazione delle tolleranze dimensionali. Inoltre, vengono presi in considerazione diversi valori della lunghezza d'onda delle ondulazioni della superficie.
Nello studio del ritardo nella transizione dello strato limite, si utilizza il metodo della discesa più rapida del gradiente (GD) per determinare la forma ottimale del profilo alare. Si propone di sfruttare le informazioni fornite dal gradiente calcolando le ampiezze delle perturbazioni geometriche come proiezione dello stesso nello spazio delle funzioni Hicks-Henne. La sezione dedicata a questo studio si concentra sullo sviluppo dell'algoritmo computazionale, presentando la metodologia e commentando le sezioni di codice sviluppate. Solamente alcuni risultati preliminari vengono presentati.

Il metodo implementato dimostra un potenziale promettente, ma ulteriori sforzi sono necessari per sfruttarne appieno le capacità nel contesto specifico del problema.
Per entrambi i problemi viene inoltre proposto l'utilizzo dell'algoritmo di ottimizzazione SLSQP per derivare la superficie di interesse, minimizzando nel contempo la L_2-norm; i risultati relativi a questo metodo non sono ancora disponibili.

Table of contents

List of figures

List of tables

Listings

Nomenclature

Roman Symbols

E	Disturbance kinetic energy
h	Scale factor
J	Objective function
k	N. of Fourier modes
κ	Heat conductivity
M	Mach number
p	Pressure
$\mathbf{Q}$	Boundary layer solution state
$\mathcal{R}$	Specific gas constant
T	Temperature
U	Mean flow streamwise velocity component
u	Disturbance streamwise velocity component
V	Mean flow spanwise velocity component
v	Disturbance flow spanwise velocity component
W	Mean flow wall-normal velocity component
w	Disturbance flow wall-normal velocity component
$\mathbf{X}$	Vector of nodal coordinates

Greek Symbols

α	Streamwise wavenumber
β	Spanwise wavenumber
γ	Ratio of specific heats
Λ	Sweep angle
μ	Dynamic viscosity
ν	Kinematic viscosity
Ω	Domain
ω	Angular frequency
ρ	Density
σ	Spatial grow rate
Θ	Wave function

Superscripts

$*$	Dimensional quantities
H	Complex conjugate transpose

Subscripts

0	Reference condition
1	Streamwise coordinate
2	Spanwise coordinate
3	Wall-normal coordinate
e	Values at boundary layer edge
f	Final location
i	General index

∞	Free stream quantities

Other Symbols

$*$	Dimensional quantities

Acronyms / Abbreviations

ACARE	Advisory Council for Aviation Research and Innovation in Europe
AoA	Angle of attack
BLE	Boundary Layer Equations
CF	Cross-flow
C_p	Pressure coefficient
c_p	Specific heat at constant pressure
GA	Gradient ascent
GD	Gradient descent
H-H	Hicks-Henne
h_{tol}	Dimensional tolerance
L2-norm	Euclidean norm
NLF	Natural laminar flow
PSE	Parabolized Stability Equations
Re	Reynolds number
SFC	Specific fuel consumption
SLSQP	Sequential least squares programming
TS	Tollmien–Schlichting

Introduction

In the 21st century, the imperative to reduce aircraft-specific fuel consumption (SFC) is paramount, driven by economic, operational, and ecological considerations. A noteworthy target, set by the Advisory Council for Aviation Research and Innovation in Europe (ACARE), aims for a 75% reduction in CO_2 per passenger kilometer emissions by 2050 compared to 2000 levels, sparking a technological race among companies. While some strategies like enhanced propulsion efficiency and weight reduction through advanced materials are already in play, others, such as the open-rotor with contra-rotating propellers, remain on the drawing board.
Natural laminar flow (NLF) design emerges as a promising technology to soon reduce SFC for larger aircraft. The potential to mitigate drag lies in achieving the laminar flow regime, given that friction drag constitutes a significant portion (40–60%) of a high-subsonic airplane's total drag. The effectiveness of NLF technology could result in a 7–16% reduction in total drag, contingent on achieving a 40% laminarization on key surfaces. Meeting the demands for maintaining the NLF regime involves considerations ranging from limited sweep angles and favorable pressure gradients to airfoil nose-radius limitations and high-lift compatibility.
Recognizing the significance of employing NLF airfoils, it becomes imperative to comprehend the factors that promote boundary layer transition over them. This understanding allows for the identification and potential mitigation of adverse effects, or the establishment of a margin of tolerance away from critical conditions. An analysis of this kind can be developed through two different approaches: imposing constraints on the realization of well-known NLF airfoil profiles found in literature, or, starting from the state-of-the-art NLF profiles, by defining an appropriate methodology and leveraging information available from a comprehensive study of boundary layer stability to define new profile shapes capable of ensuring improved aerodynamic performance. Roughly speaking, these are the foundational concepts upon which this work has been developed and conducted. These ideas, which may seem obvious, open up various

scenarios considering the infinite number of variables involved in a problem similar to this. In the following more details are provided to allow the reader to better understand the focus of this work.

Regarding the initial point we discussed, the necessity for high surface quality standards, which encompass factors such as roughness, rivet protrusions, and waviness, cannot be overstated as these factors are always present in the real applications. Operationally, considerations such as in-flight contamination, anti-icing maintenance, turbulence levels, and noise also come into play. Throughout the 1980s and 1990s, various companies conducted experimental trials, including wind tunnel studies, in-flight examinations, and flight testing initiatives. These endeavors underscored the feasibility of achieving laminar flow, with some investigations even demonstrating success while wearing protective gloves. Despite technological advancements enabling manufacturers to deliver smoother surfaces for maintaining NLF, the presence of inevitable irregularities during assembly, such as waviness, gaps, and steps, can significantly affect transition locations, thereby impacting the performance of NLF designs. Manufacturers face the challenge of determining permissible dimensions for surface irregularities to prevent sudden transitions to turbulence. Extensive literature delves into criteria for acceptable irregularity dimensions during transition, with studies by Fage, Carmichael, Crouch, Perraud, Wie, and Malik providing both empirical relationships and numerical computations.

The work of Moniripiri *et al.*, [1], introduces a physics-based numerical framework designed to determine manufacturing tolerances for smooth waviness on NLF surfaces. The identification of critical waviness profiles, which significantly influence boundary layer transition, is essential for establishing these tolerances. Such waviness profiles can be identified through an optimization process employing gradient-based methods. The approach used is similar to the one outlined by Amoignon *et al.*, [2], to compute gradients of perturbation energy within the boundary layer concerning surface deformation, utilizing an adjoint-based method. The maximum allowable surface deformation, minimizing the L_2-norm, and inducing a specific increase in disturbance amplification, has been determined through unconstrained optimization using the gradient ascent (GA) method, as well as constrained optimization employing the sequential least squares programming (SLSQP) algorithm.

Although the described work is extensive and thorough, it has the limitation of considering parallel flow conditions, specifically flow over a rectangular wing without sweep angle. Since aeronautical applications typically involve solutions with non-rectangular wings, it was decided in collaboration with the author to incorporate the treatment

of sweep angle into the available codes, thus accounting for cross-flow conditions as well. At the end of this study, it will be possible to compare the impact of dimensional tolerances and therefore assess their importance based on the specific application under consideration.
In reference to the study concerning the search for new NLF geometries capable of delaying boundary layer transition, it was decided to implement an optimization algorithm that closely resembles the structure proposed by Moniripiri *et al.*, namely the approach based on the adjoint method defined by Amoignon *et al.*. This study can therefore be seen as the opposite application of what was done in [1]. In particular, the impact of dimensional tolerances can be considered the worst-case scenario condition, while the definition of a new promising geometry serves as the best-case scenario condition. This is because the aim is to consider the disturbance responsible for boundary layer transition and leverage the sensitivity analysis to find an optimized geometric shape that reduces disturbance development and preferably damps it to ensure greater boundary layer stability. The idea is to find the ideal NLF shape modification that dumps the disturbance by solving an unconstrained optimization problem with gradient descent (GD) method and a constrained problem using SLSQP algorithm. The implementation mentioned was carried out taking into account both disturbances typical of parallel flow conditions and those of cross-flow conditions.
The absence of publications regarding studies of this kind in the literature suggests that this study can be seen as a first attempt to shed light on the idea just presented. Obviously, moving into new territory, the final outcome is by no means guaranteed, and the unexpected challenges that the study may encounter are numerous.
Thus, to summarize, in this work two distinct studies are introduced, sharing a common approach in workflow setup and utilizing similar numerical codes. To ensure clarity and coherence for the reader, the thesis is divided into three parts.
The first part addresses relevant theoretical aspects in general. The second and third sections constitute the core of the work, focusing on describing the methodologies employed, the codes used and developed and presenting the results for both studies conducted and validation test of the algorithm used. In the third part, more emphasis is given to the results related to the tolerance study, as the transition delay study requires further investigation and only some preliminary results are currently available. Finally, conclusions are presented in the last section.

Part I

Section One

Chapter 1

Governing equations

In the realm of fluid dynamics, governing equations stand as the foundational framework for studying the behavior of fluids. These equations encapsulate the principles dictating the motion and interactions of fluid elements within a given system.
At the core of fluid dynamics lies the continuity equation, expressing the conservation of mass as fluid particles traverse space and time. Simultaneously, the momentum equation, derived from Newton's second law, unveils the intricate relationship between the forces acting on a fluid and the resulting acceleration. Complementing the momentum equation, the energy equation delves into the transfer and transformation of energy within a fluid system. Whether accounting for the internal energy of the fluid or incorporating external forces, this equation encapsulates the thermodynamic intricacies influencing fluid behavior.
In the context of incompressible flows, where density variations are negligible, the Navier-Stokes equations emerge as the cornerstone, amalgamating the principles of conservation of mass and momentum. Presented in both conservative and non-conservative forms, these equations lay the groundwork for simulating and analyzing a wide array of fluid phenomena.
In compressible flows, governing equations extend to include additional considerations, such as the conservation of energy and the equation of state, reflecting the impact of varying density and temperature on fluid behavior.
Given the focus of this thesis, it is also crucial to distinguish between two extremes in fluid analysis: viscous and inviscid fluid conditions. In inviscid flow, the assumption of complete absence of viscosity simplifies the equations of motion, facilitating a more straightforward analysis. Conversely, viscous flow considers the effects of viscosity, accounting for boundary layers and internal resistance.

1.1 Inviscid Region

1.1.1 Euler Equations

The Euler equations govern the flow of an inviscid compressible fluid, expressing the conservation of mass, momentum, and energy. In steady state, the integral form for any fixed region V with boundary ∂V is given by

$$\int_{\partial V} \mathbf{f} \cdot \hat{\mathbf{n}} \, dS = 0, \tag{1.1}$$

where, $\hat{\mathbf{n}}$ is the outward-oriented unit normal of the control volume V, and $\mathbf{f}$ is a matrix of tensors

$$\mathbf{f} = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u} \otimes \mathbf{u} + \mathbf{I}p \\ \mathbf{u}(E+p) \end{pmatrix}, \tag{1.2}$$

where E, the total energy per unit volume, is related to pressure p, density ρ, and velocity $\mathbf{u}$. Assuming the law of perfect gas for ideal fluids, E is given by

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho\mathbf{u}^2. \tag{1.3}$$

At the walls, the impermeability condition is expressed as

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0. \tag{1.4}$$

The fluid state, described in terms of conservative variables, is denoted by $\mathbf{w}$, where $\mathbf{m} = \rho\mathbf{u}$. Primitive variables are also employed in the subsequent derivations and are denoted by $\mathbf{v}$. Their definitions are

$$\mathbf{w} = \begin{pmatrix} \rho \\ \mathbf{m} \\ E \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} \rho \\ \mathbf{u} \\ p \end{pmatrix}. \tag{1.5}$$

To solve equations (1.1)–(1.4) for the flow around an airfoil, a finite sub-domain Ω is defined and artificial boundary conditions have to be specified and typically farfield boundary conditions are used. We utilize the ADflow solver to solve equations (1.1)–(1.4) along with boundary conditions at the farfield. ADfow is a finite-volume structured multiblock and overset mesh solver that is available under an open-source license and can solve the Euler, laminar Navier–Stokes, and Reynolds Averaged Navier Stokes (RANS) equations using a second-order-accurate finite-volume approach for the spatial discretization.

1.2 Viscous region

1.2.1 Boundary layer equations

The flow field under consideration in this study is the boundary layer on a swept wing with infinite span. The solution is obtained by solving the mass, momentum, and energy conservation equations for a viscous compressible fluid. These equations are expressed in an orthogonal curvilinear coordinate system with streamwise, spanwise, and wall-normal coordinates denoted as x_1, x_2, and x_3, respectively. A curvilinear length element is defined as $ds^2 = (h_1 dx_1)^2 + (h_2 dx_2)^2 + (h_3 dx_3)^2$, where h_i are the scale factors. In the context of an infinite swept wing flow problem, the scale factors h_1 and h_3 are set to unity. The total flow field, denoted by q_{tot}, is decomposed into mean ($\bar{q}$) and perturbation ($\tilde{q}$) parts, as

$$q_{\text{tot}}(x_1, x_2, x_3, t) = \bar{q}(x_1, x_3) + \tilde{q}(x_1, x_2, x_3, t) \tag{1.6}$$

where $\bar{q} \in [U, V, W, p, T, \rho]$ and $\tilde{q} \in [\tilde{u}, \tilde{v}, \tilde{w}, \tilde{p}, \tilde{T}, \tilde{\rho}]$. Here, U, V, and W represent the streamwise, spanwise, and wall-normal velocity components of the mean flow, respectively. T denotes temperature, ρ density, and p stands for pressure. The lowercase variables correspond to disturbance quantities. The equations are formulated for a quasi-three-dimensional mean flow with negligible variation in the spanwise direction. The evolution of convectively unstable disturbances is investigated within the framework of nonlocal stability theory.

All flow and material properties are made dimensionless with respect to reference quantities and in the following, dimensional quantities are denoted by the superscript $*$. To maintain the treatment as general as possible, let's denote the reference length scale as l_0^*, which can be defined a priori, for example, in terms of a fixed streamwise position x_0^* and the corresponding reference flow quantities at that location. Following the description provided above, it is possible to define the reference length scale as $l_0^* = \sqrt{\frac{\nu_0^* x_0^*}{U_0^*}}$. In the case of referring to the flow around an airfoil, it is possible and common to set the reference length as the chord dimension, i.e. $l_0^* = c$, and the free stream characteristics of the flow as reference flow quantities (usually indicated with

"$_\infty$" and "$^-$"). Regarding pressure, is made non-dimensional with twice the dynamic pressure.

The Reynolds, Mach and Prandtl numbers are then formulated as

$$\begin{aligned} Re &= \frac{l_0^* U_0^*}{\nu_0^*} \\ M &= \frac{U_0^*}{\sqrt{\mathcal{R}\gamma T_0^*}} \\ Pr &= \frac{c_p^* \mu_0^*}{\kappa^*} \end{aligned} \tag{1.7}$$

respectively. Here, $\mathcal{R}$ signifies the specific gas constant, ν represents the kinematic viscosity, μ dynamic viscosity, γ denotes the ratio of specific heats, c_p is the specific heat at constant pressure and κ is heat conductivity.

1.2.2 Mean flow equations

The dimensionless boundary-layer equations governing the steady viscous compressible mean flow on an infinite-span swept wing, expressed in primitive variable form, are presented as follows:

$$\frac{1}{h_1}\frac{\partial(\rho U)}{\partial x_1} + \frac{\partial(\rho W)}{\partial x_3} = 0, \tag{1.8}$$

$$\frac{\rho U}{h_1}\frac{\partial U}{\partial x_1} + \rho W \frac{\partial U}{\partial x_3} = -\frac{1}{h_1}\frac{dp_e}{dx_1} + \frac{1}{Re}\frac{\partial}{\partial x_3}\left(\mu \frac{\partial U}{\partial x_3}\right), \tag{1.9}$$

$$\frac{\rho U}{h_1}\frac{\partial V}{\partial x_1} + \rho W \frac{\partial V}{\partial x_3} = \frac{1}{Re}\frac{\partial}{\partial x_3}\left(\mu \frac{\partial V}{\partial x_3}\right), \tag{1.10}$$

$$\begin{aligned} c_p \frac{\rho U}{h_1}\frac{\partial T}{\partial x_1} + c_p \rho W \frac{\partial T}{\partial x_3} &= \frac{1}{RePr}\frac{\partial}{\partial x_3}\left(\kappa \frac{\partial T}{\partial x_3}\right) \\ &\quad + (\gamma - 1)\frac{UM^2}{h_1}\frac{dp_e}{dx_1} + (\gamma-1)\frac{\mu M^2}{Re}\left[\left(\frac{\partial U}{\partial x_3}\right)^2 + \left(\frac{\partial V}{\partial x_3}\right)^2\right]. \end{aligned} \tag{1.11}$$

According to the boundary layer assumptions, the pressure remains constant in the direction normal to the boundary layer, thus denoted as $p = p_e(x_1)$, considering the variables with subscript e are evaluated at the boundary layer edge. The equation of state can then be expressed as:

$$\gamma M^2 p_e = \rho T, \tag{1.12}$$

and the streamwise derivative of the pressure is given as

$$\frac{dp_e}{dx_1} = -\rho_e U_e \frac{dU_e}{dx_1}. \tag{1.13}$$

Denoting the dimensional free-stream velocity as Q_∞^* and the sweep angle as Λ, for a given pressure distribution described by the pressure coefficient:

$$C_p = \frac{p^* - p_\infty^*}{\frac{1}{2}\rho_\infty^* {Q_\infty^*}^2}, \tag{1.14}$$

the values at the boundary layer edge are determined as follows:

$$\begin{aligned} p_e &= \frac{p_e}{p_\infty}\frac{1}{\gamma M^2}, \qquad & T_e &= \left(\frac{p_e}{p_\infty}\right)^{\frac{\gamma-1}{\gamma}}, \\ \rho_e &= \left(\frac{p_e}{p_\infty}\right)^{\frac{1}{\gamma}}, \qquad & U_e &= \sqrt{Q_e^2 - V_e^2}, \end{aligned} \tag{1.15}$$

where

$$\frac{p_e}{p_\infty} = 1 + \frac{1}{2}C_p \gamma M^2, \quad Q_e^2 \ = 1 + \frac{1 - T_e c_{p\infty}}{\frac{(\gamma-1)}{2}M^2}, \quad V_e = Q_e \sin\Lambda. \tag{1.16}$$

Here, we utilize the assumptions of an inviscid, steady, and adiabatic flow, where the total enthalpy remains constant along a streamline, and the isentropic relations are applied to establish the relationship between total and static quantities. We define a domain Ω_B for equations (1.8)–(1.11) such that $x_1 \in [X_S, X_1]$, $x_2 \in [Z_0, Z_1]$, and $x_3 \in [0, \infty)$. The no-slip condition is enforced for the velocity components, while the adiabatic wall condition applies to the temperature. In the free stream, the streamwise and spanwise velocity components, as well as the temperature, assume their respective values at the boundary layer edge:

$$\begin{aligned} &\left[U, V, W, \frac{\partial T}{\partial x_3}\right](x_1, 0) = [0, 0, 0, 0], \quad \forall x_1 \in [X_S, X_1], \\ &\lim_{x_3 \to +\infty} [U, V, T](x_1, x_3) = [U_e, V_e, T_e](x_1), \quad \forall x_1 \in [X_S, X_1]. \end{aligned} \tag{1.17}$$

These nonlinear equations undergo an iterative solution process. Starting from equations (1.8) through (1.11), we obtain the solution $\tilde{\mathbf{Q}} = (U, V, T)$ by applying the boundary condition mentioned above for a given W value. Next, Equation (1.8) is integrated in the wall-normal direction to determine W. Convergence is achieved when the relative change in the wall-normal derivative of the streamwise velocity component at the wall falls below a predefined threshold. In the subsequent treatment, we adopt $\mathbf{Q} = (U, V, W, T)$ as the solution for the boundary layer state.

1.2.3 Disturbance equations

The perturbations are assumed to be a combination of time and spanwise periodic waves functions as

$$\tilde{\mathbf{q}}(x_1, x_2, x_3, t) = \hat{\mathbf{q}}(x_1, x_3)\Theta(x_1, x_2, t), \tag{1.18}$$

where the wave function Θ is defined as

$$\Theta(x_1, x_2, t) = \exp\left[i\left(\int_{X_0}^{x_1} \alpha(x')\, dx' + \beta x_2 - \omega t\right)\right]. \tag{1.19}$$

In equation (1.19), α represents the complex streamwise wavenumber, β denotes the real spanwise wavenumber, and ω stands for the real disturbance angular frequency. Disturbances are introduced into the mean flow at a streamwise position denoted as X_0, indicating that the disturbances begin to amplify from this location. Based on boundary layer approximations, it is possible to assume a scale separation of Re^{-1} between the weak variation in the x_1-direction as well as between the wall-normal and streamwise mean velocity components. Further, it is assumed that $\partial/\partial x_1 \sim O(Re^{-1})$ and $W \sim O(Re^{-1})$. As explained in the work of Bertolotti *et al.* [3], by incorporating the assumption outlined in equation (1.18) into the linearized governing equations and retaining terms up to order $O(Re^{-1})$, a set of nearly parabolic partial differential equations is derived. These equations, known as the Parabolized Stability Equations (PSE), are presented here in the compact version:

$$\mathcal{A}\hat{\mathbf{q}} + \mathbf{B}\frac{\partial \hat{\mathbf{q}}}{\partial x_3} + \mathcal{C}\frac{\partial^2 \hat{\mathbf{q}}}{(\partial x_3)^2} + \mathcal{D}\frac{1}{h_1}\frac{\partial \hat{\mathbf{q}}}{\partial x_1} = \mathbf{0}, \tag{1.20}$$

defining $\hat{\mathbf{q}} = [\hat{\rho}, \hat{u}, \hat{v}, \hat{w}, \hat{T}]^T$. The coefficients of the matrices $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ are not reported here in order to keep the theoretical treatment concise, but can be found in Pralits *et al.* work, [4].

Equation (1.20) is defined in a domain Ω_P such that $x_1 \in [X_0, X_1]$, $x_2 \in [Z_0, Z_1]$, and $x_3 \in [0, \infty)$, and its boundary conditions are given by

$$\begin{aligned} \left[\hat{u}, \hat{v}, \hat{w}, \hat{T}\right](x_1, 0) &= [0, 0, 0, 0], \quad \forall x_1 \in [X_0, X_1], \\ \lim_{x_3 \to +\infty} \left[\hat{u}, \hat{v}, \hat{w}, \hat{T}\right](x_1, x_3) &= [0, 0, 0, 0], \quad \forall x_1 \in [X_0, X_1]. \end{aligned} \tag{1.21}$$

To remove the ambiguity arising from the x_1-dependence of both the amplitude and wave function in the hypothesis introduced by equation (1.18), and to maintain a slow streamwise variation of the amplitude function $\hat{\mathbf{q}}$, the so called auxiliary condition reported below is introduced:

$$\int_0^{+\infty} \hat{\mathbf{q}}^H \frac{\partial \hat{\mathbf{q}}}{\partial x_1}\, dx_3 = 0. \tag{1.22}$$

In Equation (1.22), the superscript H denotes the complex conjugate transpose. Equation (1.20) is integrated downstream with an initial condition set at $x_1 = X_1$, given by local stability theory. At each streamwise location x_1, the streamwise wavenumber α is iterated until the auxiliary condition, namely equation (1.22), is satisfied. After a converged streamwise wavenumber has been obtained, the spatial growth rate, σ, based on the disturbance kinetic energy, E, can be calculated from the relation

$$\sigma = -\alpha_i + \frac{\partial}{\partial x_1}\left(\ln\left(\sqrt{E}\right)\right), \tag{1.23}$$

where α_i is the immaginary part of α, and energy is defined as:

$$E = \int_0^{+\infty} \bar{\rho}\left(|\hat{u}|^2 + |\hat{v}|^2 + |\hat{w}|^2\right) dx_3. \tag{1.24}$$

Finally, the growth rate, also known as the N-factor, which is a measure of integrated growth of perturbations and can be used to predict the transition location using the so-called $e^N - method$, can be calculated based on its kinetic energy as:

$$N_E = \int_{X_{n1}}^{X} \sigma\, dx_1, \tag{1.25}$$

where X_{n1} is the lower branch of the neutral curve, where disturbance amplification starts to grow, namely where $\sigma = 0$. A recent review about the relationship between the e^N-method and transition location is available in the work of Van Ingen [5].
If the set of equations mentioned above (a combination of Euler, BLE, and PSE equations) is properly implemented in a numerical scheme, it allows for a stability analysis of the boundary layer over a given wing section. In the works of Hanifi *et al.* [6] and Pralits *et al.* [4] this implementation has been done, and they can be used as references to find all the numerical schemes employed in this study, along with further detailed derivations.

Chapter 2

Objective function and sensitivity analysis

The choice of the objective function is closely tied to the nature of the study being conducted. One of the goals of this research aims to utilize gradient-based optimization techniques to determine the surface waviness profile that minimizes the L2-norm while inducing a specific increase in perturbation amplification. This increase could potentially trigger the premature transition of the boundary layer, leading to an undesirable rise in viscous drag. Such an effect would manifest as a reduction in the laminar region of the wing, effectively shifting the location of the laminar-turbulent transition further upstream. Therefore, selecting an objective function that captures these physical phenomena is paramount.
As already mentioned, this case study can be seen as the opposite of the best-case scenario, which is the second goal of this work. In this scenario, the aim is to search for the optimal wing profile configuration to delay the boundary layer transition. Although the ultimate goal of the two distinct studies is diametrically opposed, we can recognize a significant overlap in the theoretical definition of the optimization problems. In particular, it is observed that in both contexts, the objective function J is chosen to quantify the growth of perturbations within the domain.
In these kind of studies one common choice is to measure the kinetic energy of a certain disturbance at a downstream position, say X_f. This can be represented mathematically as:

$$E_f = \frac{1}{2} \int_{Z_0}^{Z_1} \int_0^{+\infty} \tilde{\mathbf{q}}^H M \tilde{\mathbf{q}} h_1 \, dx_2 \, dx_3 \bigg|_{x_1 = X_f} \tag{2.1}$$

where $M = \mathrm{diag}(0, 1, 1, 1, 0)$, which, following the definitions given above, implies that the disturbance kinetic energy is calculated from the disturbance velocity components. If the position X_f is chosen as the upper branch of the neutral curve, then the measure can be related to the maximum value of the N-factor of a given disturbance. Additionally, if the N-factor of the measured disturbance is the first to reach the transition N-factor, then the position can be associated with the onset of laminar-turbulent transition. However, it is not clear a priori that minimizing such a measure will dampen the chosen or other disturbances in the entire unstable region, especially if different types of disturbances are present, such as Tollmien-Schlichting (TS) and cross-flow (CF) waves. Considering that our study focuses on the flow over a swept wing, it is common for both TS and CF waves to be present simultaneously.
An alternative approach is to measure the kinetic energy as the streamwise integral over a defined domain. Using this method, several different disturbances can be considered, each with their respective maximum growth rate at different positions. The size of disturbances, denoted by K, superimposed on the mean flow at an upstream position X_0, can be measured by their total kinetic energy as:

$$E_K = \frac{1}{2} \sum_{k=1}^{K} \int_{X_{ms}}^{X_{me}} \int_{Z_0}^{Z_1} \int_0^{+\infty} \tilde{\mathbf{q}}_{\mathbf{k}}{}^H M \tilde{\mathbf{q}}_{\mathbf{k}} h_1 \, dx_1 \, dx_2 \, dx_3. \tag{2.2}$$

Here, X_{ms} and X_{me} represent the first and last streamwise positions between which the disturbance kinetic energy is integrated (where the perturbation exists), providing the opportunity to evaluate E_K within a streamwise domain within $[X_0, X_1]$.
The kinetic energy of a single specific disturbance over a selected region, i.e. the integration of the growth of the disturbance over the region itself, is denoted by E_1. In this case, this quantity integrated into a defined region serves as the chosen measure. Mathematically, this objective function is formulated as:

$$J(\tilde{\mathbf{q}}, \mathbf{X}) = E_1(\tilde{\mathbf{q}}, \mathbf{X}) = \frac{1}{2} \int_{X_{ms}}^{X_{me}} \int_{Z_0}^{Z_1} \int_0^{+\infty} (|\tilde{u}|^2 + |\tilde{v}|^2 + |\tilde{w}|^2) \, dx_1 \, dx_2 \, dx_3, \tag{2.3}$$

where the fact that kinetic energy is an explicit function of the velocity components of the chosen perturbation is highlighted. Moreover in equation (2.3) can be seen the dependence from the vector of nodal coordinates, $\mathbf{X}$. The amplitudes of disturbances used to compute the objective function are solutions of PSE equation (1.20) for a specific mode, identified by the parameters $\alpha(X_{ms})$, β, and ω.
By analyzing the envelopes of N-factor curves, we identify the most amplified mode

among various modes characterized by different ω and β values and utilize the disturbance amplitudes of that mode to compute the objective function.
At this juncture, optimizing the objective function J with respect to design variables, using gradient based method, entails calculating the sensitivity of the objective function with respect to these design variables. Here, we compute the gradient of our objective function, J with respect to the design variables a as

$$\nabla_a J = \frac{\partial J}{\partial \mathbf{Q}} \times \frac{\partial \mathbf{Q}}{\partial p_e} \times \frac{\partial p_e}{\partial a}, \tag{2.4}$$

where the first term on the RHS is given by the adjoint of PSE, the second term by the adjoint of BLE and the last term by the adjoint of Euler equations; we employ the adjoint method to efficiently compute the gradients.
Given the specialized nature of this approach, the following sections outline some of the most important steps of the derivations to enable the reader to fully comprehend the methodology used.

2.1 Derivation of the gradient

According to equation (2.3), the objective function computed for a single disturbance E_1 explicitly relies on the solution of the PSE (equations (1.20)–(1.22)), denoted as $\tilde{\mathbf{q}}$, and on the vector of nodal coordinates $\mathbf{X}$, representing the discretization points of the Euler mesh. Hence, we can express this dependency as follows:

$$J \equiv J(\tilde{\mathbf{q}}, \mathbf{X}). \tag{2.5}$$

Depending on the specific study under consideration, whether we're examining worst-case or best-case scenarios, our objective is to either maximize or minimize J, as defined in equation (2.5). Therefore, in both scenarios, calculating the gradient is essential to determine the most promising direction for iterative movement towards achieving the optimal condition. The latter is expressed here as:

$$\mathcal{A}_q(\tilde{\mathbf{q}}, \mathbf{Q}, \mathbf{X}) = \mathbf{0}, \tag{2.6}$$

where equation (2.6) is defined for given $\mathbf{X}$ and $\mathbf{Q}$. The mean flow $\mathbf{Q}$ is a solution of the BLE (1.8)–(1.11), denoted here as

$$\mathcal{A}_Q(\mathbf{Q}, \mathbf{w}, \mathbf{X}) = \mathbf{0}, \tag{2.7}$$

which is defined for a given $\mathbf{X}$ and $\mathbf{w}$. Finally, the inviscid flow $\mathbf{w}$ is the solution of the Euler equations (1.1)–(1.4), denoted

$$\mathcal{A}_w(\mathbf{w}, \mathbf{X}) = \mathbf{0}. \tag{2.8}$$

Following what Pralits *et al.* had done in their work [4], for the derivation of adjoints it is convenient to introduce the functions J_X, J_Q, and J_w, which are simply the objective function (2.5) in which various intermediate quantities are regarded as independent variables. The definitions of these intermediate functions are:

$$\begin{array}{rcccccc}
\text{Objective function:} & V_q \times V_X & \rightarrow & \mathbb{R} & & \\
 & \{\tilde{\mathbf{q}}, \mathbf{X}\} & & J(\tilde{\mathbf{q}}, \mathbf{X}) & & \\
\text{Subject to equation (2.6):} & V_Q \times V_X & \rightarrow & \mathbb{R} & & \\
 & \{\mathbf{Q}, \mathbf{X}\} & & J_Q(\mathbf{Q}, \mathbf{X}) & \equiv & J(\tilde{\mathbf{q}}(\mathbf{Q}, \mathbf{X}), \mathbf{X}) \\
\text{Subject to equations (2.6)–(2.7):} & V_w \times V_X & \rightarrow & \mathbb{R} & & \\
 & \{\mathbf{w}, \mathbf{X}\} & & J_w(\mathbf{w}, \mathbf{X}) & \equiv & J_Q(\mathbf{Q}(\mathbf{w}, \mathbf{X}), \mathbf{X}) \\
\text{Subject to equations (2.6)–(2.8):} & V_X & \rightarrow & \mathbb{R} & & \\
 & \mathbf{X} & & J_X(\mathbf{X}) & \equiv & J_w(\mathbf{w}(\mathbf{X}), \mathbf{X})
\end{array}$$

The coordinates of the mesh nodes, denoted as $\mathbf{X}$, are obtained through a mesh movement algorithm based on the displacements $\mathbf{y}$ of the nodes on the airfoil, i.e., $\mathbf{X} \equiv \mathbf{X}(\mathbf{y})$, as elaborated in Section 3. The displacements are controlled by parameters $\mathbf{a}$, such that $\mathbf{y} \equiv \mathbf{y}(\mathbf{a})$. To facilitate analysis, given a function J_X of the variable $\mathbf{X}$, it's convenient to define J_y and J_a as follows:

$$J_y(\mathbf{y}) = J_X(\mathbf{X}(\mathbf{y})), \tag{2.9}$$

$$J_a(\mathbf{a}) = J_y(\mathbf{y}(\mathbf{a})). \tag{2.10}$$

Summarizing the main aim of this treatment, our focus lies in optimizing the objective function J constrained by equations (2.6)–(2.8) in terms of the design parameters $\mathbf{a}$ employing a gradient-based approach. This requires the computation of the gradient ∇J_a, derived from ∇J_X. The efficiency of our approach stems from the adept compu-

tation of ∇J_X utilizing the adjoint method.
In the subsequent discussion, we assume that $\tilde{\mathbf{q}} \in V_q$, $\mathbf{Q} \in V_Q$, $\mathbf{w} \in V_w$, and $\mathbf{X} \in V_X$, where V_q, V_Q, V_w, and V_X are vector spaces equipped with inner products $\langle \cdot, \cdot \rangle_q$, $\langle \cdot, \cdot \rangle_Q$, $\langle \cdot, \cdot \rangle_w$, and $\langle \cdot, \cdot \rangle_X$, respectively. Additionally, we assume that all mappings are differentiable. For instance, $\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}}$ represents the linearization with respect to $\tilde{\mathbf{q}}$ of the mapping $\mathcal{A}_q$ at the given state $\{\tilde{\mathbf{q}}, \mathbf{Q}, \mathbf{w}, \mathbf{X}\}$. Notations $(\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}})^{-1}$ and $(\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}})^*$ denote the inverse and the adjoint of the linearized mapping $(\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}})$, respectively. Finally, $(\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}})\delta\tilde{\mathbf{q}}$ denotes the application of $\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}}$ on $\delta\tilde{\mathbf{q}}$.

2.2 Sensitivity of PSE

For arbitrary variations $\{\delta\mathbf{Q}, \delta\mathbf{X}\} \in V_Q \times V_X$, of $\{\mathbf{Q}, \mathbf{X}\}$ in PSE, the first variation in the solution, denoted as $\delta\tilde{\mathbf{q}}$ and belonging to $V_{\mathbf{q}}$, is determined by the sensitivity equations:

$$\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}}\delta\tilde{\mathbf{q}} = -\frac{\partial \mathcal{A}_q}{\partial \mathbf{Q}}\delta\mathbf{Q} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}\delta\mathbf{X}. \tag{2.11}$$

Moreover, for any variations $\{\delta\tilde{\mathbf{q}}, \delta\mathbf{X}\} \in V_q \times V_X$, the first variation of the objective function J is defined as:

$$\delta J = \left\langle \frac{\partial J}{\partial \tilde{\mathbf{q}}}, \delta\tilde{\mathbf{q}} \right\rangle_q + \left\langle \frac{\partial J}{\partial \mathbf{X}}, \delta\mathbf{X} \right\rangle_X, \tag{2.12}$$

where $\langle \cdot, \cdot \rangle_q$ and $\langle \cdot, \cdot \rangle_X$ denote the inner products in V_q and V_X respectively. In the following, $\delta\tilde{\mathbf{q}}$ is the solution of the sensitivity equations (2.11), leading to a revised expression for (2.12):

$$\delta J = \left\langle \frac{\partial J}{\partial \tilde{\mathbf{q}}}, \left(\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}}\right)^{-1} \left(-\frac{\partial \mathcal{A}_q}{\partial \mathbf{Q}}\delta\mathbf{Q} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}\delta\mathbf{X}\right) \right\rangle_q + \left\langle \frac{\partial J}{\partial \mathbf{X}}, \delta\mathbf{X} \right\rangle_X, \tag{2.13}$$

where, for $\tilde{\mathbf{q}}$ being the solution of (2.6) and $\delta\tilde{\mathbf{q}}$ being the solution of (2.11), the definition of J_Q results in:

$$\delta J_Q = \delta J. \tag{2.14}$$

The gradient of the functional J_Q is $\nabla J_Q = \{\frac{\partial J_Q}{\partial \mathbf{Q}}, \frac{\partial J_Q}{\partial \mathbf{X}}\}$ and resides in the vector space $V_Q \times V_X$, such that for all $\{\delta\mathbf{Q}, \delta\mathbf{X}\}$ in $V_Q \times V_X$, we have:

$$\delta J_Q = \left\langle \frac{\partial J_Q}{\partial \mathbf{Q}}, \delta\mathbf{Q} \right\rangle_Q + \left\langle \frac{\partial J_Q}{\partial \mathbf{X}}, \delta\mathbf{X} \right\rangle_X. \tag{2.15}$$

By employing the adjoint operator $\frac{\partial \mathcal{A}_q}{\partial \tilde{q}}$ in expression (2.14) and utilizing (2.15), we derive:

$$\begin{aligned}\delta J_Q =& -\left\langle \left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{Q}}\right)^* \left(\left(\frac{\partial \mathcal{A}_q}{\partial \tilde{q}}\right)^{-1}\right)^* \frac{\partial J}{\partial \tilde{\mathbf{q}}}, \delta \mathbf{Q}\right\rangle_Q \\ &- \left\langle \left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}\right)^* \left(\left(\frac{\partial \mathcal{A}_q}{\partial \tilde{q}}\right)^{-1}\right)^* \frac{\partial J}{\partial \tilde{\mathbf{q}}}, \delta \mathbf{X}\right\rangle_X \\ &+ \left\langle \frac{\partial J}{\partial \mathbf{X}}, \delta \mathbf{X}\right\rangle_X .\end{aligned} \tag{2.16}$$

This expression can be further manipulated using the definitions of the adjoint operators

$$\left(\frac{\partial \mathcal{A}_q}{\partial \tilde{\mathbf{q}}}\right)^* \mathbf{q}^* = \frac{\partial J}{\partial \tilde{\mathbf{q}}} \tag{2.17}$$

Hence, by introducing the adjoint state $\mathbf{q}^*$ satisfying the system we conclude that:

$$\begin{aligned}\frac{\partial J_Q}{\partial \mathbf{Q}} &= -\left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{Q}}\right)^* \mathbf{q}^* \\ \frac{\partial J_Q}{\partial \mathbf{X}} &= \frac{\partial J}{\partial \mathbf{X}} - \left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}\right)^* \mathbf{q}^*.\end{aligned} \tag{2.18}$$

The cost for obtaining the gradient of J_Q is reduced to one solution of the system equation (2.17) and two matrix-vector products as shown in (2.18).

2.3 Sensitivity of BLE

For arbitrary variations $\{\delta \mathbf{w}, \delta \mathbf{X}\} \in V_w \times V_X$ of $\{\mathbf{w}, \mathbf{X}\}$ in the BLE equation (2.7), the first variation of the solution of the BLE is denoted $\delta \mathbf{Q} \in V_Q$, and is defined by the sensitivity equations:

$$\frac{\partial \mathcal{A}_Q}{\partial \mathbf{Q}} \delta \mathbf{Q} = -\frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}} \delta \mathbf{w} - \frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}} \delta \mathbf{X}. \tag{2.19}$$

Furthermore, from the definition (2.15) and the expression of the gradient (2.18), for arbitrary variations $\{\delta Q, \delta X\}$ in $V_Q \times V_X$, the variation δJ_Q is

$$\delta J_Q = \left\langle -\left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{Q}}\right)^* \mathbf{q}^*, \delta \mathbf{Q}\right\rangle_Q + \left\langle \frac{\partial J}{\partial \mathbf{X}} - \left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}\right)^* \mathbf{q}^*, \delta \mathbf{X}\right\rangle_X . \tag{2.20}$$

In the following, $\delta\mathbf{Q}$ is the solution of the sensitivity equation equation (2.19). The variation δJ_Q is expressed, making use of equation (2.20) and equation (2.19), as

$$\begin{aligned}\delta J_Q = &\left\langle -\left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{Q}}\right)^* \mathbf{q}^*, \left(\frac{\partial\mathcal{A}_Q}{\partial\mathbf{Q}}\right)^{-1}\left(-\frac{\partial\mathcal{A}_Q}{\partial\mathbf{w}}\delta\mathbf{w} - \frac{\partial\mathcal{A}_Q}{\partial\mathbf{X}}\delta\mathbf{X}\right)\right\rangle_Q \\ &+\left\langle \frac{\partial J}{\partial\mathbf{X}} - \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{X}}\right)^* \mathbf{q}^*, \delta\mathbf{X}\right\rangle_X .\end{aligned} \tag{2.21}$$

And, for $\mathbf{Q}$ solution of equation (2.7) and $\delta\mathbf{Q}$ solution of equation (2.19), the definition of J_w yields

$$\delta J_w = \delta J_Q. \tag{2.22}$$

The gradient of J_w is the vector $\{\frac{\partial J_w}{\partial\mathbf{w}}, \frac{\partial J_w}{\partial\mathbf{X}}\}$ in the product space $V_w \times V_X$ such that for all $\{\delta\mathbf{w}, \delta\mathbf{X}\}$ in $V_w \times V_X$, we have

$$\delta J_w = \left\langle \frac{\partial J_w}{\partial\mathbf{w}}, \delta\mathbf{w}\right\rangle_w + \left\langle \frac{\partial J_w}{\partial\mathbf{X}}, \delta\mathbf{X}\right\rangle_X . \tag{2.23}$$

Using the adjoint of the inverse linearized BLE operator $(\frac{\partial\mathcal{A}_Q}{\partial\mathbf{Q}})^{-1}$ in equation (2.21), δJ_w is expressed as

$$\begin{aligned}\delta J_w = &\left\langle -\left(\left(\frac{\partial\mathcal{A}_Q}{\partial\mathbf{Q}}\right)^{-1}\right)^* \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{Q}}\right)^* \mathbf{q}^*, -\frac{\partial\mathcal{A}_Q}{\partial\mathbf{w}}\delta\mathbf{w} - \frac{\partial\mathcal{A}_Q}{\partial\mathbf{X}}\delta\mathbf{X}\right\rangle_Q \\ &+\left\langle \frac{\partial J}{\partial\mathbf{X}} - \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{X}}\right)^* \mathbf{q}^*, \delta\mathbf{X}\right\rangle_X .\end{aligned} \tag{2.24}$$

Using the adjoints of $\frac{\partial\mathcal{A}_q}{\partial\mathbf{w}}$ and $\frac{\partial\mathcal{A}_q}{\partial\mathbf{X}}$, the equation of δJ_w reported above can be rewritten as

$$\begin{aligned}\delta J_w = &\left\langle \left(\frac{\partial\mathcal{A}_Q}{\partial\mathbf{w}}\right)^* \left(\left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{Q}}\right)^{-1}\right) \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{Q}}\right), \delta\mathbf{w}\right\rangle_w \\ &+\left\langle \left(\frac{\partial\mathcal{A}Q}{\partial\mathbf{X}}\right)^* \left(\left(\frac{\partial\mathcal{A}Q}{\partial\mathbf{Q}}\right)^{-1}\right) \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{Q}}\right), \delta\mathbf{q}\right\rangle_X \\ &+\left\langle \frac{\partial J}{\partial\mathbf{X}} - \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{X}}\right), \delta\mathbf{X}\right\rangle_X .\end{aligned} \tag{2.25}$$

$$\left(\frac{\partial\mathcal{A}_Q}{\partial\mathbf{Q}}\right)^* \mathbf{Q}^* = \left(\frac{\partial\mathcal{A}_q}{\partial\mathbf{Q}}\right)^* \mathbf{q}^*. \tag{2.26}$$

$$\frac{\partial J_w}{\partial \mathbf{w}} = \left(\frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}}\right)^* \mathbf{Q}^*, \quad \frac{\partial J_w}{\partial \mathbf{X}} = \frac{\partial J}{\partial \mathbf{X}} - \left(\frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}\right)^* \mathbf{q}^* + \left(\frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}}\right)^* \mathbf{Q}^*. \tag{2.27}$$

The use of adjoint equations limits the cost for obtaining the gradient of J_w to solving the systems equation (2.17) and equation (2.26), as well as four matrix–vector products: one to 'assemble' the right-hand side of the adjoint system equation (2.26) and three to obtain the final expression equation (2.27).

2.4 Sensitivity of Euler equations

For arbitrary variations $\delta\mathbf{X} \in V_X$ of $\mathbf{X}$ in the Euler equation (2.7), the first variation of the solution of the Euler equation is denoted $\delta\mathbf{w} \in V_w$, and is defined by the sensitivity equation. The sensitivity equation is given by:

$$\frac{\partial \mathcal{A}_{\mathbf{w}}}{\partial \mathbf{w}} \delta\mathbf{w} = -\frac{\partial \mathcal{A}_w}{\partial \mathbf{X}} \delta\mathbf{X}. \tag{2.28}$$

Furthermore, for arbitrary variations $\{\delta\mathbf{w}, \delta\mathbf{w}\}$ in $V_w \times V_X$, the first variation of the functional J_w is expressed from the gradient (2.27):

$$\delta J_w = \left\langle \frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}}^* \mathbf{Q}^*, \delta\mathbf{w} \right\rangle_w + \left\langle \frac{\partial J}{\partial \mathbf{X}} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}^* \mathbf{q}^* + \frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}}^* \mathbf{Q}^*, \delta\mathbf{X} \right\rangle_X. \tag{2.29}$$

In the following, $\delta\mathbf{w}$ is the solution of the sensitivity equation (2.28), which enables us to rewrite expression (2.29) as:

$$\delta J_w = \left\langle \frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}}^* \mathbf{Q}^*, -\frac{\partial \mathcal{A}_w}{\partial \mathbf{w}}^{-1} \frac{\partial \mathcal{A}_w}{\partial \mathbf{X}} \delta\mathbf{X} \right\rangle_w \tag{2.30}$$

$$+ \left\langle \frac{\partial J}{\partial X} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}^* \mathbf{q}^* + \frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}}^* \mathbf{Q}^*, \delta\mathbf{X} \right\rangle_X, \tag{2.31}$$

and, for $\mathbf{w}$ solution of (2.7) and δw solution of (2.28), the definition of J_X yields:

$$\delta J_X = \delta J_w \tag{2.32}$$

The gradient of J_X is the vector ∇J_X in the space V_X such that for all $\delta\mathbf{X}$ in V_X we have:

$$\delta J_X = \langle \nabla J_X, \delta\mathbf{X} \rangle_X \tag{2.33}$$

The adjoint of the linearized Euler operator is used in (2.31) to express δJ_X (2.32) as:

$$\delta J_X = \left\langle \frac{\partial \mathcal{A}_w}{\partial \mathbf{w}}^{-1*} \frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}}^{*} \mathbf{Q}^*, -\frac{\partial \mathcal{A}_w}{\partial \mathbf{X}} \delta \mathbf{X} \right\rangle_w + \left\langle \frac{\partial J}{\partial \mathbf{X}} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}^{*} \mathbf{q}^* + \frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}}^{*} \mathbf{Q}^*, \delta \mathbf{X} \right\rangle_X \tag{2.34}$$

The adjoint instead of the linear operator $\frac{\partial \mathcal{A}_w}{\partial \mathbf{X}}$ is used in (2.34) and leads to:

$$\delta J_X = \left\langle \frac{\partial J}{\partial \mathbf{X}} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}^{*} \mathbf{q}^* + \frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}}^{*} \mathbf{Q}^*, \delta \mathbf{X} \right\rangle_X \tag{2.35}$$

$$- \left\langle \frac{\partial \mathcal{A}_w}{\partial \mathbf{X}}^{*} \frac{\partial \mathcal{A}_w}{\partial \mathbf{w}}^{-1*} \frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}}^{*} \mathbf{Q}^*, \delta \mathbf{X} \right\rangle_X . \tag{2.36}$$

The method of adjoint is again applied as we define an adjoint state $\mathbf{w}^*$, here solution of the system:

$$\frac{\partial \mathcal{A}_w}{\partial \mathbf{w}}^{*} \mathbf{w}^* = \frac{\partial \mathcal{A}_Q}{\partial \mathbf{w}}^{*} \mathbf{Q}^*. \tag{2.37}$$

This enables us to give expression for the gradient in expression (2.33):

$$\nabla J_X = \frac{\partial J}{\partial \mathbf{X}} - \frac{\partial \mathcal{A}_q}{\partial \mathbf{X}}^{*} \mathbf{q}^* + \frac{\partial \mathcal{A}_Q}{\partial \mathbf{X}}^{*} \mathbf{Q}^* - \frac{\partial \mathcal{A}_w}{\partial \mathbf{X}}^{*} \mathbf{w}^* \tag{2.38}$$

The total cost of this gradient evaluation is three adjoint systems (2.17), (2.25), and (2.37), and five matrix-vector products: two for the assembly of the right-hand sides of the systems (2.25) and (2.37), and three for the final expression (2.38).

Chapter 3

Adjoint equations

It has been already mentioned that the efficient method for computing the sensitivities is the adjoint method. The cost of solving an adjoint equation is comparable to the cost of solving the governing equation. Once the adjoint solution is obtained, the sensitivities to any number of parameters can be obtained with little effort.
In principle, two distinct methods exist for deriving these adjoint equations. The first method, sometimes referred to as the discrete approach or discretize-then-differentiate, involves deriving the adjoint equations from the discretized set of state equations. Conversely, the second method, known as the continuous approach or differentiate-then-discretize, entails deriving the adjoint equations from the continuous state equations. Subsequently, the continuous adjoint equations undergo discretization, typically following a similar procedure as the corresponding state equations.
The preceding discussion has led us to derive the expression of the gradient ∇J_X, which, as displayed in equation (2.38), depends on the adjoint states of the PSE (2.17), the BLE (2.25), and the Euler equations (2.37), namely $\mathbf{q}^*$, $\mathbf{Q}^*$, and $\mathbf{w}^*$.
The adjoints of the BLE and PSE were derived using a continuous approach by Pralits *et al.* in [4]–[7]–[8] and Amoignon *et al.* in [2], as implemented in the codes used. Detailed derivations of the adjoint equations can be found in their respective studies cited above. For the adjoint of the Euler equations, we utilize instead the discrete adjoint implementation provided by ADflow developers.
In order to avoid verbosity regarding the treatment of the adjoints, only the brief notes reported above are provided here, omitting mathematical details. For details on the adjoint of the Euler equations implemented in ADflow, readers can refer to the comprehensive work by Kenway *et al.* [9](especially in AppendixA). This work also includes a review of adjoint methods available in literature, discussing their advantages

and disadvantages. The choice of particular TransformationAD [1] method for partial derivative computation and Krylov as the adjoint solver in ADflow is discussed in detail. It is important to note that the flow solver used here, namely ADflow code, differs from that used by Pralits *et al.* in their previous optimization studies.

[1]Automatic Differentation (AD) focuses directly on the computer code and uses the fact that any code consists of a sequence of elementary arithmetic operations. These operations can then be differentiated to create differentiated versions of the code that compute derivatives using the chain rule. There are two derivative computation modes for AD: forward and reverse. Both use the same chain rule but accumulate the derivatives in different directions. In forward mode AD, the input variables of interest are specified and the derivatives with respect to those variables are accumulated in the forward direction, together with the execution of the original code. In reverse-mode AD, on the other hand, the outputs of interest are specified and the derivatives of those outputs are accumulated backward. Before this reverse accumulation is performed, however, the original code must be run (forward) and all intermediate variables must be stored for use in the reverse accumulation.

Part II

Section Two

Chapter 4

Methodology

As mentioned in the introduction, the goal of this work is to use sensitivity analysis for optimization studies applied to the shape of NLF airfoils. In Part I, the theoretical framework for deriving sensitivity analysis has been presented, highlighting the significant reduction in computational cost achievable by employing the adjoint method, specifically discussed in Chapter 3. At this point, it is important to clarify the solution scheme actually followed, and this is done schematically in Figure 4.1.

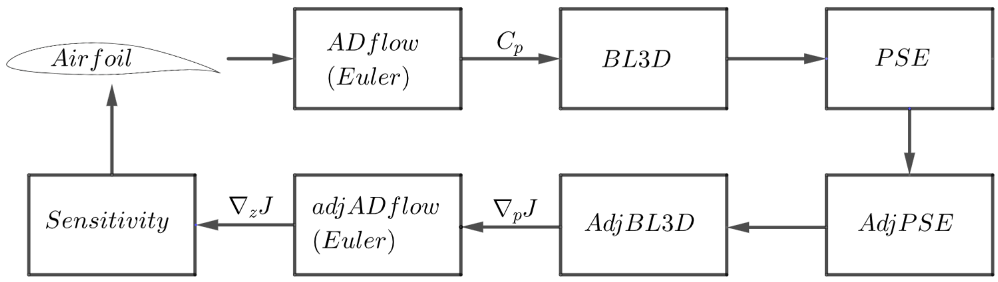


Figure 4.1 Flowchart for calculating the gradient.

Computational steps can be summarized as follows:

1. solve Euler equations to find pressure distribution C_p on the airfoil surface;
2. solve compressible BLE using the distribution of C_p as input, and compute the mean flow quantities inside the boundary layer;
3. solve compressible PSE inside the boundary layer;
4. solve adjoint PSE (APSE);

5. solve adjoint BLE (ABLE) to find sensitivity with respect to pressure distribution, $\nabla_{P_e} J$;
6. solve adjoint Euler equations to find sensitivity with respect to design variables, $\nabla_z J$;
7. use sensitivity, $\nabla_z J$, to update the geometry.

In this solution scheme, as presented in Section I, two distinct regions of the flow are considered: the solution of the Euler equation for the external flow and the solution of the BLE in regions where the viscous effects of the flow cannot be neglected as they dominate the physics of flow development. Solutions of this type are commonly referred to as coupled viscous-inviscid, and the literature extensively discusses the potential and accuracy of this method. In this regard, in the work of Amoignon *et al.* [10], a numerical test was performed to evaluate the accuracy of the gradient computation in the framework of coupled solution, using adjoint approach.
In the connection between the inviscid and viscous solution, two significant issues greatly affect the precision of the gradients. Firstly, the grid resolution typically employed to yield outcomes using the Euler equations proves inadequate to achieve converged results with the stability equations. Secondly, the derivation process of the adjoint equations raises concern. While the adjoint Euler equations stem from the discretized Euler equations, the accuracy of gradients computed via this method ought not to hinge on the grid resolution. Conversely, the derivation of the ABLE and APSE relies on the continuous approach, possibly implicating the accuracy of gradients obtained through this route on grid resolution.
Moreover, Amoignon *et al.* in their work, [10], stated that solving the gradient based on the discretized problem is a critical issue in optimization because of difficulties related to low accuracy, such as finding descent directions even far from the optimal design, are quite common. Indeed, the derivation of the adjoint of the discretized Euler equations involves an approximation by not linearizing the coefficients of the second-order artificial dissipation[1]. Despite the limitations just highlighted, based on the results of the same authors' observations, we can say that, by using a sufficiently accurate grid, the presented solution algorithm is accurate.
To summarize, the approach aims to derive the mean flow by solving the boundary-layer

[1]Second-order artificial dissipation is a numerical technique used in CFD to stabilize the solution of PDEs governing fluid flow by dissipating spurious oscillations or numerical instabilities that may arise, particularly near discontinuities or shock waves. It specifically refers to a type of dissipation that is proportional to the second spatial derivative of the solution.

equations. The mean flow solution, together with the coordinates, acts as input for the stability equations. The solutions of the adjoint equations corresponding to the stability and boundary-layer equations are then utilized as input for the adjoint equations of the Euler equations. The final gradient is computed by coupling these three systems of adjoints. From the adjoint equations of the boundary-layer equations, an expression for the partial derivative of the objective function with respect to the pressure distribution on the edge of boundary layer (p_e)is obtained. Similarly, from the adjoint equations of the Euler equations, an expression for the gradient with respect to design variables is derived.

4.1 Codes used

For this study, it became necessary to utilize several codes, some of which are open-source and publicly available, while others were developed in-house. Performing optimization routine requires that these different codes can operate sequentially, ensuring that the output produced by one code is executable by the next. The installation of the various required modules (in Fortran and Python) is inherently tricky, and thus, following the guidance provided at the link *https://mdolab-mach-aero.readthedocs-hosted.com/en/latest/index.html* in the installation section has proven beneficial. This approach helps to prevent conflicts between different module versions.
The framework employed in this work can be seen, in part, as an extension of the *MACH-Aero framework* developed by the *Multidisciplinary Design Optimization Laboratory (MDO) Lab*, from the University of Michigan. Specifically, they share the same flow equation solver. As previously mentioned, the first step is to find the pressure distribution, C_p, on the airfoil surface by solving the Euler equations. To accomplish this, we utilized the ADflow code (for more details, refer to [11]) , which serves as a 3D structured compressible Euler and RANS finite-volume solver. To enhance the solver's robustness, we employed the approximate Newton–Krylov solver implemented in ADflow (for more details, refer to [12]) .
Noting the pressure distribution around the profile, some in-house codes are employed for solving the compressible flow equations and stability equations within the boundary layer. Regarding these steps, in Table 4.1, the codes used are mentioned. As seen in Table 4.1, the computation of mean flow within the boundary layer is carried out using the BL3D code, which takes as input the pressure distribution around the airfoil obtained by solving the Euler equations. The code has been developed to account for

Table 4.1 In-house codes employed for computing mean flow quantities and perform stability analysis within the boundary layer.

Equations	**Code**	
solved	**Direct solution**	**Adjoint solution**
BLE	BL3D	ABL3D
PSE	NOLOT	ADNOLOT

non-parallel flow conditions, allowing for the consideration of cross-flow components that may arise on wings with non-zero sweep angles, as in the case study considered in this work.

The details regarding the NOLOT code are extensively covered in [13], and for this reason, they are not reiterated here. The code is utilized for solving compressible Parabolized Stability Equations (PSE) and has been developed to consider compressibility, surface curvature, and non-parallel effects within the boundary layer, encompassing all conditions that may occur in the study of flow around the airfoil.

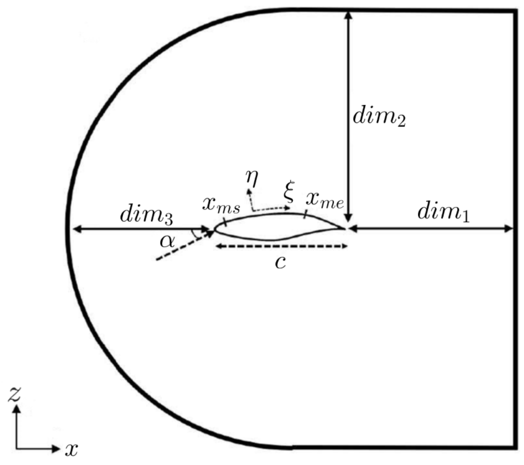
After solving all direct equations, we will proceed to solve the adjoint equations in reverse order: firstly, the adjoint of the PSE (APSE) will be solved using the ADNOLOT code, followed by the adjoint of BLE (ABLE) using the ABL3D code. Both APSE and ABLE are adjoint codes of the previously mentioned codes, and their reference documentation is [4]. After solving the adjoint equations within the boundary layer, the adjoint of the Euler equations is solved using the open-source code ADflow [9]. At the end of this chain, upon solving the adjoint of the Euler equations, we obtain the desired gradients. Hence, sensitivity analysis can be performed to modify the geometry and achieve the variations desired in the objective function. The complete set of options used in the various aforementioned codes is provided for completeness in the corresponding Appendix at the end of this work.

Chapter 5

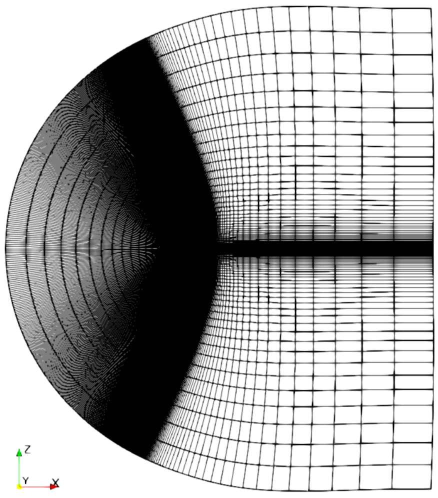
Flow case

The objective of this study is to implement an optimization calculation referring to the infinite swept wing condition. The infinite swept wing problem is a classical aerodynamic problem used to study the flow over a wing with an infinite span. This simplified configuration allows the analysis of fundamental aerodynamic phenomena without the complexities introduced by wingtip effects and finite span. In this problem, the wing is assumed to have an infinite span in the chordwise direction, meaning that the flow conditions at any given chordwise location are identical. Typically, the wing is modeled with a uniform cross-section along the chord, and the flow is assumed to be two-dimensional in the plane perpendicular to the chord. This problem is particularly interesting in the field of aerodynamics due to its relevance to real-world aircraft configurations, especially high aspect ratio wings found in many modern aircraft designs. Researchers use the infinite swept wing problem to investigate various aerodynamic characteristics, such as lift and drag distributions, spanwise flow patterns, and the effects of sweep angle on aerodynamic performance. The insights gained from studying this simplified problem can then be applied to more complex configurations to improve aircraft design and performance.
The reason for considering this type of problem is to extend the preliminary study conducted by Moniripiri et al. [1] to the case of a non-zero sweep angle, Λ. This involves considering not only the typical TS instabilities of a rectangular wing but also the cross-flow instabilities that occur in the flow around tapered wings. The subsequent objective of this work is to implement a new optimization algorithm to delay the transition of the boundary layer on tapered wings by appropriately modifying the wing profile shape, since wings with this configuration are widely used in general aviation planes.

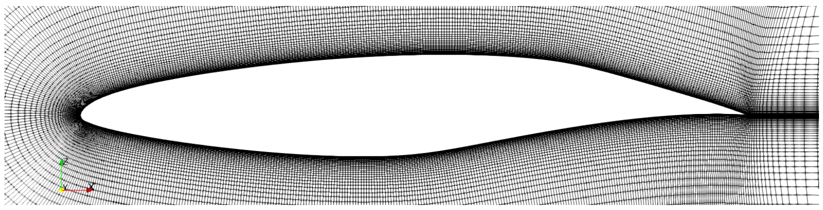
Since boundary conditions significantly influence the results of the study, the developed algorithm has been kept as generalized as possible so that it can accommodate different flow conditions with minimal implementation effort. It's worth noting the ease with which the geometry of the considered wing (baseline airfoil coordinates, Λ, etc.) and consequently the properties of the employed mesh can be modified.
In this specific case, the NLF(2)-0415 airfoil, designed for commuter aircraft applications, was selected for analysis to potentially compare the results of Moniripiri *et al.* obtained for $\Lambda = 0°$ with those reported here for $\Lambda \neq 0°$. This airfoil is optimized for natural laminar flow, making it interesting to assess how surface waviness may affect boundary layer stability characteristics, in addition to attempting to modify its surface to pursue the benefits of delaying the transition of the boundary layer.
As a purely informative note, some preliminary simulations regarding the study of boundary layer transition were also conducted for the NLF Mj3 airfoil. This decision was made because a similar study, albeit following an experimental approach, was conducted at TU Delft, yielding satisfactory results. Building on this observation, the initial intention was to partially recreate what was highlighted by the Dutch colleagues, while attempting to optimize the profile's performance. However, since the results of their study have not yet been made public, a comparison would not have been possible, albeit certainly interesting. Nevertheless, since the development of this algorithm still needs to be refined, the results related to the Mj3 profile are not published here; in any case, this once again, confirms the high flexibility of the developed algorithm.
For the investigation of flow conditions, a parametric study approach was adopted to analyze and compare the most influential variables. The selected flight conditions were defined by specifying the values of Mach number, M, Reynolds number, Re, and the angle of attack, AoA, in the numerical codes. Additionally, the thermodynamic properties of the air were computed for an altitude of 9600m, following the U.S. Standard Atmosphere 1976, [14].
In the results sections, the specific flow conditions utilized will be detailed. The computational domain, employed for solving the Euler equations, along with the AoA, and the boundaries (x_{ms} and x_{me}) delineating the region on the airfoil used for stability analysis and optimization, are depicted in Figure 5.1. It's important to note that a C-domain grid structure was employed, typical for studies involving sharp trailing-edged airfoils; this allows the utilization of a structured mesh. A farfield boundary condition encompasses the domain, while a no-penetration boundary condition is enforced on the airfoil surface. Although Figure 5.1 presents a 2D cross-section, it is important to



(a) Schematic representation of computational domain.



(b) Mesh of computational domain.



(c) Representation of computational domain.

Figure 5.1 Computational domain and meshes.

emphasize that the actual mesh utilized is three-dimensional, as ADflow is a 3D CFD solver. Thus, the domain is extruded in the spanwise direction with one volume cell. Regarding the choice of boundary conditions, it's noteworthy to observe the following: the desired flow condition resembles that of an infinite sweep problem, necessitating the use of periodic boundary conditions on the domain's sides instead of the conventional symmetric boundary conditions. This approach allows for consideration of spanwise invariance while maintaining the angle Λ condition and ensuring rigor in defining the incoming and outgoing flow surfaces. However, during the implementation of simulations, it was observed that periodic boundary conditions are not implemented within the ADflow code. Therefore, after suitable geometric transformation, symmetric boundary conditions were employed. Further details regarding this transformation are provided in Section 5.1.

Concerning other aspects of the mesh, a structured grid with 1198 grid points (598 volume cells) on the airfoil surface was utilized. The total mesh comprised 69,201 volume cells generated using the Construct2D mesh generator code [15]. The mesh dimensions depicted in Figure 5.1a are not to scale; the values of the dimensions (dim_1,

dim_2, dim_3) were set relative to the chord length, c (unitary dimension), with respective values of $150c$, $170c$, and $140c$. It's important to highlight that, for the purpose of validating the adjoint implementation, a mesh with 792 grid points (395 volume cells) on the airfoil was employed to mitigate computational costs. However, the finer grid (with 1198 grid points on the surface) was utilized for all other simulations conducted in this study.

5.1 Transformation 2.5D

In Chapter 5, we discussed the necessity of considering the transformation of the geometric coordinates of the airfoil due to the absence of periodic boundary conditions in ADflow. Specifically, we need to transition from the streamwise coordinates, which are the conventional coordinates used to describe the wing geometry (i.e., considered in the direction of the undisturbed flow), to the coordinates in the direction referred to as normal-to-chord, i.e., in the direction normal to that defined by the wing's leading edge. Obviously, in the case where the angle Λ is zero, the two directions coincide. In this way, employing symmetric boundary conditions, it will be possible to bypass the limitation of the chosen CFD solver, while still considering the desired flow conditions. For clarity, refer to Figure 5.2, which illustrates the reference problem and where the aforementioned coordinate systems, streamwise (x, y, z) and normal-to-chord (x', y', z'), are shown. For completeness, the definition of the curvilinear abscissa s and the curvilinear coordinate system (ξ, η, z) are also provided, the latter of which is not particularly relevant to our discussion; the infinitesimal coordinate ds is instead defined as $ds = dx^2 + dy^2 + dz^2$.

Therefore, it is necessary to introduce the transformation. Actually, two types of transformations are possible, which are subsequently reported in mathematical formula:

$$\begin{cases} x \;\rightarrow\; x\cos\Lambda \\ z \;\rightarrow\; z \end{cases} , \qquad \begin{cases} x \;\rightarrow\; x \\ z \;\rightarrow\; \dfrac{z}{\cos\Lambda} \end{cases} . \tag{5.1}$$

In the treatment performed, it was chosen to use the transformation on the right as reported above; however, the same result could have been obtained using the other proposal. At this point, it is necessary to transform the quantities of the flow appropriately, in particular:

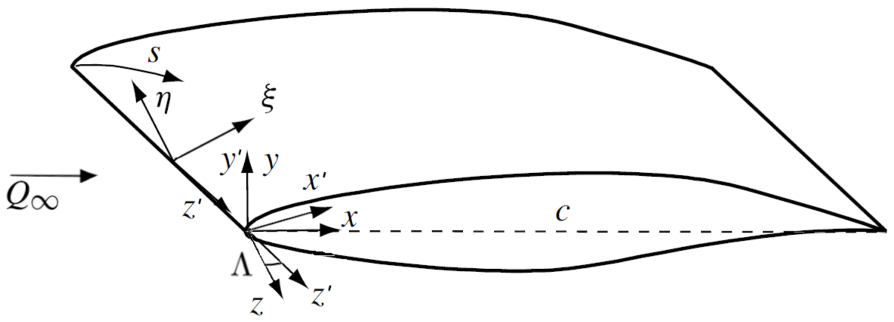


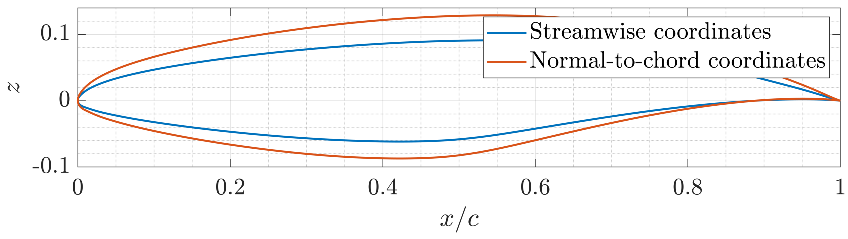
Figure 5.2 Representation of the infinite swept wing problem studied with their respective coordinate systems used. Please note that in this figure, the y-z axes (and the corresponding rotated ones) are inverted compared to the rest of the treatment.

$$\begin{aligned} Q_{2D} &= Q\cos\Lambda \\ M_{2D} &= M\cos\Lambda \end{aligned} \tag{5.2}$$

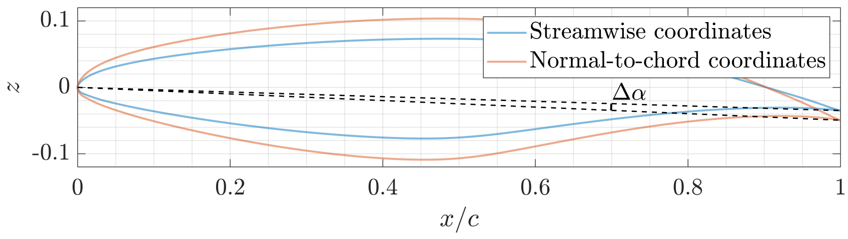
where the quantities without subscript refers to the variables of the actual three-dimensional flow, while the subscript 2D refers to the flow conditions corresponding to the actual ones but considered in such a way that it is possible to apply the symmetric boundary conditions. The rigorous definition of these quantities is of fundamental importance for a correct execution of the study, as the output of interest of the simulation in ADflow is the pressure distribution, C_p, which will be used as input data in the BL3D code for solving the equations within the boundary layer, and this obviously depends on the velocity field around the profile. Here it is noted that BL3D is capable of considering non-parallel flow conditions, so with Λ different from zero; this means that after having calculated the solutions of Euler, it is possible to return to the streamwise coordinate system.
So the calculation of C_p must account for the wing's sweep angle, as depicted in the following equations:

$$\begin{gathered} C_{p_{2D}} = \frac{p-p_\infty}{\frac{1}{2}\rho_\infty Q_{2D}^2} = \frac{p-p_\infty}{\frac{1}{2}\rho_\infty Q^2}\frac{1}{\cos^2\Lambda} = C_p\frac{1}{\cos^2\Lambda} \\ C_p = C_{p_{2D}}\cos^2\Lambda \end{gathered} \tag{5.3}$$

Referring to the NLF(2)-0415 airfoil already mentioned, in Figure 5.3a, we want to show the transformation of the coordinate system just mentioned; as observed, the geometry in the direction normal to the chord has a greater thickness compared to the streamwise one. The procedure just described, which allows to switch from the aerodynamic performance of a straight wing to those of a wing with sweep angle, in the literature is also known by the name of 2.5D transformation; further information regarding this can be found in [16], where the treatment is also extended to the case of wings with sweep and taper (2.75D transformation).



(a) Representation of the NLF(2)-0415 airfoil profile in streamwise and normal to chord coordinates at AoA.



(b) Representation of the NLF(2)-0415 airfoil profile in streamwise and normal to chord coordinates at $AoA = 2°$ for both cases. The normal to the chord geometry is obtained by rotating the original one by the angle AoA and then applying the transformation described by Equations 5.1. The observed Δ angle originates from the transformation if the corrective factor of the angle of attack for passing from AoA to AoA_{2D} is not considered (refer to Equations 5.3).

Figure 5.3 Graphical visualization of the 2.5D transformation described in Section 5.1 with reference to the NLF(2)-0415.

It is worth noting that the transformation just described is not sufficient to ensure that the flow conditions in the two different coordinate systems are equivalent. It is observed that, by introducing a non-zero AoA, rotating the coordinates of the original airfoil, i.e., giving incidence to the flow, and then transforming them as described in Equations 5.1, results in the creation of a Δ angle, as illustrated by way of example in Figure 5.3b for the NLF(2)-0415 airfoil. Following simple algebraic steps, it is possible to introduce the correction for AoA, while maintaining a strict correspondence:

$$\begin{aligned} AoA &\rightarrow AoA_{2D} \\ AoA &= \arctan\left[\frac{z}{x}\right] \\ AoA_{2D} &= \arctan\left[\frac{z/\cos\Lambda}{x}\right] = \text{atan}\left[\frac{\tan AoA}{\cos\Lambda}\right] \end{aligned} \tag{5.4}$$

For greater clarity, Figure 5.4 shows an illustration of the solution process followed. Despite the considerations mentioned above, throughout the following, reference is always made solely to the streamwise flow quantities, as they are the ones that actually matter for comparative studies with other results, etc., since the other quantities have been introduced only to bypass the ADflow limit.

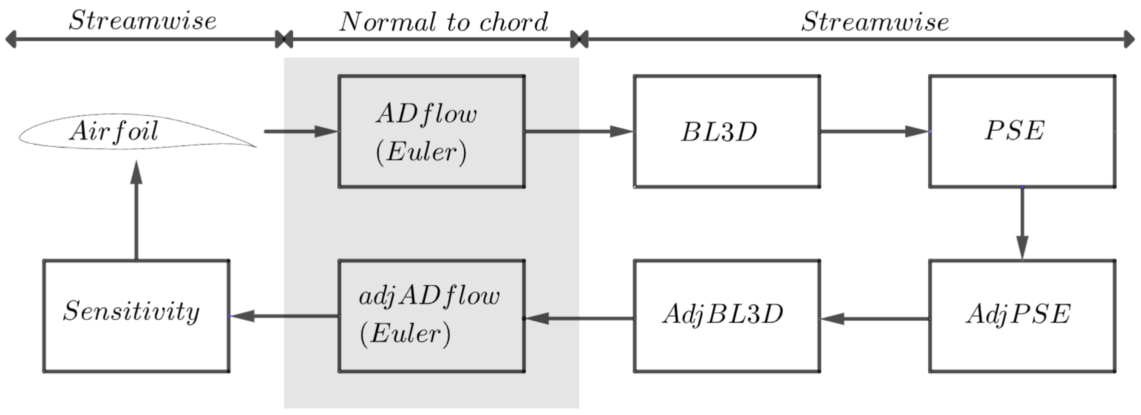


Figure 5.4 Flowchart of the coupled solution method in the streamwise and normal to chord directions.

Chapter 6

Algorithms for finding manufacturing tolerances

In this Section, we describe the methodology used to calculate dimensional tolerances, which we defined at the outset of this study as the worst-case scenario condition. The approach follows the philosophy developed by Moniripiri et al. [1].

To identify manufacturing tolerances, the largest permissible waviness profile with the minimal L_2-norm of surface deformations that could potentially trigger premature transition due to waviness on the airfoil has to be defined. Roughly speaking, the waviness profile mentioned is the one that will induce first transition at a particular position of the chord if compared with other surface deformations with the same L_2-norm. Once this waviness profile is established, manufacturing tolerances can be computed accordingly. This profile is derived through solving an optimization problem utilizing gradient-based methods. As mentioned extensively previously, in this study, the calculation potential of the adjoint method is employed to compute gradients.

To determine the allowable waviness profile, actually, two different optimization problems are solved: an unconstrained optimization employing gradient ascent (GA) method and a constrained optimization using sequential least squares programming (SLSQP). Here, it has been decided to consider two different optimization approaches in order to compare the capability of both methods to find the waviness profile we are interested in. In particular, it is noted that the GA method has the advantage of solving the optimization problem with a good ratio between accuracy and time for computing the solution, despite not achieving the optimal solution. The GA approach, in fact, approximates the solution closely to the optimum. On the other hand, the SLSQP has the peculiarity of finding the optimal solution but with the cost of requiring much

more time to reach convergence.
By definition of the algorithm, SLSQP performs multiple function evaluations in the process of searching for a new direction; SLSQP is time-consuming when compared with the GA, where function evaluation is performed only once between each new gradient computation. Obviously, the comparison between two optimization algorithms has to take into account the computational cost of function evaluation which is commonly quite high in fluid flow simulation, i.e., CFD. In our case, whereas we recur to viscous-inviscid coupled solution and the cost of computations is not extremely high, we can easily notice the difference in time solution. The subsequent sections describe these optimization problems and the algorithms employed to solve them.

6.1 Gradient ascent method (GA)

The methodology presented in the flowchart in Figure 6.1 provides an overview of the approach. The sensitivity of kinetic energy with respect to surface grid points ($\nabla_z E(z)$) for the baseline airfoil is established through the steps delineated in Figure 5.4. Initially, employing NOLOT code to perform stability analysis, the growth history of the most unstable mode over the airfoil is determined, presenting the N-factor curve, which serves as a monitoring metric in the algorithm. Afterward, an appropriate increase in the maximum N-factor value, denoted as ΔN, is determined based on the critical N-value employed in the design of the NLF airfoil. Once ΔN is established, the waviness profile aimed at augmenting the maximum N-factor by ΔN units is obtained by incrementally moving in the direction of the gradient of E. Note that there is no objective function to be minimized in this approach. The GA algorithm employed to find the waviness profile can be expressed as:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \gamma \frac{\nabla_z E(\mathbf{z}_n)}{\max|\nabla_z E(\mathbf{z}_n)|}. \tag{6.1}$$

Here, $\mathbf{z}_n$ represents the vector of the airfoil's surface nodal vertical coordinates at iteration n, and γ denotes a small step size. It should be noted that at each iteration, the step that we take in the direction of the gradient is $\frac{\nabla_z E(\mathbf{z}_n)}{max|\nabla_z E(\mathbf{z}_n)|}$. This results in an optimal deformation in a local sense, as the gradient shows the direction of maximal increase of E for each iteration; however, as will be shown later, such local optima could not lead to an optimal deformation in a global sense after several iterations.
In this algorithm the expected value for ΔN represents both the input and the

convergence criterion.

In section 6.1.1, we describe the process of projecting the gradient onto a specified waviness profile and taking sufficiently small steps in the gradient direction to derive the new waviness profile. Subsequently, the algorithm updates the geometry and grid utilizing the IDWarp package (refer to [17]) to solve all direct and adjoint equations for the newly modified airfoil and determines the new gradient. This iterative process continues until the convergence criterion ΔN is attained.

It is noteworthy that as the airfoil shape changes at each iteration, the most amplified disturbance mode within the boundary layer may also change. Hence, as an option, the algorithm can identify the most unstable mode for the corresponding geometry at each iteration and solve all equations based on the newly updated mode. Moniripiri in his work highlighted that the geometric changes induced during the optimization process are typically very small. Consequently, the parameters of the most unstable modes tend to remain relatively unchanged throughout iterations. Therefore, updating the mode for each iteration is often unnecessary, leading to significant time savings in performing the optimization calculations. Even though this observation might have been predictable, it was necessary to verify it numerically, as it was done.

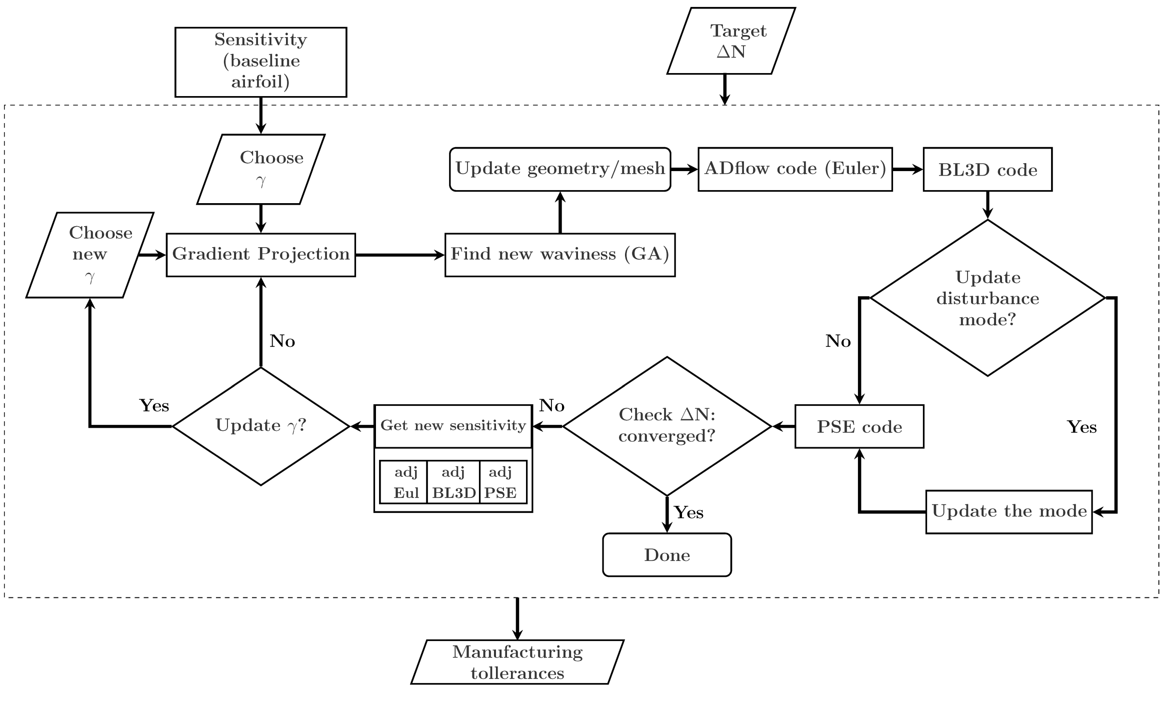


Figure 6.1 Algorithm for finding the waviness profile for a target ΔN using GA method.

6.1.1 Gradient projection

At this point, it's worth noting that the wavelengths of potential waviness on airfoils typically stem from the manufacturing process and the structural composition, particularly with composite panels, along with induced deformations due to the presence of numerous rivets on the surface. Following this observation, it appears logical to introduce a kind of control parameter related to the wavelengths, aiming to constrain the possible deformation of the airfoil's surface. To achieve this, we perform a projection of the gradient onto a specific basis. In this particular approach, at each iteration, we project the gradient onto a basis of k sine Fourier modes, expressed as:

$$(\nabla_z E)_{pr} = \sum_{n=1}^{k} \left[b_n \sin\left(\frac{n\pi(\mathbf{x} - x_{ms})}{x_{me} - x_{ms}} \right) \right], \tag{6.2}$$

where b_n in the coefficient of the n_{th} Fourier mode considered, and it is calculated as

$$b_n = \frac{2}{x_{me} - x_{ms}} \int_{x_{ms}}^{x_{me}} \nabla_z E(x) \sin\left(\frac{n\pi(\mathbf{x} - x_{ms})}{x_{me} - x_{ms}} \right) dx, \tag{6.3}$$

In Equation 6.3, $\mathbf{x}$ is the vector of surface nodal coordinates along the chord.
For the NLF(2)-0415 used in this work, it has been decided to consider $x_{ms} = 8\%$ and $x_{me} = 70\%$ of the chord to discard sensitivity where numerical issues corrupt the results; thus, only the meaningful part of the airfoil is considered. To ensure that displacements at the extremes of the domain $[x_{ms},\, x_{me}]$ are always zero, effectively eliminating forward- or backward-facing steps on the airfoil, only half of the Fourier basis has been chosen. Specifically, the sine basis is utilized, as it is zero at the extremes. After projecting the gradient, we use $(\nabla_z E)_{pr}$ in the algorithm.

6.2 SLSQP method

In this Section, we present how to derive the solution of the optimization problem using the SLSQP algorithm from the SciPy library in Python. The SLSQP algorithm, based on Kraft's work [18], determines a local search direction by solving the second-order local approximation of a cost function that satisfies the constraints [19]. We propose this approach because, as previously highlighted, relying solely on the gradient direction as in GA method does not assure achieving the profile with the minimum

L_2-norm necessary to reach the specified ΔN. To overcome this limitation, we solve a constrained optimization problem to ensure that the solution with the minimum L_2-norm value is obtained. The surface of the airfoil is be parameterized using k sine Fourier bases, which also determines the minimum wavelength of waviness on the surface as

$$\mathbf{z} = \mathbf{z}_0 + \sum_{n=1}^{k} \left[b_n \sin\left(\frac{n\pi(\mathbf{x} - x_{me})}{x_f - x_{ms}} \right) \right]. \tag{6.4}$$

In SLSQP approach the Fourier coefficients, b_n, are considered as design variables and they are found as the solution to the following optimization problem:

$$\begin{aligned} &\min_{\Delta z \in \mathbb{R}} && J = {\Delta z_1}^2 + {\Delta z_2}^2 + ... + {\Delta z_{np}}^2 \\ &\text{with respect to} && b_1, b_2, ..., b_k \\ &\text{subject to} && E = E_{target} = E_R E_0 \end{aligned} \tag{6.5}$$

Here, J represents the objective function, which corresponds to the L_2-norm of the deformations. $\Delta z_i = z_i - z_{0,i}$ denotes the vertical surface deviation of the i_{th} node on the surface. n_p stands for the number of surface grid nodes to be modified. E represents the kinetic energy of disturbances in the boundary layer, while E_{target} denotes the target kinetic energy of disturbance in the boundary layer, which corresponds to a specific ΔN. Additionally, E_0 represents the kinetic energy of disturbances in the boundary layer for the clean baseline airfoil, and E_R signifies the ratio between these two.

To solve (6.5) using SLSQP, it is necessary to compute the gradients of the objective function and constraint with respect to design variables, i.e., $\frac{dJ}{db_n}$ and $\frac{dE}{db_n}$. The gradient $\frac{dJ}{db_n}$ can be derived analytically, employing the chain rule, from the definition of the objective function in (6.5) and the parameterization of the surface (6.4)

$$\frac{dJ}{db_n} = \frac{dJ}{dz}\frac{dz}{db_n}. \tag{6.6}$$

Also, $\frac{dE}{db_n}$ can be calculated using the chain rule from gradients of the kinetic energy of perturbation with respect to surface deformation $\frac{dE}{dz}$, which is obtained from the adjoint method, and $\frac{dz}{db_n}$ as

$$\frac{dE}{db_n} = \frac{dE}{dz}\frac{dz}{db_n}. \tag{6.7}$$

Once $\frac{dJ}{db_n}$ and $\frac{dE}{db_n}$ are calculated, they are passed to the optimizer to find the optimal values for the Fourier modes amplitude, represented by the design variables b_n. Subse-

quently, the waviness profile is reconstructed using Equation 6.4.
One of the major differences between this approach and gradient ascent is that in gradient ascent, the target ΔN serves as both the input and stopping criterion for the algorithm. While the algorithm progresses in the direction of the gradient of the kinetic energy of perturbations, it monitors the value of ΔN at each iteration until the desired value is reached. In the SLSQP approach, however, we cannot directly use the target ΔN as a constraint because we calculate the gradient of kinetic energy, not the N-factor, with respect to surface deformation. Additionally, we need to pass the gradient of the constraint with respect to the design variables to the optimizer. Therefore, we need to establish a correlation between the target ΔN and its corresponding kinetic energy of the domain (E_{target}). This correlation can be determined through trial and error by selecting a specific E_{target} and examining the resulting ΔN, or it can be approximated from the results of the gradient ascent method. In this study, the second approach is feasible since we have the results from the GA method. Therefore, it seems reasonable to pursue this.

6.3 Manufacturing tolerances definition

The waviness profiles obtained through GA and SLSQP algorithms represent the largest allowable profiles, characterized by the minimum L_2-norm of surface deviations when using the second mentioned, which will increase the maximum growth of convective instabilities by a predefined amount, defined as ΔN.
Largest allowable profile with minimum L_2-norm of surface deviations implies that any other waviness profile with the same L_2-norm is not as worse as that case in terms of amplifying the instabilities inside the boundary layer. A definition of the tolerance in terms of the L_2-norm of surface deviations of the largest allowable deformation profile with respect to the clean airfoil is available in the literature, and its form is as follows:

$$h_{\text{tol}} = \sqrt{\frac{1}{(x_{me} - x_{ms})} \int_{x_{ms}}^{x_{me}} (\Delta z_i)^2 \, dx} \tag{6.8}$$

where the surface nodal vertical deformation, calculated using both optimization methods previously presented, for each point in the interval $[x_{ms}, x_{me}]$ is denoted with Δz_i. Obviously, in Equation 6.8, only one flight condition in terms of angle of attack, Reynolds, and Mach number is considered, as well as only one k number of Fourier modes.

Chapter 7

Algorithms for transition delay of BL through geometric optimization

This study aims to define a methodology for shape optimization of an airfoil to delay boundary layer transition, which can also be described as the best-case scenario. The analysis can be conducted starting from any baseline geometry, but it is particularly interesting when applied to a natural laminar flow airfoil.
To define the optimal geometric shape, a methodology very similar to that presented in Chapter 6 is employed, which involves solving a gradient-based optimization problem based on the adjoint equations, as explained earlier. As presented for the tolerance study case, in this study, two problems are solved to determine the optimal shape: an unconstrained problem using the gradient descent (GD) method, and a constrained optimization problem using SLSQP. The reasons for these choices are the same as those presented in Section 6, so we will not repeat ourselves.

7.1 Gradient descent method (GD)

The idea behind the approach is very similar to the one proposed in Section 6.1, as can be seen by comparing the previous scheme reported in Figure 6.1 with the one of the current study depicted in Figure 7.1. After using the NOLOT code for performing stability analysis and determining the growth history of the mode we are interested in, namely the N-factor curve, our aim is to decrease the maximum value of the N-factor to dampen this unstable mode and attempt to delay the transition of the boundary layer. In contrast to what was done for the worst-case scenario, here we aim to decrease the N-factor from its initial value. To adapt the algorithm already implemented for the

other case, we can set a negative value for the target change in N, ΔN. This accounts for the fact that we are examining the best-case scenario.
After selecting ΔN by taking small steps in the opposite direction of the gradient, aiming towards the minimum of the objective function, i.e. kinetic energy E, we can find the optimal shape. Unlike what was presented in Section 6.1, here our goal is to minimize the objective function because it is directly linked to the phenomena that trigger the transition of the boundary layer. The sensitivity of energy with respect to surface grid points, $\nabla_z E$, for the baseline airfoil is determined again using the steps outlined in Figure 5.4.
The perturbed geometry for the purpose of transition delay is determined by:

$$\mathbf{z}_{n+1} = \mathbf{z}_n - \gamma \frac{\nabla_z E(\mathbf{z}_n)}{\max\{|\nabla_z E(\mathbf{z}_n)|\}} \tag{7.1}$$

where the formulation is identical to the one in Equation 6.1, except for the direction in which we are moving, which is defined by the minus sign in front of the step size γ, which is assumed to be positive. In Equation 7.1, n represents the number of iterations. As previously mentioned, since the gradient is calculated for each iteration, indicating the direction of maximum decrease of the kinetic energy for the corresponding iteration, this optimization scheme leads to a locally optimal deformation. Hence, we also propose considering the SLSQP optimization algorithm to aim for a globally optimal solution. The methodology is the same as presented in Section 6.1: subsequent iterations continue until the convergence criterion ΔN is reached, and for each iteration, the IDWarp package is utilized to create a new grid considering the updated airfoil geometry.
Since the geometry changes from one iteration to the next, the stability analysis within the boundary layer could result in a different most amplified disturbance mode. To address this, the NOLOT code can be run for each iteration, considering a different perturbation mode for each iteration.
While the steps mentioned above are largely the same as in the worst-case scenario, it is crucial to note that the main difference between the two studies lies in how the output of the sensitivity analysis is used to update the geometry. This is explained in the following sections.
The projection of the gradient is necessary to smooth the output of the adjoint loop and parameterize it appropriately, both for creating a surface geometry with physical meaning and for ensuring the shape of the final solution can be reproduced during

manufacturing processes. After projection, the gradient information is used to move by a sufficiently small step in the locally optimal direction.

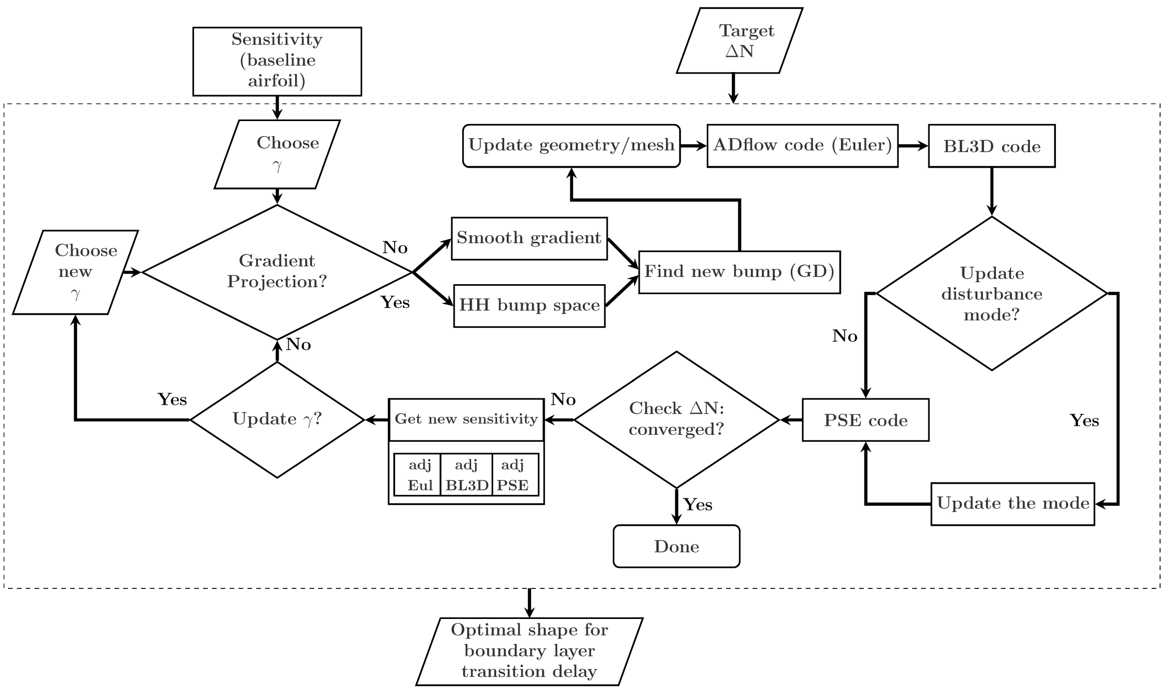


Figure 7.1 Algorithm for finding the surface shape for transition delay of the boundary layer using GD method.

7.1.1 Gradient projection

A powerful technique for shape parameterization in aerodynamic optimization was introduced by Hicks and Henne in the 1970s, [20]. They advocated using a series of smooth functions to deform the initial geometry. The different Hicks and Henne (H-H) functions can be chosen such that only specific regions are refined, while the rest of the object to be optimized remains virtually undisturbed. The main advantage is that fewer design variables are needed to provide an adequate design space. Another benefit compared to the mesh point approach is that the computed gradient always remains smooth. This ensures that the successive surface shapes remain smooth. On the other side, one of the few disadvantage of the these functions is that they are not orthogonal, and they are unable to represent the complete set of continuous functions that vanish at limits of the domain of definition, wheres they tend to zero at the ends. Thus, they do not guarantee that a solution, for example, of the inverse problem for a certain target pressure distribution will necessarily be attained.

Specifically, following the idea of Hicks and Henne, the geometry modification can be described as a weighted sum of smooth sine bump functions as illustrated below:

$$f(\mathbf{x}, a, h, t) = a\left[\sin(\pi\mathbf{x})^{\frac{\log(0.5)}{\log(h)}}\right]^t = af_0(\mathbf{x}, h, t), \tag{7.2}$$

where a, h, and t are the parameters defining the shape of the function itself while $\mathbf{x}$ is the vector of coordinates where the function is defined. Specifically, the above-mentioned parameters control respectively the maximum value of the function, the point in the domain $\mathbf{x}$ where the bump takes its maximum value, and the width of the bump itself. In the more general definition, these parameters are to be understood as vectors, and the n_{th} function obtained by considering the corresponding n_{th} component of the vectors $\mathbf{a}$, $\mathbf{h}$, $\mathbf{t}$, can be rewritten as follows:

$$f_n(\mathbf{x}, a_n, h_n, t_n) = a_n\left[\sin(\pi\mathbf{x})^{\frac{\log(0.5)}{\log(h_n)}}\right]^{t_n} = a_n f_{0,n}(\mathbf{x}, h_n, t_n). \tag{7.3}$$

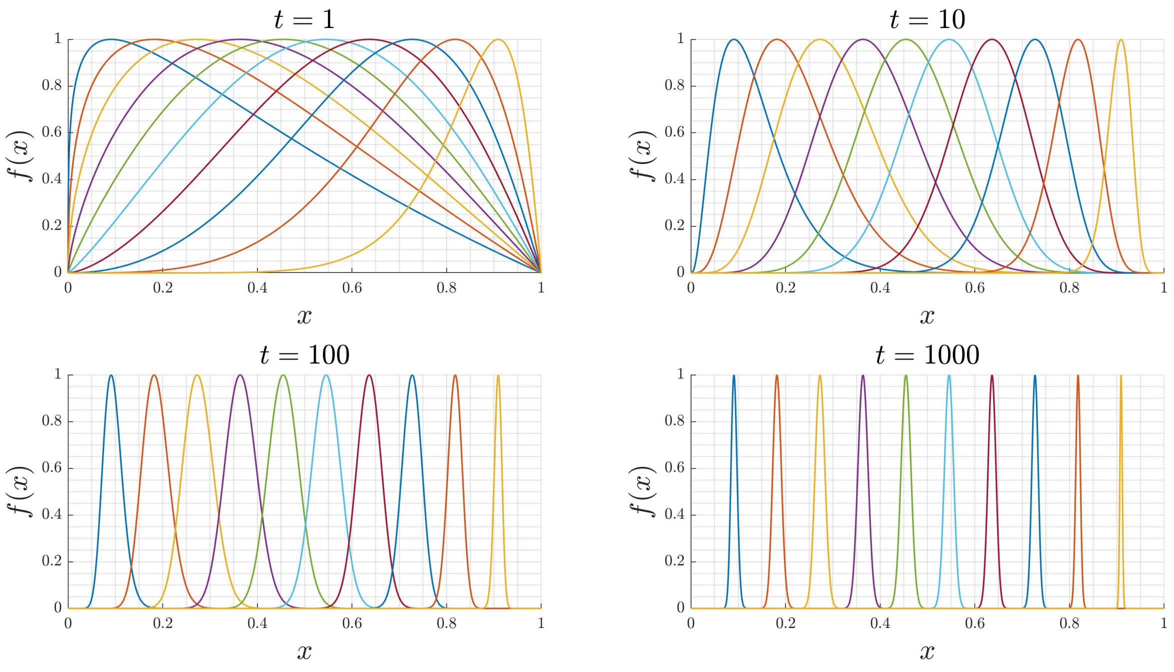


Figure 7.2 Shape of Hicks-Henne bump functions for different values of parameters a, h, and t.

In this case, we can obtain a series of functions that define a space onto which quantities such as the gradient can be projected for parameterization. Thus, for example, in the case presented here, this limits the shapes that the new geometry could assume to some extent. In Figure 7.2, a series of H-H bump functions are shown, illustrating how the function varies with respect to two of the parameters on which these functions depend. As can be seen, the functions exhibit a rather peculiar behavior: by keeping all parameters fixed except one, the shape of the function does not remain invariant. Just for clarity, in Figure 7.2, note that fixing the parameter t, although related to the width of the function, does not strictly constrain it as other parameters vary. This aspect must be taken into account when numerically implementing the problem, especially when it is desired to specify the parameters defining the shape of the function in relation to geometric reference quantities, such as the chord length of the airfoil.

Our goal is to utilize gradient information to define the amplitude of the bump functions, which have been chosen as the basis for sensitivity projection; for this reason, the parameters a_n in Equation 7.3 are set equal to one, essentially assuming a unit maximum value for the H-H functions we are dealing with. The projection of the gradient will provide us with the amplitudes, indicating in a sense the similarity between a space defined by a certain number of chosen bump functions and the shape of the function $\nabla_z E$. The choice of the H-H function space, essentially the dimension of the vectors $\mathbf{a}$, $\mathbf{h}$, $\mathbf{t}$, in a way constrains the problem; for these reason it is required a systematic solution approach which consider different parameter sets in order to evaluate which among these sets effectively represents achieving the condition of optimality (while remembering that in the use of the gradient-based approach, optimally condition intrinsically does not represent the global optimum). Regarding this aspect, to identify the optimal solution, we might be interested in selecting the geometry that minimizes the objective function while simultaneously ensuring the minimum L_2-norm from baseline configuration.

The projection of the gradient onto a basis of k predefined H-H functions is performed as

$$(\nabla_z E)_{pr} = \sum_{n=1}^{k} b_n f_{0,n}(\mathbf{x}, h_n, t_n) = \sum_{n=1}^{k} \left(b_n \left[\sin(\pi \mathbf{x})^{\frac{\log(0.5)}{\log(h_n)}} \right]^{t_n} \right), \tag{7.4}$$

where the coefficient of each bump function, b_n, is calculated as

$$b_n = \int_{x_{ms}}^{x_{me}} \nabla_z E(x) f_{0,n}(\mathbf{x}, h_n, t_n) dx = \int_{x_{ms}}^{x_{me}} \nabla_z E(x) \left[\sin(\pi \mathbf{x})^{\frac{\log(0.5)}{\log(h_n)}} \right]^{t_n} dx. \tag{7.5}$$

In Equations 7.4–7.5, $\mathbf{x}$ denotes the vector of surface nodal coordinates along the chord. The Fourier basis projection carried out in Section 6.1.1 holds advantages over the one performed here using the H-H functions. Firstly, as previously stated, the chosen functions do not become null at the boundaries of the domain, although they approach zero. The issue may arise due to the parameters defining the shape of the H-H functions, which are intended to be specified in terms of the characteristic dimensions of the airfoil profile being optimized, namely the chord length c. However, we aim to modify only a segment of the profile surface, precisely from x_{ms} to x_{me}. This could lead to complications for certain combinations of parameters $\mathbf{h}$, $\mathbf{t}$, especially when the x-coordinate at which the function reaches its maximum is positioned near the domain boundaries $[x_{ms},\, x_{me}]$. This situation might result in a discontinuity between the unaltered and modified segments of the profile, thereby causing convergence issues in solving the equations within the boundary layer using the BL3D code. An additional issue may arise when using an overlap of multiple closely spaced H-H functions; in such cases, sharp convex-shaped regions may be created that, instead of delaying, promote the transition of the boundary layer. In this regard, it has been proposed to consider, in problematic cases, not the overlap of the H-H functions as initially proposed in Equation 7.4, but rather their envelope, ensuring that not only the chosen functions for parameterization are smooth, but also their composition. What explained could be represented by

$$(\nabla_z E)_{pr} = \text{envelope}\left\{\sum_{n=1}^{k} b_n f_{0,n}(\mathbf{x}, h_n, t_n)\right\}. \tag{7.6}$$

Therefore, unlike the Fourier basis projection, it is necessary to manually verify whether the set of parameters for the chosen H-H functions is optimal to ensure that the gradient $\nabla_x z$ in the vicinity of x_{ms} and x_{me} is smooth (as well as the curvature, second derivative) and the global shape is reasonable. These aspects make the generalization of the study to any flow conditions and airfoil geometry considerably complex, essentially due to the behavior that $\nabla_z E$ can exhibit. Deeper studies are required in this regard to find the optimal way to utilize the information from the sensitivity analysis and employ it most effectively to move towards the optimal solution as fast as possible.
As for the tolerance study, for the NLF(2)-0415 airfoil utilized, $x_{ms} = 8\%$ and $x_{me} = 70\%$ of the chord are designated to exclude the front portion where sensitivity is highly spiky, and the rear portion where the BL3D code fails to converge due to separated flow.
Subsequent to the gradient projection, we employ $(\nabla_z E)_{\mathrm{pr}}$ in the algorithm.

7.1.2 Gradient smoothing

In this section, a more rudimentary approach of utilizing the gradient information is presented, which has the advantage of being much more direct but also the disadvantage of not constraining, in a sense, the possible shape of the new profile that will be used in the subsequent iteration of the optimization process until convergence. The proposal is to employ a smoothing function to make the shape of $\nabla_z E$ as physical as possible and apply it directly to define the new geometry. There are several smoothing functions available in the literature; however, it is necessary to consider only those that allow eliminating oscillations while at the same time not altering the overall trend of the sensitivity so as not to modify the information provided by the gradient itself. Some examples of functions available in the literature and already implemented within numerical codes are Savitzky-Golay Filtering, Moving Average Filtering, Local Regression Smoothing (refer to [21]). These functions are particularly flexible as they have various adjustable parameters that regulate how the filtering operation modifies the data compared to the original ones.
For this treatment, an in-house smoothing function has been used, which has the peculiarity of generating a particularly rounded curve without altering the overall trend of the original one. Indeed, it has been observed that this latter one, compared to the previously mentioned functions, allows for a better output. The in-house function is an iterative weighted averaging filter as the smoothing function is applied iteratively for the number of times specified as an input parameter; this function is presented directly in the Section 8.3 dedicated to the numerical implementation of the same.
Again, the region of interest of the profile is included between x_{ms} and x_{me}. At the end of the smoothing operation, we employ $(\nabla_z E)_{\text{smth}}$ in the algorithm.

7.2 SLSQP algorithm

The implementation of the SLSQP algorithm for the study of transition delay exactly overlaps with what was presented in Section 6.2, therefore, to avoid repetition, we refer directly to that section.
Of course, what will change is the parameterization of the airfoil geometry compared to what was proposed in 6.4. It indeed takes the form as follows:

$$\mathbf{z} = \mathbf{z}_0 + \sum_{n=1}^{k} \left(b_n \left[\sin(\pi \mathbf{x})^{\frac{\log(0.5)}{\log(h_n)}} \right]^{t_n} \right). \tag{7.7}$$

The implementation of this algorithm within the SciPy library of Python can indeed be used somewhat as a black box once the principle of operation of the algorithm and how the objective function, design variables, and constraints should be defined are understood. In the study of transition delay, these quantities are entirely analogous to the worst-case scenario; the only difference will be that, since the objective is to dampen the most unstable mode, the final kinetic energy of the system, E, used as a constraint in the optimization problem will be lower than the energy corresponding to the baseline geometry. The design variables will still represent amplitudes, but in this case, they will be those of the chosen H-H functions for the projection instead of the Fourier modes. In conclusion, the objective function, J, remains unchanged, which is the L2-norm of the deformation added to the baseline airfoil. Therefore, it is possible to refer directly to Equation 6.5.

Chapter 8

Code developing

In this Chapter, the codes developed during the project that enabled the implementation of the study are presented. It was decided to include the description of the scripts within the thesis itself rather than solely listing them in the Appendix. This decision was made because writing the scripts consumed a significant portion of the internship period, and therefore they are to be considered as one of the main activities that were carried out.
Specifically, the tasks related to code development required writing scripts in Python and Bash languages. Below is a description of the codes used for the implementation of the optimization algorithm. Scripts are reported in Appendix A.

8.1 Mesh generation

The initial stage of the numerical simulation requires mesh generation to solve the fluid flow equations, specifically the Euler equations, as outlined in Section 1.1. To overcome the limitation of the boundary conditions available in ADflow, in Section 5.1, the need for performing the 2.5D transformation of the baseline profile coordinates was discussed in case a non-zero sweep angle, Λ, was desired. Since the aim of the work was to develop a workflow as general as possible, the possibility was considered that the baseline profile geometry could be provided either in the form of a matrix containing the coordinates (x, y, z) in a `.dat` file or in three-dimensional format within a `.cgns` mesh file. Based on this observation, the algorithm was developed so that the 2.5D transformation is performed automatically depending on the file provided as input containing the geometry.

8.1.1 Construct2D

For mesh generation, the open-source code Construct2D [15] was used as it allows extremely rapid generation of C-grid or O-grid meshes around airfoil profiles. The code was chosen because the quality of the meshes it produces is very good, especially considering that this work deals with the coupled solution of the viscous-inviscid problem, i.e., solving Euler for the flow region outside the boundary layer, therefore extreme grid refinement is not required. Additionally, Construct2D allows the creation of three-dimensional meshes with unit dimension in the span direction, which is suitable for studying the infinite-swept-wing problem intended to be addressed here, considering that ADflow is a 3D solver and thus requires a three-dimensional grid.
In Code A.1, an excerpt of the entire code in Bash language to be execute for optimization studies is provided. In lines `10-22`, the extension of the input file is checked, which, as mentioned, can be either `.dat` or `.cgns`; the file name is provided by the user in the initial section of the complete code, along with all other simulation parameters. In case the format of the input file is different from those allowed, an error message is displayed to interrupt the execution of the algorithm. Depending on the file extension, Code A.2 or Code A.3 is executed for the execution of the coordinates transformation, which ore commented in Sections 8.1.2 and 8.1.3, respectively.
In line `28`, the number of discretization points of the profile surface is counted by counting the number of rows (ignoring possible header rows), as this value is used by Construct2D for mesh generation. Specifically, in lines `29-43`, the configuration files for grid creation are updated; in Appendix B, the above-mentioned files are reported so that their structure and the type of input variables considered can be understood. Here, only some parameters are considered; it is recommended to refer to the code documentation for more details on the other parameters.
The variable `nwke` corresponds to the number of grid nodes in the wake region, in the flow direction, and at line `39`, the number of points on the profile in the `nsfr` variable is taken into account. The definition of these variables is essential for the correct creation of the mesh as the information contained in `info_split.txt` depends on what has been set in the `grid_options.in` file. The number of points in the wake region is determined in the code, considering that the last point on the horizontal wake line departing from the trailing edge (TE) is the first, while the one on the TE is one plus the number of grid points (`nwke`) specified in the `grid_options.in` file. This point can also be considered as part of the lower surface of the airfoil after accounting for the points on the airfoil in counterclockwise order, corresponding to `nwke`+1 plus the

number of points on the airfoil. In the `info_split.txt` file, the last number in the first row represents the final point on the wake line coinciding with the trailing edge (TE), while the last number in the second row indicates the number of points coinciding with the TE as part of the lower surface.
In lines `45-50`, the Construct2D code is executed, taking as input the aforementioned files as well as the file containing the profile coordinates in the normal-to-chord direction, in order to obtain the corresponding mesh, as specified in Section 5.1. An important note to prevent errors: the coordinate file provided to ADflow must necessarily contain a header line (`#`) as the first line, due to the operation of the Construct2D code.
Subsequently, still using the file `info_split.txt`, the mesh file in `.plot3d` format is translated into `.cgns` format and divided into three different zones using the `cgns_utils` tool, so that it can be used in ADflow. The type of boundary condition used during the simulation on each of the six faces of the newly created zones is specified by passing the `info_bc.txt` file, reported in Appendix B; in this file, it is possible to understand the correspondence of each boundary condition with the respective surface of the mesh, considering also that some faces are shared by adjacent regions.

8.1.2 Transformation 2.5D from `.cgns` file

In Code A.2, the Python code for the 2.5D transformation starting from the coordinates provided in the `.cgns` mesh file is presented. Lines `4-18` load the necessary modules for extracting the coordinates from the mesh file and pass the values of the variables required for the transformation, which are specified in the `.sh` script from which the following Python code is launched. To ensure that there is a mesh file inside the targeted folder, the coordinate extraction procedure is enclosed within an if condition; if the file is not found, the execution of the algorithm is terminated. If the file is present, the coordinates are extracted using the appropriate Python function implemented within the ADflow code, called `.getSurfaceCoordinates()`. At this point, the surface coordinates are saved in a file `.dat`, and in lines `45-47`, the scaling factor is applied. It is noteworthy at this stage that for the mesh creation with Construct2D, we need a two-dimensional geometry file, while the function used to extract the coordinates from the mesh object, being generally three-dimensional, provides us with the coordinates of the profile at both ends along the spanwise coordinate. Therefore, to consider only one slice of the mesh in lines `49-52`, only the coordinates corresponding to the minimum spanwise coordinate are considered (at the root); this procedure is correct in the case where the mesh from which the coordinates are extracted refers to a rectangular wing without geometric twist; otherwise, it must be appropriately corrected. Before saving the modified coordinates in the `.dat` file, the second and third columns are swapped; this last step must also be checked, particularly by comparing the lift index variable specified within the ADflow code (in `Coupling*.py` files) with the actual orientation of the mesh.

8.1.3 Transformation 2.5D from `.dat` file

In Code A.3, we're essentially doing the same thing as in Section 8.1.2, but using a `.dat` as input file, containing coordinates organized in columns (x, y, z) in the format of the well-known Xfoil code (refer to [22]), for a wing section only. If the reference profile is available in Xfoil or if one desires to enhance the geometry discretization by increasing the number of nodes on the surface, it is advisable to use the mentioned code as it interpolates the data to create a smoothed geometry. The geometry refinement procedure can also be performed if the input file is in `.cgns` format, as seen in the previous section, since `.dat` files are generated that can be easily read by Xfoil. The structure of Code A.3 is very similar to that of Code A.2, the operations are essentially the same, although different functions are used; therefore, they will not be commented on again.

8.2 Smoothing function

In this section, we comment on Code A.4, written in Python, in which the smoothing function mentioned in Section 7.1.2 is implemented. The smoothing function is designed to be used both to make the C_p and the geometry smoother, as both play a crucial role in the implementation of the adjoint method, which is particularly sensitive to sudden variations in the gradient of these quantities. Naturally, the smoothing operation modifies the original values, so it is necessary to evaluate the introduced variation each time and determine whether it is acceptable or not.
The input file for Code A.4 is `geo.dat_1001`, which contains the solution from ADflow, specifically geometry and C_p in the streamwise direction, as they have been properly subjected to the inverse 2.5D transformation, considering that BL3D requires this format. At lines `20-21`, the coordinates (x, y) are redefined accordingly, as the smoothing function relies on the definition of the curvilinear abscissa s.
The `myfilter` in-house smoothing function is defined between lines `26` and `39`. This

function implements an iterative smoothing filter used to make a series of data, represented by `y`, smoother. The function then executes an outer loop to iterate the smoothing process the number of times specified by the variable `n`. Within this loop, two nested loops are executed to iterate through all the rows and columns of the matrix `y`. For each row of `y`, except for the first and last one, weighting factors `ds1` and `ds2` are calculated using the `s` values in the previous and next rows. The values of the current row of `yout` are updated using a weighted average of neighboring values, with weights based on the factors `ds1` and `ds2`. At the end of the iterations, the function returns the `yout` array containing the smoothed data. In summary, this function applies an iterative process to reduce noise in the data while maintaining the general trend of the original data. The parameter that controls the degree of smoothing applied to the data is represented by the variable `n`. As `n` increases, signifying the number of iterations to perform, the effect of the filter becomes more pronounced, leading to a greater reduction in noise in the data.
The quantity to which the smoothing operation is applied is selected using the variable `smooth_quantity`, which in turn is defined in the input parameters section of the `.sh` code from which Code A.4 is launched.
At the end of the code, in lines `52-56`, the recalculated variable values using `myfilter` are overwritten to the original file.
The function presented in Code A.4 is also suitable for smoothing the gradient operation described in Section 7.1.2. It is well-suited for this task as it minimally perturbs the behavior of $\nabla_z E$, while avoiding the introduction of non-physical oscillations in the resulting geometry. This ensures that the boundary layer development in subsequent iterations is not compromised.

8.3 Sorting function

In this section, we comment on Code A.5, which represents a revision of a function previously used in the algorithm by Moniripiri *et al.*, aiming to make the code more concise and understandable. The function was written by the supervisor Hanifi.
The purpose of Code A.5 is to rearrange the solution file from ADflow, specifically extracting only the solution points from the stagnation point to the trailing edge (TE) of the airfoil. This section of the airfoil is the only one relevant for the execution of the BL3D code to solve the equations within the boundary layer, given the studies we are conducting. As illustrated in Chapter 4, it is necessary to provide the BL3D code with

the pressure distribution C_p and the geometry of this specific region as input data. After loading the Python modules and the necessary variables, in lines `20-21` the last line of the solution file of ADflow is removed because the code considers the TE twice, once as belonging to the upper surface and once to the lower surface. Consequently, it reports the solution twice, which would cause problems in reorganizing the file. In lines `25-27`, the stagnation point is identified, which corresponds to the point where the flow velocity is zero and the pressure becomes maximum. Once the index of the corresponding row of the matrix is known, it is possible to distinguish the upper surface from the lower surface based on this point. Lines `30-47` perform this operation; in particular, it is observed that once the solution is divided into two subgroups, it is necessary to verify that the solution being considered is indeed the one of interest. Specifically, we are interested in the extrados of the profile, so it is sufficient to take the subgroup of the solution in which the geometry increases the value of the z coordinate in the immediate vicinity of the stagnation point while moving in the direction of the flow. From line `50` until the end, the files are saved in the desired formats, with each of them containing in the header the parameters of most interest for the simulation being considered.

8.4 Gradient projection on H-H bump functions

In this section, we present Code A.6, which has been written to implement in Python what is described in Section 7.1.1. Neglecting the initial part where the modules are loaded, in lines `17-18`, `zero_LE` and `norm_TE` are defined, which are subsequently used to define the region for normalizing sensitivity, the output of the adjoint ADflow. In particular, `zero_LE` corresponds to x_{ms} defined in Section 7.1.1 (x_{me} is instead defined in the script `Modify_mesh.py`, which is not shown here, and it is executed after the projection to incorporate the modifications into the previous geometry). The input file is `sens_cut.dat` and contains for each node of the mesh on the profile the corresponding values of $\nabla_z E$, for a fixed spanwise coordinate (similarly to what was done in Section 8.1.2). In lines `25-34`, the H-H function is defined, in a manner entirely analogous to what was presented analytically in Equation 7.3. In the numerical implementation, the parameter names correspond to those reported in Section 7.1.1 except for the vector $\mathbf{x}$, which is renamed as `n`; there is an additional parameter `SF`, which is, however, neglected by setting it to one. The output of the function is the vector `b`, which contains the discretized values of the H-H function for each point of the domain where it is defined,

particularly considering the width.
At this juncture, it is important to recall that the parameters that characterize these types of functions are associated with their width, position of the maximum, and amplitude. However, it has been observed that the relationship between these parameters and the shape of the function is not strictly defined.
Considering this fact, it is then necessary to define a certain methodology to uniquely relate the width to the parameter `t`, specified as a ratio of the chord of the airfoil. Furthermore, it is also noted that the shape of the function varies with the variation of a single parameter while keeping the others fixed. This fact complicates the numerical implementation slightly because it requires discretizing the set of parameters to be considered and through two nested `for` loops, find which combination of discretized parameters comes closest to the desired one. These operations are performed between lines `37-76`; specifically, at line `37`, the step with which to discretize the definition domain of the H-H functions is defined. In lines `38-48`, arrays representing the desired parameters of the H-H function to be used for gradient projection are defined. As explained in Section 7.1.1, it is interesting to consider the sum of different bump functions and evaluate the amplitude of each of them. The implementation in Code A.6 allows for considering any number of H-H functions.
At line `40`, a threshold value is defined to relate the parameter `h` to the actual width of the function. It is assumed that the width of the function is defined at the coordinates where it takes a value equal to 1×10^{-5}, considered sufficiently small to be practically zero. After finding the parameter `h` in lines `65-67`, the error between the set value and the one resulting from the numerical discretization of the space of H-H functions considered is evaluated. It has been verified that the average relative error, in percentage, obtained with this threshold method is always very low, less than 1%.
The outermost `for` loop ends with the definition of each individual H-H using the newly calculated parameters. This operation is repeated for the desired number of functions, which is the length of the vectors `a`, `h`, and `t`. At line `74`, the various H-H functions are collected, each organized into column vectors.
The definition of these functions above can be considered practically continuous since the discretization step is very small, allowing for a smooth definition. From lines `78` to `88`, the H-H function values are calculated for the discretization points of the mesh where the solution has been computed in the codes (ADflow, BL3D and PSE). Interpolation is employed to determine these values.
From lines `90` to `131`, the core of the code is implemented, where the sensitivity is

projected onto the space of previously defined H-H functions. Lines `96-99` implement Equation 7.5 to calculate the amplitude of each function, utilizing the `numpy.trapz` function for integration over the discretized domain, accounting for variable integration steps. This integration process is nested within a `for` loop to accumulate the different amplitude coefficients in the `bn_values` variable.
Subsequent lines, up to `126`, are dedicated to saving the coefficient values into a single `.dat` file, properly labeled for tracking during various optimization iterations. Equation 7.4 is executed between lines `128-131`.
As observed in Section 7.1.1, various methods can be employed to overlap the H-H functions, one of which is considering the envelope.
It is now important to note that the values resulting from the sensitivity analysis are particularly high, around $\pm 10^{7 \div 10}$. Consequently, even projecting the gradient onto the bump functions yields extremely high values. Therefore, normalization is used to scale down the gradient values, making them more manageable and limiting their magnitude, while still preserving the information contained within them. These operations are carried out in lines `134-138`, neglecting sensitivity values outside the region that we want to modify. In conclusion, the rest of the code is dedicated to saving the calculated values in an appropriate format.

Part III

Section Three

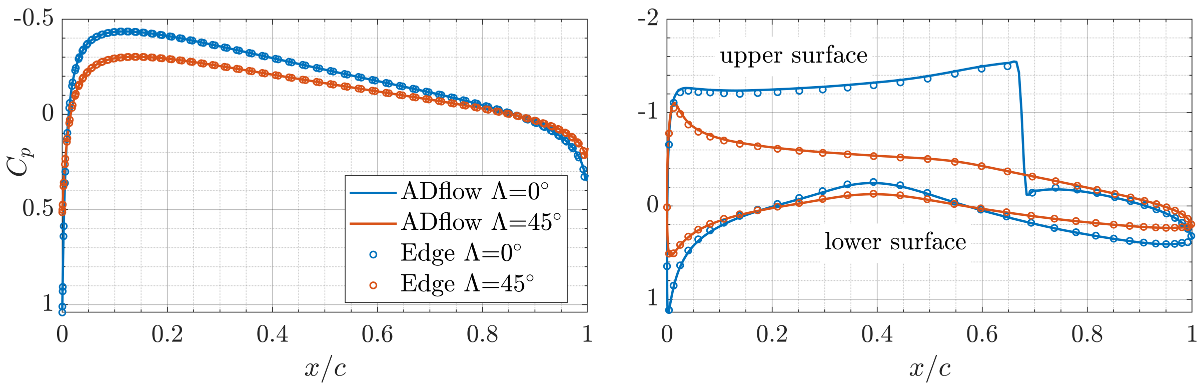
Chapter 9

Algorithm validation

Before proceeding to the presentation of the results, it is necessary to validate the solution algorithm used. The introduction of this Chapter aims to demonstrate the reliability of the output of the various codes, with particular reference to the sections implemented during the development of this work. In addition, some considerations about the limitations of the method used are presented, as well as the issues we encountered during its usage and the way in which these were addressed.

9.1 Validation of 2.5D transformation

In Section 5.1, the need to perform the 2.5D coordinate transformation was introduced to bypass the absence of periodic boundary conditions within the ADflow code. To validate this transformation, two airfoils were considered: specifically, the NACA 0012 and the RAE 2822. For these airfoils, previously validated results were available, considering a non-zero sweep angle Λ, specifically $45°$ and inviscid conditions (Euler solver). The reference results were obtained using the Edge software, which allows setting periodic boundary conditions for the mesh. One might wonder why the studies were not conducted using the aforementioned code; the reasons are related to the desire to utilize an open-source code such as ADflow, along with the adjoint method version implemented within it.
Figures 9.1 show the obtained results; as can be observed, the matching is perfectly achieved. Therefore, the introduced procedure can be considered validated for any airfoil geometry.



(a) Pressure coefficient C_p for the NACA 0012 airfoil profile. Flow conditions: $M = 0.3$, $AoA = 0°$.

(b) Pressure coefficient C_p for the RAE 2822 airfoil profile. Flow conditions: $M = 0.73$, $AoA = 2.79°$.

Figure 9.1 Validation of the 2.5D transformation presented in Section 5.1 by comparing the results obtained with the ADflow code to the reference results obtained with the Edge code, for two different airfoils. The colors of the curves in the left panel match the ones in the right panel. The results reported in the two panels share the following flow conditions: inviscid flow, $p_{\text{ref}} = 40540.2\,\text{Pa}$, $T_{\text{ref}} = 300\,\text{K}$ and $\Lambda = 45°$.

9.2 Grid convergence study

In this section, a convergence study of the mesh is conducted to ensure the meaningfulness of the results obtained from the optimization problem. It is well-known that the result of CFD simulations strongly depends on the number of points used for discretizing the domain of interest, and that increasing the number of points will asymptotically converge towards a certain value that needs to be appropriately validated, for example by resorting to experimental results if available. As the number of points increases, the computational effort also increases, meaning the time required to converge to the solution effectively increases. Therefore, the objective is to identify the number of grid points beyond which it is not sensible to proceed, as it would result in an increase in computational time to achieve the same degree of accuracy. Since an optimization study is being conducted, in this work, in addition to the computational effort of a single simulation, it is necessary to consider that this operation is iterated a certain number of times, so even seemingly negligible reductions in time can be significant in reality.

The sensitivity to the number of discretization points is crucial to consider in this work. The BL3D and NOLOT codes, along with their adjoints, can face convergence challenges if the grid refinement is excessive. This situation might result in non-convergence scenarios. Conversely, when solving the Euler equations for the outer flow, it is essential to avoid geometries with few points and sharp edges. The absence of the boundary layer (inviscid flow) amplifies sensitivity to these issues and flow separation can occur.

Table 9.1 Comparison of lift coefficient C_l and computation time values for four different meshes characterized by the number of points on the specified surface for the NLF(2)-0415 airfoil profile. Flow condition $M = 0.5$, and $\Lambda = 45°$.

N. surface grid points	**798**	**998**	**1198**	**1598**
C_l	1.0591	1.0592	1.0593	1.0594
Time $[sec]$	741	746	784	825

In Table 9.1, the values of the lift coefficient C_l and the computational time for four different meshes for the NLF(2)-0415 airfoil profile under the same flow condition M=0.5, $\Lambda = 45°$ are reported. The outer dimensions of the mesh were kept the same (refer to Chapter 5 and Figure 5.1a), as the issues that could arise concern the BL3D, NOLOT and PSE codes, which are sensitive to the number of discretization points of the profile and not the grid points of the whole considered volume. As can be observed, by increasing the values of C_l, numerical convergence is achieved; in any case, the variation is significantly small, considering only the 4^{th} decimal digit is changing. In Figure 9.2, the pressure distribution C_p for the four considered meshes is reported; since C_l essentially derives from the integration of the pressure coefficient C_p, it is observed, as mentioned earlier, that the different curves are perfectly overlapping, suggesting that convergence has been reached. In Figure 9.3a, the first and second derivatives of the coordinate z with respect to the coordinate x are reported to highlight some of the characteristics that guided the choice of the number of grid points to be used in the optimization studies carried out. As can be observed, the first derivative is practically insensitive to the number of discretization points considered; actually in the case of low number of discretization points near LE the magnitude tend to increase a lot because of rough definition of curvature. It is important to note the behavior of the curvature, i.e. the second derivative, as it plays a significant role in the codes used for boundary layer stability analysis. In this regard, it is worth mentioning that within the NOLOT code, a curvature correction function has been utilized, which takes into account the deviation of the abscissa traced on the surface of the profile from the straight abscissa, as well as the tendency for the dimensions of the sides of the cells to increase when moving away from the airfoil along the normal direction.

A completely analogous reasoning can be made regarding C_p; in Figure 9.3b, the respective values for the four different mesh configurations just mentioned are shown. It is essential to consider the sensitivity of the pressure coefficient (and its derivatives) to the number of discretization points, as the input of the codes responsible for solving the equations within the boundary layer is the geometry of the airfoil profile as well as the C_p coefficient. Therefore, in order for the adjoint solution to be manageable, i.e. smooth behaviour, it is necessary to control both parameters.

Considering the results reported, the mesh with 1198 grid points on the surface was chosen as it proved to be the one with the best compromise between accuracy, computational times, and sensitivity of geometry and pressure coefficient derivatives to the number of points. Despite the good trend shown in Figure 9.3b, to ensure sufficiently smooth first and second derivatives of the pressure coefficient, the smoothing function described in Section 8.2 was employed; the variation of the results following the introduction of the aforementioned function was controlled and validated.

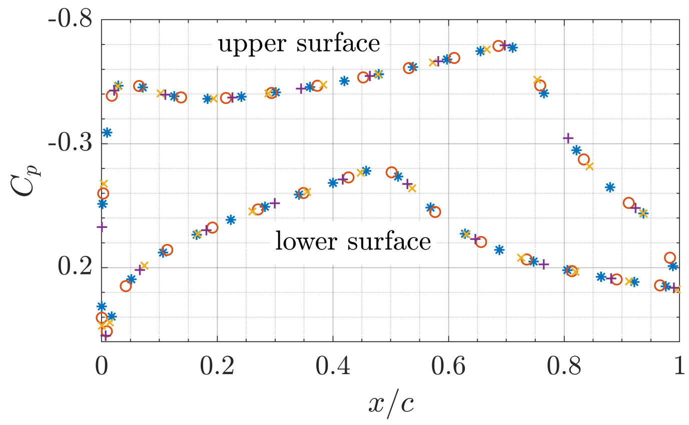
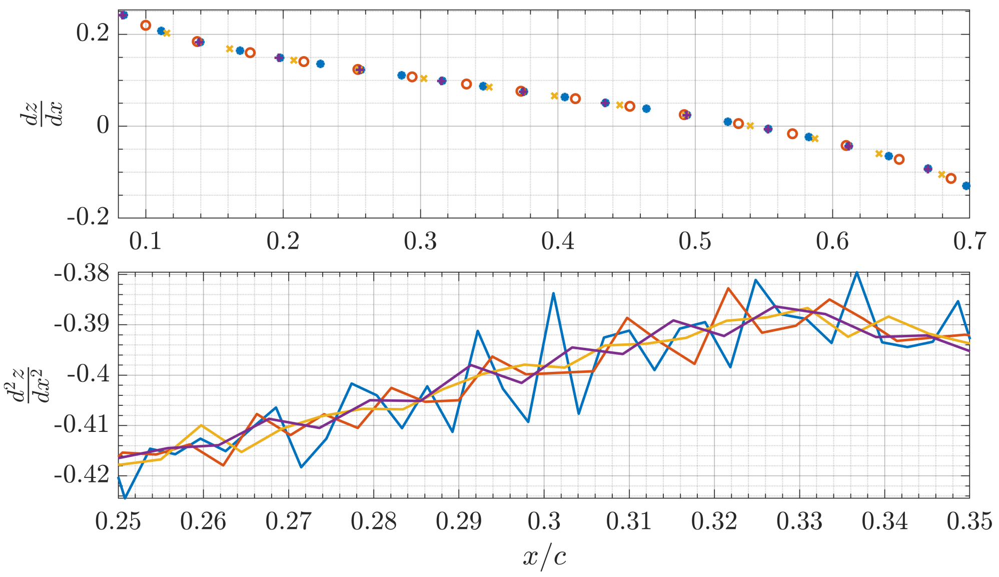
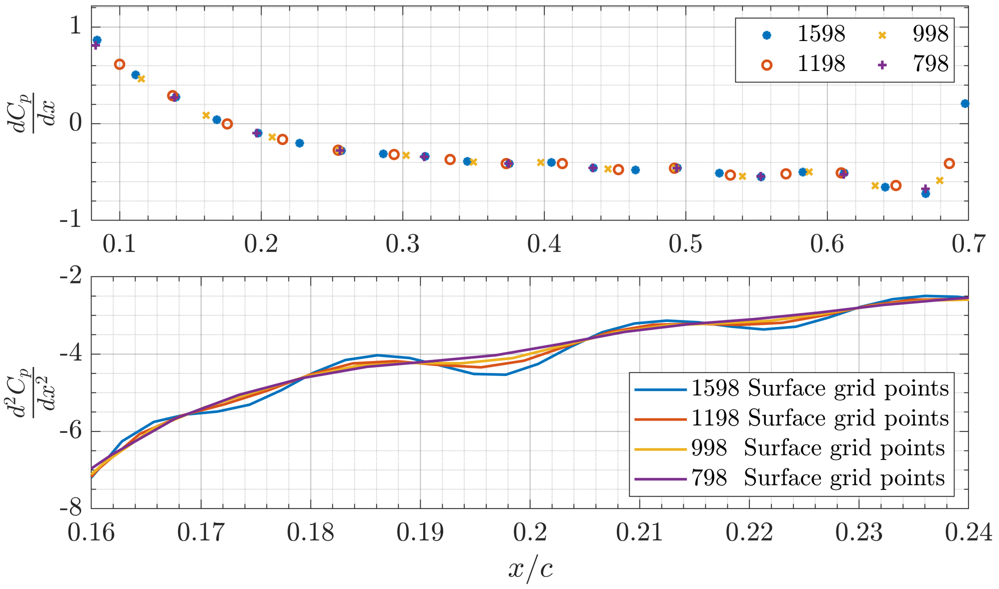


Figure 9.2 Pressure coefficient C_p for the four meshes used. Flow conditions $M = 0.5$, $\Lambda = 45°$, and NLF(2)-0415 airfoil profile. Colours match with the legend in Figures 9.3. For each case only one result is reported every ten, for better readability.



(a) In the top panel, the variation of the first derivative of the upper geometry, $\frac{dz}{dx}$, of the NLF(2)-0415 airfoil profile is shown as a function of the number of discretization points considered. In the bottom panel, the second derivative or curvature of the airfoil geometry, $\frac{d^2z}{dx^2}$, is reported for a limited section of the chord, highlighting the spiky behavior.



(b) In the top panel, the variation of the first derivative of the upper pressure distribution, $\frac{dC_p}{dx}$, of the NLF(2)-0415 airfoil profile is shown as a function of the number of discretization points considered. In the bottom panel, the second derivative of C_p, $\frac{d^2C_p}{dx^2}$, is reported for a limited section of the chord, highlighting the wavy trend.

Figure 9.3 Sensitivity analysis of the first and second derivatives of the NLF(2)-0415 airfoil geometry and the pressure coefficient C_p concerning the refinement of the mesh used for optimization calculations.

9.3 Gradient validation

9.3.1 Numerical instabilities in $\nabla_{p_e} E$ computation

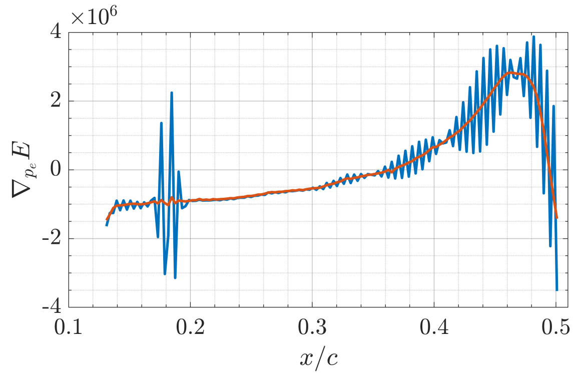
It is well-known that the Parabolized Stability Equations (PSE) are not fully parabolic equations, see Haj-Hariri [23] and an elliptic behavior, coming from the gradient of the disturbance pressure, still exists in the equations. Consequently, if the step size in the streamwise direction is too small, the solution will show strong oscillations and eventually diverge. Li and Malik [24] presented a limit for the step-size restriction, valid when a first-order backward Euler scheme is used. They also showed that dropping the streamwise derivative of pressure in the primitive variable formulation[1], and the derivative of streamwise wavenumber in the stream-function formulation, reduces the step-size restriction considerably. However, this would not remove the ellipticity completely and dropping these terms can affect accuracy of solution for some flows. Andersson *et al.* [25], showed that by adding a term proportional to the truncation error to the first-order backward Euler scheme one can remove the step-size restriction. However, in some crossflow cases the multiplying factor may become too big and solution may be affected.
In some cases of crossflow perturbations studied here, we observed that though for a given streamwise step-size the growth rates did not show any oscillatory behavior, the gradient $\nabla_{p_e} E$ showed a strong oscillatory behavior. We found that by removing the terms originated from $\frac{\partial \hat{p}}{\partial x}$ in PSE and their adjoint we could remove these oscillations while the stability results remain unaffected. Note that in the current work, we use density instead of pressure as one of primitive variable. This means that

$$\frac{\partial \hat{p}}{\partial x} = \frac{1}{\gamma M^2} \frac{\partial}{\partial x} \left(\hat{\rho}\, \bar{T} + \bar{\rho}\, \hat{T} \right). \tag{9.1}$$

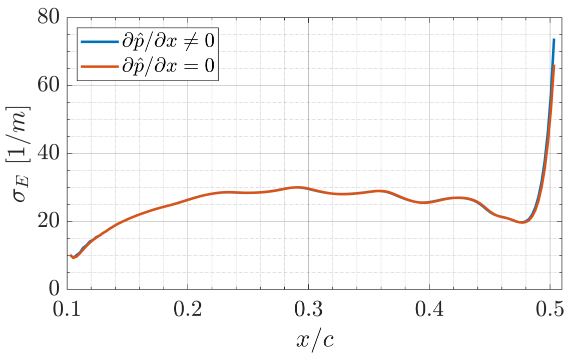
Figures 9.4 show an example, where results for a crossflow mode with $\beta = 3611.03\ m^{-1}$ and $f = 682.690\ Hz$ is presented.

Since a smooth behavior of $\nabla_{p_e} E$ is crucial for obtaining a usable sensitivity of kinetic energy with respect to design variables, $\nabla_z E$, in order to reduce its variability in slope, we chose to apply to it the smoothing function already presented in Section 8.2. In Figures 9.5, the trends of the energy gradient with respect to the two quantities

[1]For incompressible flows, the governing equations may be represented either in primitive variables or by using other formulations obtained by eliminating the pressure gradient (e.g., vorticity-streamfunction formulation). On the other hand, for compressible flows, primitive variables offer a natural and the only choice.



(a) $\nabla_{p_e} E$ as a function of the dimensionless streamwise coordinate, considering both the case where the term $\frac{\partial \hat{p}}{\partial x}$ is neglected and where it is included. Strong numerical oscillations are highlighted for $\frac{\partial \hat{p}}{\partial x} \neq 0$.



(b) σ_E as a function of the dimensionless streamwise coordinate; the two overlaid cases match perfectly, indicating that $\frac{\partial \hat{p}}{\partial x} = 0$ can be considered, thus circumventing instability issues in gradient computation.

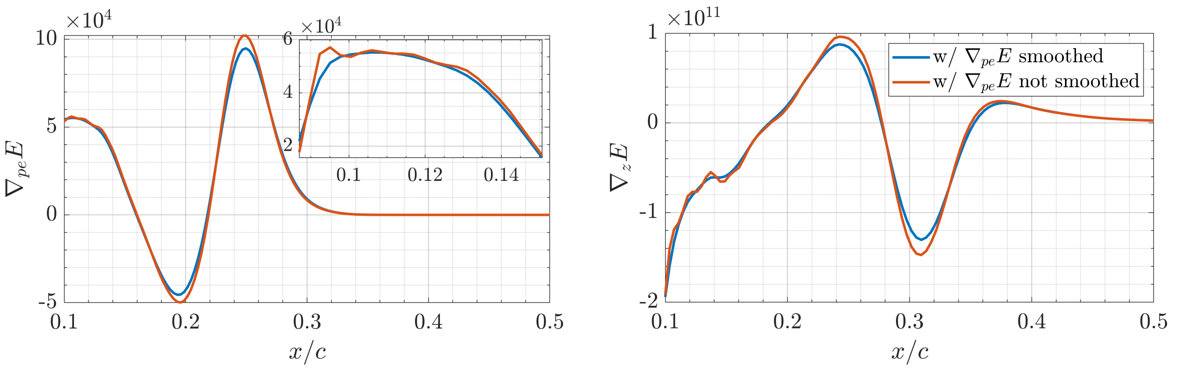
Figure 9.4 Validation of the approximation introduced in the calculation of the gradient $\nabla_{p_e} E$ by setting $\frac{\partial \hat{p}}{\partial x}$ to avoid numerical oscillation issues in solving the PSE using the NOLOT code. The colors on the left panel correspond to those in the legend of the right panel.

of interest are shown, before and after smoothing. The trends we are referring to were obtained after a certain number of iterations. The parameters of the simulations corresponding to the mentioned results are not important in this context, as the trend we want to comment on here is entirely generalizable, having occurred in each of the studied cases. As observable, despite a minimal variation in the curve after smoothing (with a consequent reduction in variability), there is a significant improvement in the trend of $\nabla_z E$. It is important to note that the application of the selected smoothing function, although slightly modifying the numerical values treated, does not alter the overall trend and therefore does not corrupt the results obtained from the sensitivity analysis; if this had not occurred, it would certainly not have been correct to consider it.

The latter operation clearly holds more relevance in the context of studying transition delay rather than tolerance study; in the former scenario, as outlined in the implementation detailed in Section 8.4, correctly positioning the bumps based on the values assumed by $\nabla_z E$ is crucial. Otherwise, the result may be exactly the opposite of what is expected. Thus, having regions of x/c where the sign of $\nabla_z E$ is clear becomes important.

Conversely, in the case of a tolerance study, this consideration holds less significance.

Here, sensitivity is projected onto the left-hand side of the Fourier basis, which is periodic. Consequently, only the amplitude is sensitive to the value assumed by the sensitivity at various distributed points (rather than at a localized one), while the sign is imposed by the sine function itself.
In simpler terms, when projecting sensitivity onto the space of H-H bump functions, as in transition delay studies, the computation of the amplitude of the bumps considers the entire $\nabla_z E$ profile. However, the sign of each bump is strongly dependent on the value of $\nabla_z E$ where the maximum has been positioned (the position of the maximum, i.e. the sign, is a constraint as it is an input of the optimization process).



(a) $\nabla_{p_e} E$ as a function of x/c before and after the smoothing operation. A zoomed-in view is provided for a certain range of x/c to highlight the oscillations responsible for undulations in $\nabla_z E$.

(b) $\nabla_z E$ as a function of x/c before and after applying the smoothing operation to $\nabla_{p_e} E$. The improvement in shape is easily noticeable, as well as the fact that the general trend remains unchanged.

Figure 9.5 Effect of smoothing operation on $\nabla_{p_e} E$ on the behavior of $\nabla_z E$. The colors on the left panel match the ones in the label reported in rigth panel.

9.3.2 Gradient validation

For validating the gradient calculation, to alleviate computational cost, we opted to validate under the same conditions employed by Moniripiri *et al.* in [1]. This approach allowed us to cross-verify their results while leveraging the outcomes of finite difference calculations which were employed for validation. Hence, for gradient validation purposes, we selected the fluid flow conditions and the unstable mode with parameters as detailed in Table 9.2.
To validate the implementation of the adjoint method, we conducted a comparison between gradients obtained from the adjoint method and those computed using a

central finite difference scheme, as described by:

$$\left(\frac{\partial J}{\partial g_k}\right)_{FD} \approx \frac{J(\mathbf{q}(\mathbf{z}+\Delta\mathbf{z}),\mathbf{z}+\Delta\mathbf{z}) - J(\mathbf{q}(\mathbf{z}-\Delta\mathbf{z}),\mathbf{z}-\Delta\mathbf{z})}{2\Delta g_k}. \tag{9.2}$$

Here, J represents the objective function, $\mathbf{q}$ is the state vector, $\mathbf{z}$ denotes the surface nodal coordinate vector of the baseline airfoil, g_k is the k_{th} design variable, and $\Delta\mathbf{z}$ is the vector of surface deformation resulting from Δg_k. To compute the gradients using the finite difference method (FD), in Equation 9.2, we must specify g_k and the corresponding $\Delta\mathbf{z}$. As mentioned earlier, the design variables in the adjoint method correspond to all surface grid points. Hence, to compare the gradients from both methods, selecting the same design variable for FD, i.e., perturbing one point of the surface mesh grid at a time ($g_k = z_k$ and $\Delta\mathbf{z} = \Delta g_k \hat{e}_z$), appears to be a reasonable choice. However, this approach often leads to sharp edges in the mesh, resulting in sudden changes in pressure distribution and consequently yielding inaccurate and noisy gradients. To avoid this problem, it is possible to parameterize the surface of the airfoil as

$$\mathbf{z} = \mathbf{z}_0 + \sum_{i=1}^{N} a_i f_{0,i}(\mathbf{x}, h, t) \tag{9.3}$$

where $f_{0,i}(\mathbf{x}, h, t)$ represent the Hicks–Henne bump functions, as presented in Equation 7.2 (refer to the respective section for the definition of parameters), $\mathbf{z}$ and $\mathbf{z}_0$ denote the baseline and perturbed geometry, respectively. The advantage of using the H-H functions, as mentioned in Section 7.1.1, is related to the smoothness of the functions themselves. Since our objective is to use FD to find the gradients and compare them with the ones obtained with the adjoint method, we have chosen to consider the maximum height of the bump, a, as the design variable. Therefore, in the Equation 9.2 it is assumed $g_k = a_k$ and $\Delta\mathbf{z} = \Delta a_k f_{0,k}(\mathbf{x}, h, t)\hat{e}_z$.

In order to calculate the gradient using FD in each of the surface nodal points the parameter h has to be used to move the position of the bump along the surface; for each position of the bump, the geometry and grid should be updated. To perform this,

Table 9.2 Flow conditions and parameters (frequency, spanwise, and initial streamwise wave numbers) of the instability mode used for the gradient validation procedure calculated with the adjoint method.

Re	M	AoA [°]	Λ [°]	f $[Hz]$	β $[m^{-1}]$	$\alpha_{x_{ms}}$ $[m^{-1}]$
6×10^6	0.5	1.25	0	5850	0	600

the inverse distance method implemented in the IDWarp package has been used.
When using FD, one of the main challenges is to select a suitable step size to achieve a good compromise in terms of truncation and cancellation errors. This aspect has been taken into account in the work of Miniripiri *et al.*, where they calculated the gradient using FD for x/c ranging from 0.15 to 0.60, using a fixed value for the parameter that controls the width of the bumps and repeated the computation for four different step sizes (Δa) in the range of 10^{-7} to 10^{-4}. They observed that FD convergence was achieved for $\Delta a = 10^{-5}$. The next step involves calculating the sensitivity of the objective function, namely E, with respect to the same design variable as used in FD, i.e., the height a of the H-H function. This can be done by computing an inner product between the gradient and the surface deformation induced by the H-H functions centered at each surface grid point in turn, as described by the Equation below:

$$\left(\frac{\partial J}{\partial a_i}\right)_{\text{Adjoint}} = \int \nabla_z J f_{0,i}(\mathbf{x}, h, t)\, dx. \tag{9.4}$$

In Equation 9.4, $f_{0,i}(\mathbf{x}, h, t)$ represents the bump function with the maximum value located at the i_{th} grid point in x.
The comparison of $\nabla_z E$ obtained from the adjoint method and central finite difference is shown in Figure 9.6; the good agreement between the gradients allows us to validate the implementation of the adjoint method.

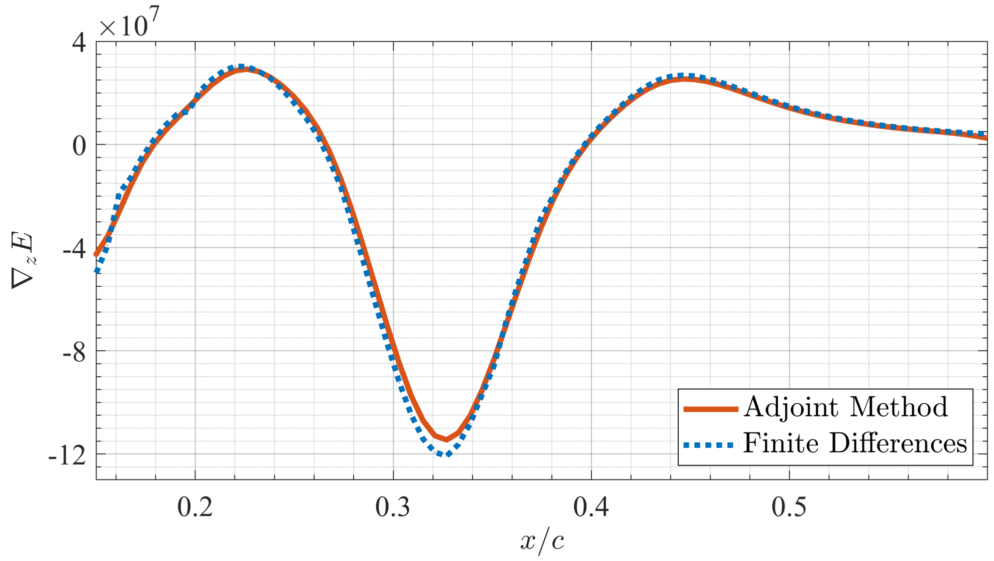


Figure 9.6 Comparison of the gradient obtained using the adjoint and central finite differences methods for the upper surface of the NLF(2)-0415 airfoil. The flow and instability mode conditions considered are those reported in Table 9.2.

9.3.3 Limitations of adjoint-based approximations

Despite the positive outcome of the validation procedure presented in the previous Section 9.3, [1] introduces a limitation that must be considered in the application of the adjoint method, failing which may lead to poor accuracy of the results.
Considering the kinetic energy of perturbations, denoted as E, if we aim to approximate the change (ΔE) within the boundary layer resulting from a smooth surface deformation on an airfoil, we can resort to a Taylor expansion around the baseline airfoil:

$$\Delta E = E(\mathbf{z}_0 + \Delta\mathbf{z}) - E(\mathbf{z}_0) \approx \frac{\partial E}{\partial \mathbf{z}}\Delta\mathbf{z} + \frac{1}{2}(\Delta\mathbf{z})^T \frac{\partial^2 E}{\partial \mathbf{z}^2}(\Delta\mathbf{z}). \tag{9.5}$$

In this equation, $\mathbf{z}_0$ is the clean airfoil surface nodal coordinates vector and $\Delta\mathbf{z}$ the vector of surface deformation. The adjoint method allows us to obtain in an advantageous ways the term $\frac{\partial E}{\partial \mathbf{z}}$ in the Taylor series representing the first-order term in Equation 9.5. By doing so, higher-order terms beyond the first are neglected, which for some conditions, however, play a significant role. In particular, in the work of Moniripiri *et al.*, considering two different bump functions, it is observed that as the height of the bump is increased, the deviation between the exact and approximated solution increases. This is due to the fact that the linear approximation is accurate only when the deformation is sufficiently small, therefore when the geometry deformation does not induce a change in mean flow and pressure abruptly in the boundary layer, i.e., avoiding conditions of stronger nonlinearity. In the study mentioned above, two cases with bumps of different widths were also investigated. The exact final energy and ΔE were calculated solving Euler, BLE, and PSE using the new deformed geometries. These results were then compared with those obtained using a first-order approximation based on Equation 9.5, along with the gradient of the baseline airfoil obtained from the adjoint method. As already mentioned, increasing the height of the bump increases the deviation between the exact and approximated solution, but if the bump is wider, the nonlinearity effects are less significant. It can therefore be concluded that in the case of larger and narrower deformations, second-order derivatives $\frac{\partial^2 E}{\partial z^2}$ have to be considered. However, this requires significant computational effort as it involves computing the elements of the Hessian matrix[2]. Therefore, to avoid introducing second-order terms, it was chosen to solve optimization problems introducing small deformations to avoid nonlinearity effects, albeit at the cost of a greater number of iterations.

[2]Square matrix containing second-order derivatives of the objective function with respect to design variables, i.e., the coordinates z_i.

9.3.4 Choice of step size γ for the gradient approaches

A crucial aspect when employing gradient approaches is selecting the step size appropriately, denoted here as γ. If γ is chosen too small, convergence will be achieved but at a sluggish pace; conversely, selecting γ too large can lead to convergence issues due to the potential influence of higher-order terms in the Taylor series expansion of E, resulting in deviations from the steepest ascent direction.

To determine the optimal step size, one should compare the final wavy profiles obtained using different values of γ under identical flow conditions, instability mode, and final ΔN values. Despite considering the same ΔN value for each, employing different γ values will yield distinct L_2-norm values; particularly increasing the step size the L_2-norm will increase as well. Such an analysis was conducted in [1], and drawing from its findings, $\gamma = 6 \times 10^{-6}$ m was selected as it reliably ensures convergence to a consistent solution. Consequently, it is utilized in all gradient ascent calculations.

Chapter 10

Results for tolerances study

The results for the study of dimensional tolerances were obtained following a parametric approach, considering different flow conditions to identify trends useful in the design process. The results were obtained considering various Mach numbers, Reynolds numbers, and angles of attack, while keeping the thermodynamic properties fixed and equal to those at a flight altitude of 9600 meters, as already mentioned in Chapter 5. For the stability analysis, the most unstable mode was identified for each of the cases considered, particularly for the baseline profile NLF(2)-0415, among a wide range of different feasible modes. In this study, considering the case of a swept wing and having access to the results obtained by Moniripiri *et al.* in their study [1] for zero sweep angle, it was observed that in the former case, the number of instability modes is significantly higher due to the coexistence of Tollmien-Schlichting (TS), pseudo-cross-flow, and cross-flow modes. This fact has complicated the selection of the unstable mode to consider since several of them were eligible to be elected as the most critical. Furthermore, the impossibility of assuming the boundary layer transition at a well-defined critical N-factor value, given the high variability of the phenomenon, also complicated the mode selection. It has already been mentioned that in some cases, a critical N value of 9 can be assumed as a reference, but at the same time, various experimental results have shown that this value can be considerably lower or higher depending on specific conditions (geometric shape of the profile and flow conditions). Following this necessary observation, it is therefore considered appropriate to make the following disclaimer: the modes chosen for each flow condition were selected based on observations and comparisons, but this does not exclude the possibility that there may be a mode that has more pronounced effects on ΔN among the practically infinite number of possible cases. Tables 10.1–10.2–10.3 report the values of Reynolds number (Re), angle of attack

(AoA), and parameters for stability analysis, including the frequency (f), spanwise wave number (β), and the initial value for the complex-value streamwise wave number ($\alpha = \alpha_r + i\alpha_i$) of the disturbance mode selected. All the results reported below were obtained for a NLF(2)-0415 airfoil profile with a sweep angle condition Λ of 45°, unless specifically indicated otherwise.

Table 10.1 Parameters used for analysis of different cases. Reynolds number is considered as a variable parameter while others are fixed.

$AoA = 1.25°$, $M = 0.5$

$Re(\times 10^6)$		**9**	**12**	**15**
f	$[Hz]$	6066.15	9761.95	9106.24
β	$[m^{-1}]$	682.69	608.06	760.26
α_r	$[m^{-1}]$	348.92	878.43	736.33
α_i	$[m^{-1}]$	-2.09	-22.63	-18.38

Table 10.2 Parameters used for analysis of different cases. Angle of attack is considered as a variable parameter while others are fixed.

$Re = 15 \times 10^6$, $M = 0.5$

$AoA[°]$		**-1.00**	**-0.50**	**0.50**	**1.25**	**1.50**	**1.75**
f	$[Hz]$	6666.22	5650.49	7833.74	9106.24	15515.70	17526.20
β	$[m^{-1}]$	2295.65	2300.24	1138.26	760.26	947.09	1052.75
α_r	$[m^{-1}]$	-1012.32	-1074.54	224.31	736.33	1295.35	1394.85
α_i	$[m^{-1}]$	-28.64	-23.69	1.62	-18.38	-1.81	-2.28

Table 10.3 Parameters used for analysis of different cases. Mach number is considered as a variable parameter while others are fixed.

$Re = 15 \times 10^6$, $AoA = 1.25°$

M		**0.45**	**0.50**	**0.55**	**0.60**
f	$[Hz]$	10646.60	9106.24	10165.20	9709.12
β	$[m^{-1}]$	1216.96	760.26	754.23	840.66
α_r	$[m^{-1}]$	722.87	736.33	726.43	495.35
α_i	$[m^{-1}]$	4.04	-18.38	-19.50	-13.40

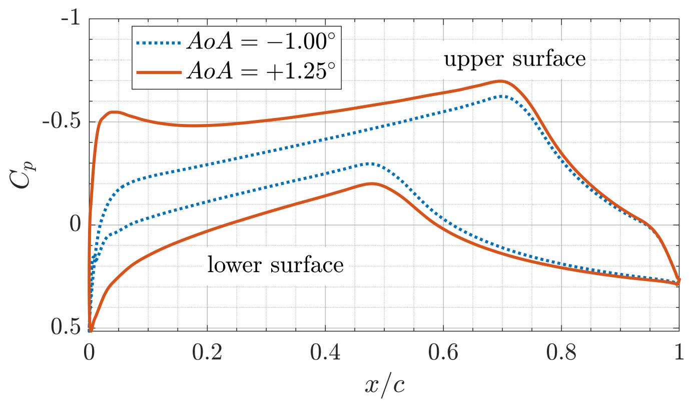
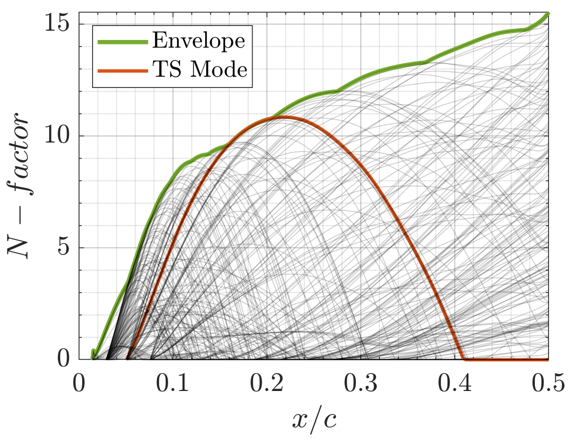


Figure 10.1 Pressure coefficient C_p for $Re = 15 \times 10^6$, $M = 0.5$ for two angles of attack as indicated in the legend. It is observed in particular how the regions of APG and FPG vary as a function of the angle of attack.

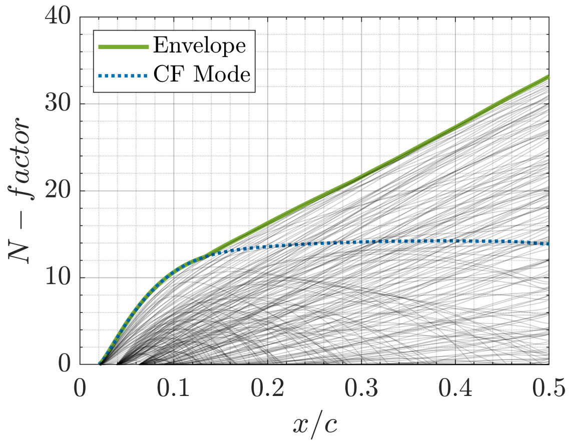
10.1 Types of instabilities

As mentioned, the consideration of flow conditions with a non-zero sweep angle complicates the process of selecting the most unstable mode due to the absence of a clear predominant mode, given the coexistence of Tollmien-Schlichting (TS) and crossflow modes (CF). However, it is known from the literature that, depending on the flow conditions considered, TS or crossflow instabilities may be predominant. In particular, the parameter that plays a fundamental role in this regard is the angle of attack. TS instabilities tend to dampen in the case of favorable pressure gradient (FPG), while crossflow instabilities tend to dampen in the presence of adverse pressure gradient (APG). The distinction between the two types of instability can be made qualitatively by observing the behavior of the N-factor as a function of the streamwise coordinate, x, or quantitatively by comparing the value of the wave angle, ψ, with a reference value. The shape of the N-factor curve for TS instabilities is characterized by a downward concave parabolic trend, with the maximum point usually located in the advanced region of the airfoil; these instabilities tend to increase their N-factor value before decaying due to FPG. In the case of crossflow instabilities, on the other hand, the N-factor typically has a monotonically increasing trend and therefore reaches its maximum value at the end of the domain for which the stability analysis is performed. From the literature, a reference wave angle value ψ of around 70° to 75° is considered. Below this range, the instability is classified as TS type, and above it, it is classified as crossflow type.

In Figure 10.1 is shown the variation of the pressure coefficient C_p for two flow con-



(a) N-factor curves for the different instability modes and envelope curve for the baseline airfoil as a function of the x/c coordinate. The most unstable TS mode has been highlighted. Flow conditions: $Re = 15 \times 10^6$, $M = 0.5$, $\Lambda = 45°$, and $AoA = 1.25°$.



(b) N-factor curves for the different instability modes and envelope curve for the baseline airfoil as a function of the x/c coordinate. The most unstable CF mode has been highlighted. Flow conditions: $Re = 15 \times 10^6$, $M = 0.5$, $\Lambda = 45°$, and $AoA = -1.00°$.

Figure 10.2 Comparison of the N-factor trends and envelope curves for two different flow conditions to highlight and distinguish Tollmien-Schlichting (TS) and crossflow (CF) instabilities. For the parameters of the highlighted instabilities, refer to Table 10.2.

ditions characterized by the same $Re = 15 \times 10^6$, $M = 0.5$ and sweep angle, Λ, but different AoA ([1.25°, -1.00°]). As expected, it is observed that for the condition with higher angle of attack, i.e., 1.25°, a region of APG is present near the leading edge, and consequently, the TS-type modes (characterized by a similarly parabolic N-factor), as shown in the left panel of Figure 10.2, are predominant. In the right panel CF modes are dominant due to the absence of adverse pressure gradient; in the first part of the upper surface the pressure decreases constantly from the stagnation point and with it the N-factor of cross flow modes tends to increase till the coordinate where the code is solving, i.e. before separation point (note that separation is before reaching the maximum value of C_p). The results just mentioned in Figure 10.2 are outputs of NOLOT code.

For the two highlighted instability modes, which have been considered for the aforementioned flow conditions as most unstable modes for tolerances study, the profile of the streamwise and spanwise velocity components for three different coordinate values are shown in Figure 10.3. The analysis of velocity profiles is important because it is necessary to verify that the modes being considered are indeed related to the physics and are not spurious, i.e., they do not exhibit strange oscillations determined by purely numerical factors. As can be observed, the treated modes have a particularly smooth variation.

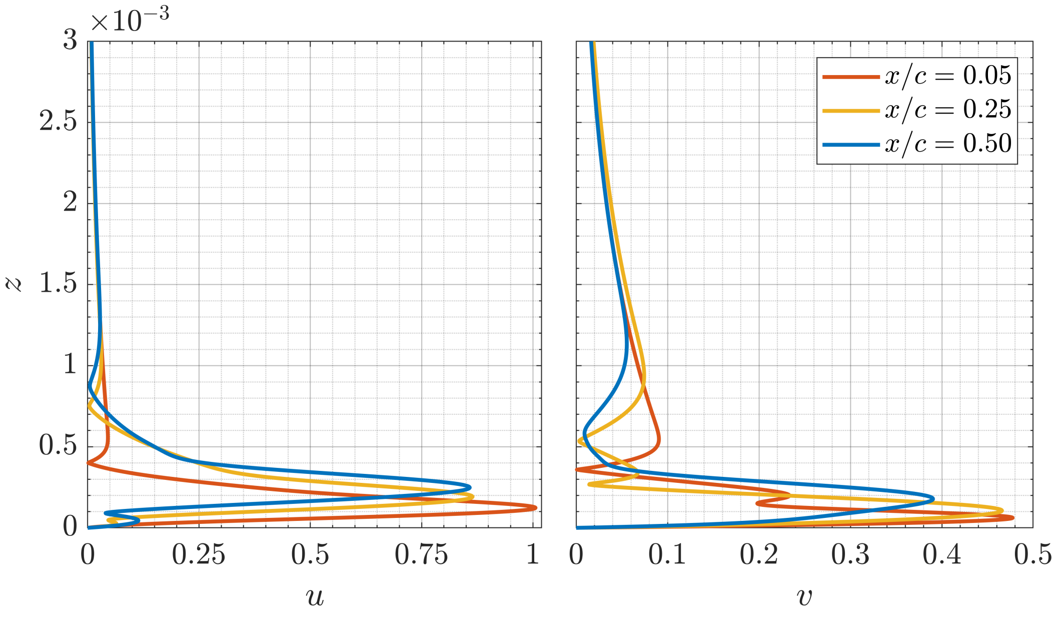
At this point, the values of the previously mentioned wave angle are also available, which vary with the streamwise coordinate. The values of the angle ψ corresponding to the coordinates $x/c = [0.05, 0.25, 0.5]$ in degrees are [28.24, 31.63, 28.96] and [74.85, 77.36, 76.75] respectively for AoA 1.25° and -1.00°.

Therefore, this allows us to confidently define that the highlighted mode for the AoA equal to 1.25° a TS-type instability. Conversely, for the negative AoA considered, the instability is of the CF type since the values of the angle ψ are greater than 70° to 75°.

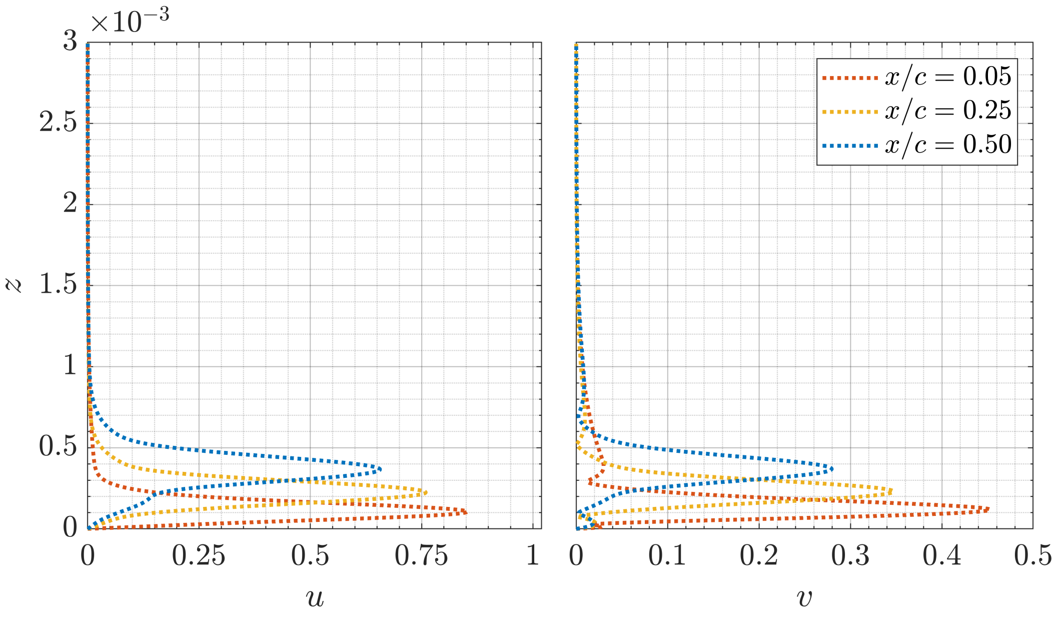
Having already introduced Figure 10.2, we can make further observations regarding the selection of the mode to be used for optimization studies. By comparing the N-factor values for the two different AoA conditions, it is observed that the maximum values reached are very different even considering the same flow conditions in terms of Re, Ma, and Λ. In the presence of APG, CF-type instabilities tend to exhibit a more pronounced growth trend compared to TS modes, thus being responsible for boundary layer transition. As confirmation of this, focusing solely on Figure 10.2b, it is observed that the N-factor curves with a similar parabolic trend are all within the envelope curve, which in turn is shared with the highlighted CF instability.

To conclude, further insight will be provided later regarding the effect of selecting the most unstable mode to be used in the optimization algorithm on the calculation of dimensional tolerances, specifically for TS-type instabilities. For this latter type of instability, indeed, it is observed that for several of them, values of the N-factor close to what can be considered critical for transition are reached.

Regarding CF-type instabilities as visible in Figure 10.2, it is quite evident which mode should be considered as the most unstable. In this case, it certainly does not make sense to consider the maximum value of the N-factor (as in the case of TS-type), since the maximum, as already observed, is reached at the right limit of the analysis domain, assuming values of the order of 25 to 30, thus definitely post-transition. Therefore, in this case, it makes sense to consider the growth trend of the modes near a value of the N-factor that can be assumed as a limit beyond which the boundary layer transition occurs; the one that reaches that value first and with the steepest trend is the one to be considered.



(a) Profiles of streamwise (left panel) and spanwise (right panel) velocity components for the TS instability mode highlighted in Figure 10.2a.



(b) Profiles of streamwise (left panel) and spanwise (right panel) velocity components for the crossflow instability mode highlighted in Figure 10.2b.

Figure 10.3 Comparison of the streamwise and spanwise velocity components for three different x/c coordinates for the two instabilities highlighted in Figure 10.2, aiming to demonstrate the physical nature of the Tollmien-Schlichting (TS) and crossflow (CF) instabilities, rather than being spurious. For the parameters of the instabilities considered, refer to Table 10.2.

10.2 Largest allowable deformation profiles

The final deformation profiles, obtained considering the NLF(2)-0415 airfoil as the baseline geometry for a 45-degree swept wing, and flow conditions of $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$, are reported. Different numbers of Fourier modes were considered for introducing wavy shapes on the surface, specifically $k \in [10, 12, 14, 16]$. The results are presented based on the final shape obtained when the change in the ΔN is approximately 2. The results obtained exclusively employing the GA algorithm; based on the observations of Moniripiri in [1], it does not yield the largest allowable deformation profiles that minimize the L_2-norm for a certain N-factor increment.

Table 10.4 L_2-norm values obtained for $\Delta N \approx 2$ and different number of Fourier modes ($k \in [10, 12, 14, 16]$). Flow conditions are $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$.

$\Delta N \approx 2$

N. Fourier modes	**10**	**12**	**14**	**16**
L_2-norm($\times 10^{-4}$) [m]	11.0	7.15	6.20	6.48

Considering the L_2-norm values reported in Table 10.4, it is evident that as the number of Fourier modes considered decreases, i.e. the longer the characteristic wavelength of the undulation, the root mean square deviation increases. Particularly, looking at Figure 10.4, it is observed that the major contribution to the increase in the L_2-norm is given by the large bumps presented in the upstream position of the chord, where the boundary layer is thinner and where the influence in disturbing the flow is higher. The importance of the boundary layer thickness in the growth rate of the disturbance is well supported by the analysis of the case $k = 16$, where the L_2-norm shows a lower value compared to $k = 10$, due to the more pronounced initial perturbation of the surface, which helps limit the oscillations in the latter part of the profile. It is worth noting that the introduction of large bumps near the leading edge results in steep slopes downstream, which significantly influence the pressure gradient distribution. Finally, it is observed that the case $k = 16$ presents a slightly higher value of L_2-norm compared to $k = 14$, and this can once again be attributed to the magnitude of the first oscillation.

The information regarding the pressure distribution and its derivatives is crucial for understanding the physical significance of the discussed wavy shapes. The sensitivity to design variables of kinetic energy, within the boundary layer, depends directly on the pressure distribution along the airfoil chord. To further clarify this concept, it is easier to examine the quantities depicted in the various panels of Figure 10.5 for the cases $k = 10$ and $k = 16$ and conditions $Re = 15 \times 10^6$, $M = 0.5$ and $\Delta N \approx 2$.

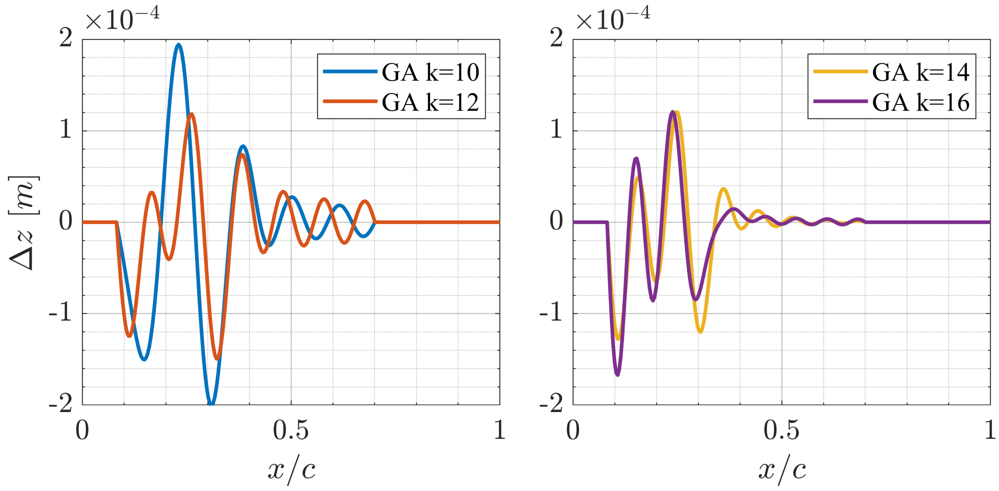


Figure 10.4 Largest allowable deformation profiles using $\gamma = 6 \times 10^{-6}$ for $\Delta N \approx 2$ at $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$ considering different numbers of Fourier modes on the NLF(2)-0415 airfoil; obtained using the GA method. The results are reported in two separate plots for ease of readability.

Considering that the aim of this study is to investigate the effect of introducing wavy geometric perturbations (on the upper surface of the airfoil) on the growth of the N-factor, i.e. on the increase of kinetic energy within the boundary layer, we are interested in examining the regions where the sensitivity $\nabla_z E$ is positive. Looking at the bottom panel in Figure 10.5 and comparing it with $d^2(-C_p)/dx^2$ is negative. This signifies that kinetic energy within the boundary layer tends to increase moving towards APG regions, and this is physically reasonable. The increase in energy can be linked to the rise in the N-factor and, consequently, to the increase of the instability condition of the selected mode, with the potential to trigger transition.
To allow easy readability of Figure 10.5 vertical lines indicating the positions where $d^2(-C_p)/dx^2 = 0$ are reported (the style of the lines match the legend references, dotted for $k = 10$), so distinguishing regions where $d(-C_p)/dx$ changes slope sign is straightforward. The relation described earlier between different quantities reported, is clearly evident; considering $k = 16$, in the region preceding $x/c \approx 0.165$ delimited by the first vertical solid line, sensitivity is positive while the second derivative of C_p is negative. For $k = 10$, the trend in this region is the opposite, which aligns with the observations made in Figure 10.4; due to the longer wavelength compared to $k = 16$, the top of the first protrusion is reached at a more rearward position, consequently

delaying the effects that contribute to the increase in energy. Referring back to $k = 16$, after $x/c \approx 0.165$, as the pressure gradient starts to move towards the favorable type, i.e., $d(-C_p)/dx$ begins to increase ($d^2(-C_p)/dx^2$ is positive), the sensitivity tends to become negative. This trend is observable across the entire surface of the airfoil. For $k = 10$, the first region where $\nabla_z E$ is positive occurs immediately after $x/c \approx 0.197$. Comparing the numerical values of the derivatives of C_p for the two values of k considered here, it is evident that as the wavelength of surface oscillations decreases, the introduced pressure gradients become more critical, as they assume higher values; this effect is combined with a clearly increased oscillation periodicity. Additionally, there is a greater sensitivity of energy to changes in profile coordinates in the vertical direction, as indicated by the significantly higher values of $\nabla_z E$. These findings are consistent with the results of the tolerance calculation discussed later.

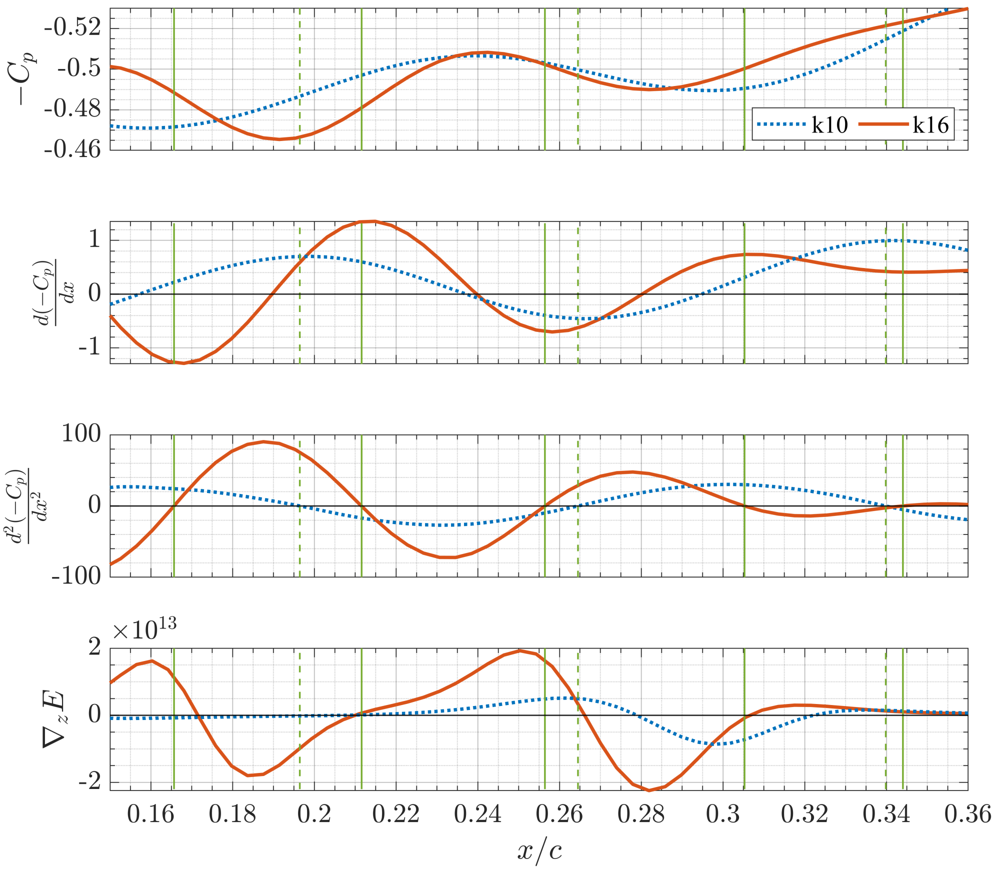


Figure 10.5 Effect of the pressure distribution on the gradient for $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$, $\Delta N \approx 2$ and $k = [10, 16]$. Please note that for clarity colors do not match with the ones in Figure 10.4.

In Figure 10.6, the envelope of N-factor curves and the instability mode considered for the baseline airfoil are presented, along with the shape of the N-factor for $k = 10$ and $k = 16$ for their respective final wavy configurations; as visible and previously mentioned, the increment in the considered N-factor, ΔN, is approximately 2. In addition, it is observed that the trend of the N-factor closely follows the oscillations of the airfoil surface, allowing the identification of the number of waves introduced by the Fourier series. Comparing the N-factor trends for the two cases, it is noted that the presence of the first peak, visible in Figure 10.4, with a similar amplitude but positioned further forward in the case of $k = 16$ (and thus characterized by a steeper slope), results in a more pronounced increase in the growth rate value. As noted above, disturbances occurring further forward on the surface have more pronounced effects due to the thinner boundary layer. To confirm this, we can also refer to the number of iterations required to reach a value of $\Delta N \approx 2$, as we used the same value of γ to advance in the direction suggested by the gradient. For $k = 10$, 150 iterations were required, while for $k = 16$, only 95 iterations were needed, which is approximately one-third fewer.
With reference to Figure 10.6, it is also noteworthy the importance of considering the selected unstable mode rather than the one corresponding to the maximum at the right boundary of the stability calculation domain. This choice proves crucial in the transition process due to the sudden increase occurring near $x/c \approx 0.2$.
What one would expect to find employing the SLSQP algorithm under the same conditions is a further increase in the amplitude of the oscillations in the first part of the profile, leading to an overall decrease in the root mean square deviation.

10.3 Calculating manufacturing tolerances

In this Section, we are discussing the results obtained from the tolerance study. For the methodology used to compute tolerances, h_{tol}, considering the L_2-norm of surface deviations of the allowable deformation profile with respect to the baseline airfoil NLF(2)-0415, please refer to Section 6.3.
Since the study lends itself to a parametric analysis aimed at defining a design space, the results are presented in distinct subsections to clearly illustrate the effect of each parameter on the obtained dimensional tolerances. In particular, the parameters varied include Re, M, AoA, as well as the number of Fourier modes considered, k, and the characteristic parameters of the instability mode under consideration (specific to each

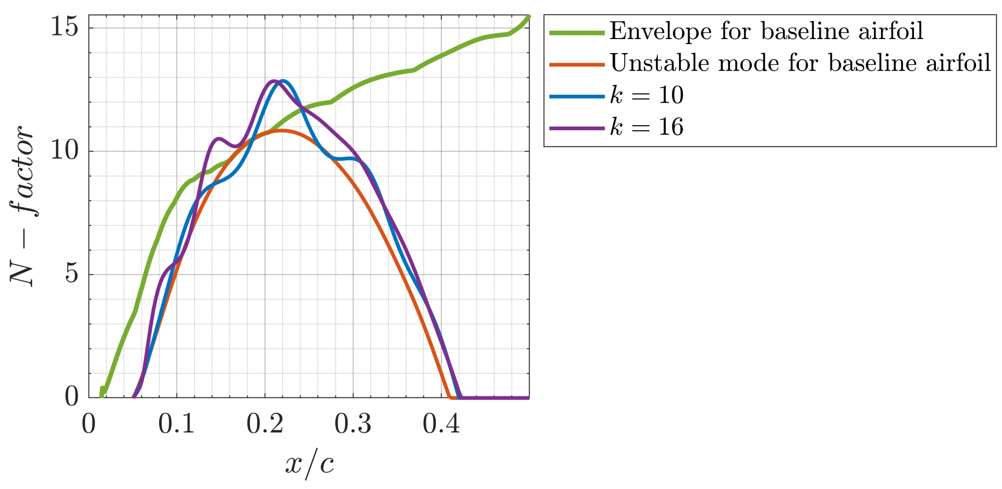


Figure 10.6 Envelope of the N-factor and evolution of the N-factor corresponding to the chosen most unstable mode for the baseline airfoil compared to the ones obtained after reaching $\Delta N \approx 2$ using gradient ascent method. Conditions are $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$ and $k = [10, 16]$.

case). We chose to keep fixed the value of the sweep angle Λ, considering that most civil aviation aircraft exhibit this geometry or a very similar one. Nevertheless, given the flexibility of the numerical algorithm developed, it is possible to repeat the same study quite easily.

10.3.1 Effect of Reynolds number and wavelength of waviness

The selection of Reynolds numbers was based on the usual aerodynamic conditions and mean chords of civil aviation aircraft characherized by tapered wings operating at maximum operational speeds, V_{MO}, under ISA conditions. The results has been obtained for $Re = [9, 12, 15] \times 10^6$, $M = 0.5$, $AoA = 1.25°$ and $k = [10, 12, 14, 16]$; Table 10.1 provides information on the values of the parameters f, β, and α_{ms} for the most unstable modes corresponding to the mentioned Reynolds numbers. Note that for these conditions the unstable modes considered critic are all of the type of TS modes; for the case $Re = 15 \times 10^6$ the respective N-factor curve is represented in Figure 10.2a. Let us consider Figures 10.7–10.9, which depict h_{tol} as a function of the increase in N-factor, utilizing the GA method. It is noted that not all combinations of parameters have results available up to a value of $\Delta N = 2.5$, as in such cases the number of iterations performed was very high. Nevertheless, the maximum value of ΔN considered is determined within the scope of studying a specific case, and in some instances, even an increase of a single point in the N-factor may suffice. As explained in Section 6.1, in the algorithm employed, the value of ΔN serves both as the initial and the stopping criterion and is chosen "a priori".

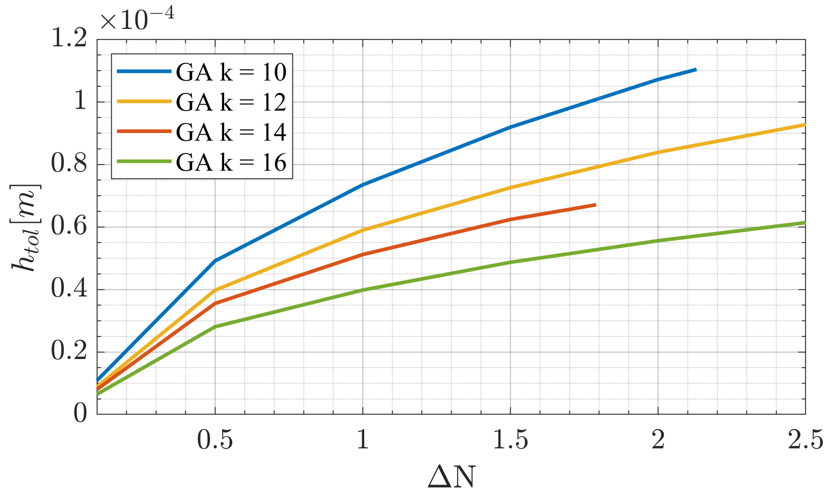


Figure 10.7 Evolution of h_{tol} for $Re = 9 \times 10^6$ and different ΔN and k Fourier modes. Conditions are $M = 0.5$, $AoA = 1.25°$.

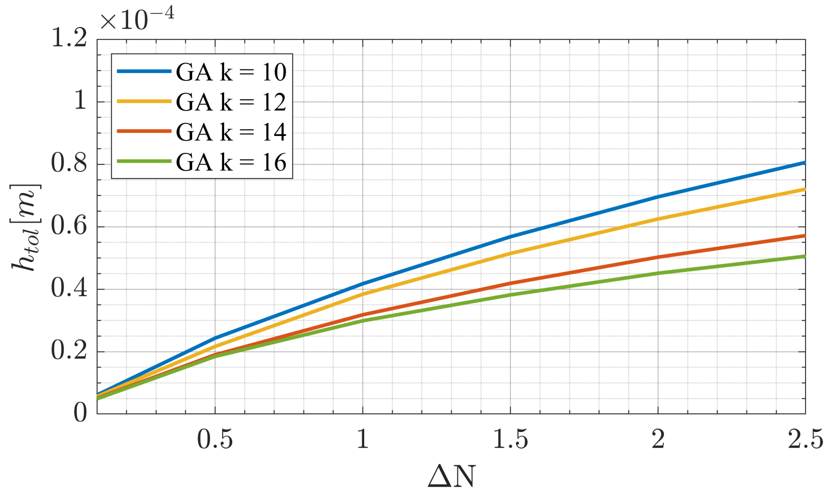


Figure 10.8 Evolution of h_{tol} for $Re = 12 \times 10^6$ and different ΔN and k Fourier modes. Conditions are $M = 0.5$, $AoA = 1.25°$.

In all the mentioned figures, it is possible to observe that the tolerance increases as ΔN increases. This observation is reasonable because, in a certain sense, it indicates that by allowing a larger increase in the N-factor, we can relax the tolerance requirements. Similarly, it suggests that a more pronounced wavy shape is needed to achieve higher increases in kinetic energy within the boundary layer.
In addition, examining one figure at a time, i.e. keeping Re number fixed, it can be

observed that for a specific target ΔN, decreasing the wavelength of the undulations on the airfoil profile (increasing the value of k), leads to stricter tolerances. This is associated with introducing more pronounced and frequently pressure variations along the chord.

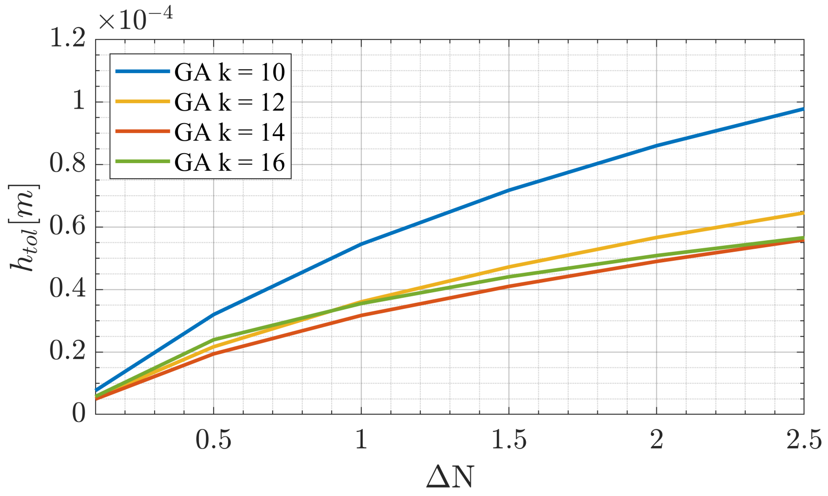


Figure 10.9 Evolution of h_{tol} for $Re = 15 \times 10^6$ and different ΔN and k Fourier modes. Conditions are $M = 0.5$, $AoA = 1.25°$.

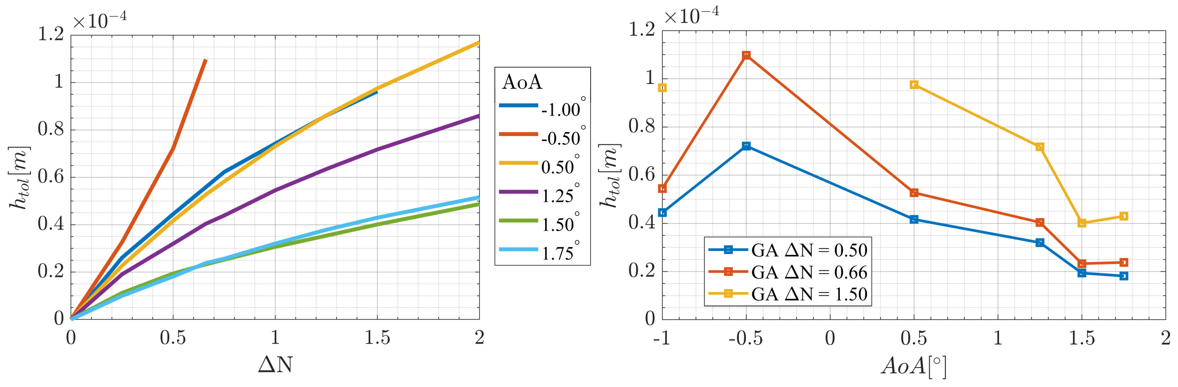
The observation is more pronounced for low Reynolds numbers, as a higher value of this parameter results in a thinner boundary layer. In that case, even small surface deviations have a significant effect on the pressure coefficient distribution, leading to pronounced growth in disturbances.
Supporting this, let us compare the tolerance values for $\Delta N = 2$: $h_{tol} = [1.07, 0.56] \times 10^{-4}[m]$ for $Re = 9 \times 10^6$ and $h_{tol} = [0.86, 0.51] \times 10^{-4}[m]$ for $Re = 15 \times 10^6$, respectively for $k = [10, 16]$. At a fixed Reynolds number, transitioning from higher to lower value of k yields a percentage reduction in h_{tol} of approximately 48% and 41%, respectively for $Re = [9, 15] \times 10^6$. To conclude, transitioning from $Re = 9 \times 10^6$ to $Re = 15 \times 10^6$ results in a percentage reduction in the allowable value of h_{tol} of about 20% and 9%, respectively for $k = [10, 16]$, indicating that the thickness of the boundary layer significantly influences the definition of the quantity we are evaluating. It is also noted that as Re increases, the effect of the profile surface wavelength tends to decrease; for each ΔN, the curves in Figures 10.8–10.9 tend to be closer together compared to what is shown in Figure 10.7. It is noteworthy that for the case $Re = 15 \times 10^6$, the observed trend is slightly violated especially for lower values of ΔN; however, this is in line with the fact that the L_2-norm presented in Table 10.4 for $k = 16$ is a little higher compared to $k = 12$, $k = 14$. In these conditions, therefore, the periodicity in the alternation of

regions with APG and FPG becomes more critical for longer wavelengths.
Considering the properties of the SLSQP method, what we expect when comparing its results with those obtained using the GA method is a lower value of h_{tol} for each value of ΔN. This expectation arises from the fact that the SLSQP method allows us to find the waviness surface with the minimum L_2-norm, and the tolerances are defined based on this quantity. Hopefully, these results will be available shortly so that they can be incorporated into the already reported plots.

10.3.2 Effect of angle of attack

Given the consideration of a non-zero sweep angle, Λ, AoA emerges as one of the most significant parameters in this study. As outlined in Section 10.1, two types of instability exist: Tollmien-Schlisting (TS) and crossflow (CF). The prevailing type is determined by the AoA, which defines the pressure distribution along the chord of the profile. Each type of instability will exhibit a distinct response to surface disturbances (waviness of C_p).
The results reported in this section have been obtained for $Re = 15 \times 10^6$, $M = 0.5$, $k = 10$, and variable AoA; Table 10.2 provides information on the values of the parameters f, β, and α_{ms} for the most unstable modes corresponding to the different AoA considered. It is worth noting that for negatives AoA chosen the modes considered are of the crossflow type while the others are of the TS type. Reference can be made to Figures 10.2 for an example of the evolution of the N-factor for AoA equal to $-1.00°$ and $1.25°$.
Observing Figures 10.10, we can see that, considering positive (or nearly positive) values of the angle of attack, as it decreases, we expect an increase in manufacturing tolerances, since the presence of FPG has a stabilizing effect in the boundary layer, requiring larger waviness amplitudes to cause the same variation in the N-factor. This holds true, but only if the observation is limited to the values of AoA mentioned; furthermore, this observation is consistent with the results obtained in [1] by Moniripiri *et al.* However, this study, assuming the presence of CF-type instabilities, i.e., considering wing taper, highlights that in the presence of higher (in absolute value) negative angles of attack, dimensional tolerances start to decrease again. This fact is related to the sensitivity of CF-type instabilities to the waviness of the profile surface; in particular, for these modes the presence of FPG (see Figure 10.1) tends to trigger destabilization effects. Conversely, for the largest positive angles of attack the tolerance to reach the target ΔN

decreases because the APG, especcially close to LE becomes stronger and destabilize the TS waves resulting in a faster growth in their amplitude.



(a) Evolution of h_{tol} as a function of ΔN, considering as a parameter the AoA.

(b) Evolution of h_{tol} as a function of AoA, considering as a parameter ΔN.

Figure 10.10 Evolution of h_{tol} for different values of AoA, and ΔN. Conditions are: $Re = 15 \times 10^6$, $M = 0.5$ and $k = 10$.

At this juncture, it is important to remember that the values of ΔN considered for creating the plot were chosen "a priori". In fact, it should be reminded to the reader that the tolerance values we refer to, simply correspond to an increase in ΔN and do not represent the tolerance for which the transition associated with the maximum N-factor value occurs. The value of N-factor at which the boundary layer transition occurs, i.e. below which one must stay to preserve laminar flow, is not generally known. It is important to mention again this assumption because the maximum N-factor value is very sensitivy to the AoA. Thus here that we are actually considering different angles, increase in ΔN is the same but the value in absolute sense could be significantly different.

For clarity regarding what is highlighted here, is useful to consider the following results. Under the condition $AoA = 0.50°$, the clean wing showed a maximum N-factor of approximately 9.9; conversely, for $AoA = -1.00°$, the clean wing exhibited a maximum N-factor value of approximately 14.2. As mentioned, since the maximum value of h_{tol} calculated using our methodology corresponds to $(N_{max,init} + \Delta N)$, and $N_{max,init}$ is quite different we are referring to different N_{max} for the last iteration (to reach target ΔN). Particularly these values could be lower or higher to the N-factor value assumed as a reference for the transition in the initial approximate analysis. Regarding this, the exact N_T value is challenging to define, often requiring wind tunnel experiments or flight tests. The value of $N_T = 9$ is not universal and varies with turbulence intensity (flight tests have identified the beginning of transition for $N_T = 13.5$ and $N_T = 17.2$, among others).

Therefore, once more, keep in mind that the present methodology allows the determination of manufacturing tolerances based on acceptable ΔN values to be decided by the aerodynamicist, that will know in advance the maximum expected ΔN that the geometry can produce.

Referring to the results shown in Figure 10.2, one might wonder why tolerance values up to $\Delta N = 2$ are not available for some $AoAs$, especially considering that these cases correspond to negative values. The reason is actually quite simple; it has been observed that CF instabilities are much more resistant to destabilization, especially for $AoA = 0.5°$ and require a greater number of iterations to reach the same ΔN value. So the algorithm was stopped earlier with respect to the other cases.

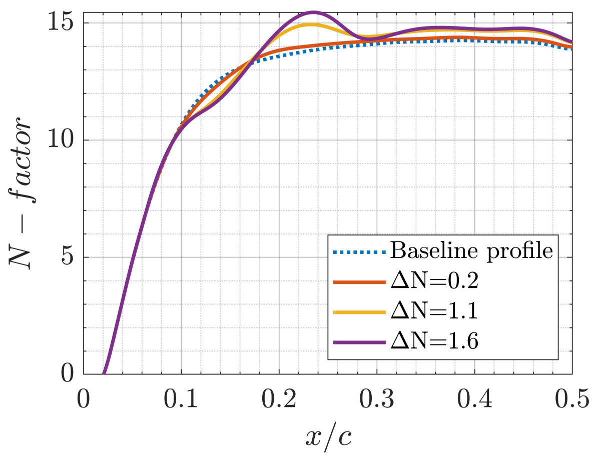


Figure 10.11 Envelope of the N-factor and evolution of the N-factor corresponding to the chosen most unstable mode for the baseline airfoil compared to the ones obtained after reaching the ΔN values outlined in the legend. Conditions are $Re = 15 \times 10^6$, $M = 0.5$, $AoA = -1.00°$, and $k = 10$.

It is also important to consider how ΔN is calculated for this type of instability. Refer to Figure 10.11 which shows the evolution of the N-factor for three successive iterations as well as the one for the baseline airfoil profile, for $AoA = -1.00°$. Since, as previously noted, we are not interested in the maximum value of the N-factor at the end of the domain, we have chosen to define ΔN as the difference in the N-factor evaluated at the x/c coordinate where the perturbed curves have their first maximum. Note that in some cases, this maximum is relative and not absolute, as the N-factor could further increase, but to a lesser rate of growth; it is usually possible to distinguish a point where the introduction of waviness has the most pronounced negative effect.

10.3.3 Effect of Mach number

Here, we aim to consider the effect of the Mach number on the variation of the N-factor curve following the introduction of undulations on the airfoil surface. The findings presented in this section were obtained for $Re = 15 \times 10^6$, $AoA = 1.25°$, $k = 10$, and varying M. Table 10.3 provides details on the values of the parameters f, β, and α_{ms} for the most unstable modes corresponding to the different Mach numbers under consideration.

Observing Figure 10.10 it can be stated that, as known from the theory, increasing the Mach number directly impacts the pressure distribution across the entire airfoil. This augmentation magnifies all existing pressure gradients by a uniform factor, which can be estimated through subsonic compressibility equations like the Karman–Tsien equation. As a result, raising the Mach number intensifies the APG caused by the wavy bumps and this leads to reduced tolerances.

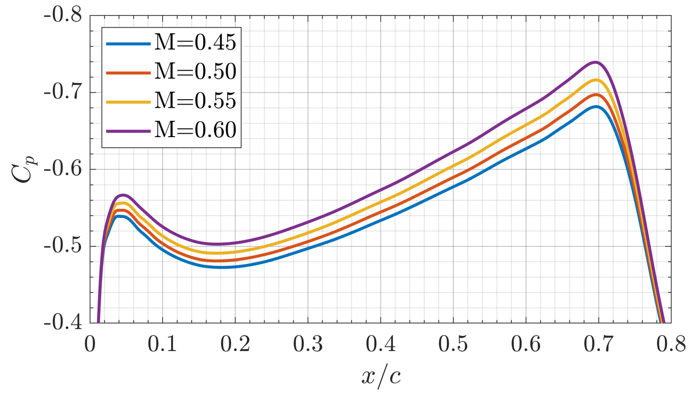


Figure 10.12 Pressure coefficient C_p as a function of the coordinate x/c analyzed for various Mach number values. Conditions are $Re = 15 \times 10^6$, $AoA = 1.25°$, and $k = 10$.

In Figures 10.11, the tolerances for $AoA = 1.25°$ and $Re = 15 \times 10^6$ are presented. Specifically, 10.14 illustrates how compressibility influences h_{tol} across four different Mach numbers, with a target ΔN value of 1.5. The obtained result is not trivial since, although pressure gradients tend to become more severe, it is also known that compressibility effects play a positive role in stabilizing TS-type instabilities. In this regard, reference can be made, for example, to the work of Arnal *et al.* [26].

Unfortunately, results for different values of k and especially for different values of AoA are not available. This could have allowed us to assess whether a different periodicity of the surface would have altered the trends. Additionally, the impact of compressibility on CF-type instabilities could have been evaluated. Most likely, we would have observed a reduction in h_{tol} considering the shape of C_p for negative AoA and their sensitivity to FPG situations.

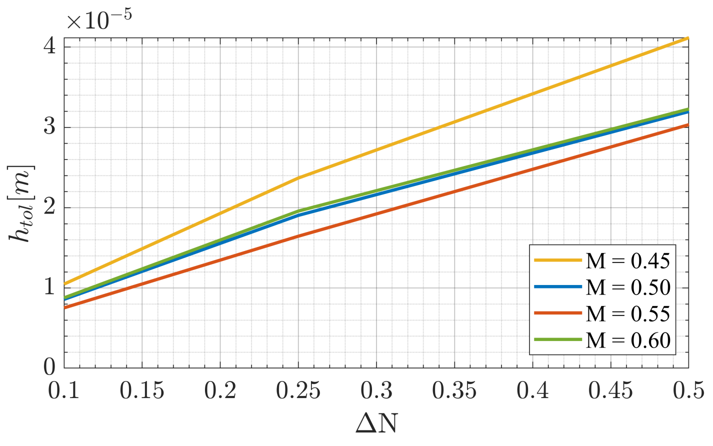


Figure 10.13 Evolution of h_{tol} with respect to ΔN for different values of M. Conditions are $Re = 15 \times 10^6$, $AoA = 1.25°$, and $k = 10$.

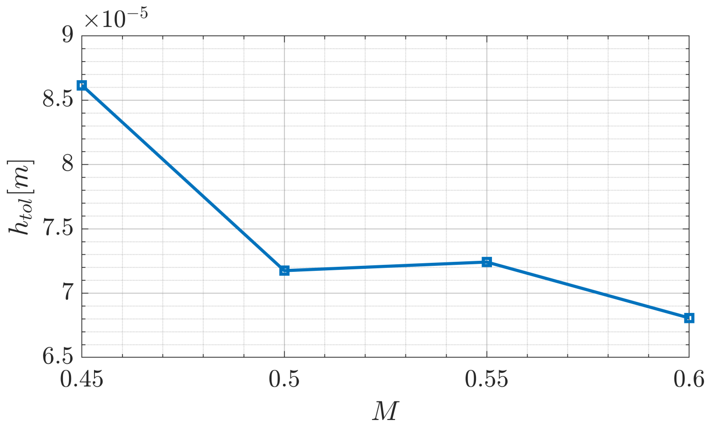


Figure 10.14 Evolution of h_{tol} with respect to M for $\Delta N \approx 1.5$. Conditions are $Re = 15 \times 10^6$, $AoA = 1.25°$, and $k = 10$.

10.3.4 Effect of mode selection

In this section, we aim to assess the influence of the selected instability mode on the calculation of dimensional tolerances. To conduct the stability analysis, we needed to designate the modes to be used from a wide range of different feasible modes, and this would like to be the most unstable one. Identifying the mode of interest can be challenging; in fact, claiming a mode to be the most amplified among all potential modes is not entirely accurate, given the impossibility of finding an infinite number of instability modes. Once the mode is selected, it will be utilized in PSE equations as the initial condition. In the PSE approach, unlike in local linear stability analysis, the streamwise wavenumber is not constant, and the initial mode acts as the starting condition for solving PSE with a marching algorithm.

Fortunately, it is known from theory that optimizing kinetic energy based on a single mode affects the growth of numerous disturbances. Consequently, if sensitivities are derived from the most amplified mode identified, it is anticipated that no other mode will exceed its amplification during the process. This assertion has also been demonstrated by Moniripiri et al. in [1]. Consequently, throughout the process, we opted to maintain the mode constant, as it results in significant computational time savings.

However, a potential issue in the mode selection process is the existence of multiple modes with nearly identical maximum N-factors but differing frequencies. Consequently, various modes could qualify as strong contenders for being considered the most unstable and, therefore, utilized for computing h_{tol}. In that case, a specific mode with slightly lower maximum N-factor might result in lower tolerances due to resulting pressure gradients in the final airfoil shape.

We therefore chose to consider three different modes characterized by similar maximum N-factor values, around 10. Specifically, the results were obtained for the condition of $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$, and $k = 10$. For this flow configuration, it has already been observed that TS-type instabilities dominate.

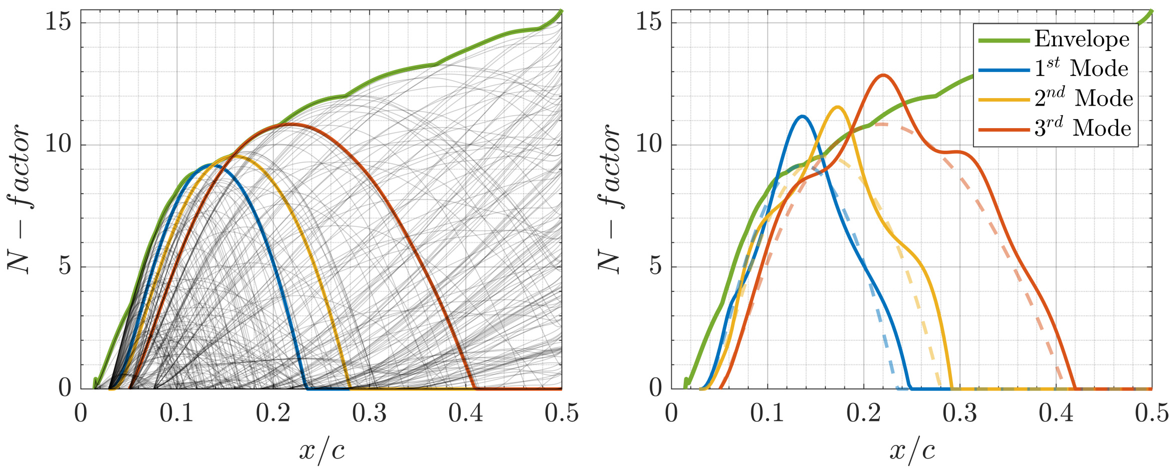
Below, in Table 10.5, the parameters corresponding to the different instability modes considered are reported, used in the stability analysis. The evolution of the N-factors are reported in Figures 10.15. Figure 10.16 shows the tolerances in relation to ΔN

Table 10.5 Parameters used for analysis considering three different instability modes; flow conditions are fixed as reported in the header of the Table. Please note that the 3^{rd} mode corresponds to the one considered in the results presented in the previous sections for the mentioned flow conditions.

$Re = 15 \times 10^6$, $AoA = 1.25$, $M = 0.5$

$Mode$		1^{st}	2^{nd}	3^{rd}
f	$[Hz]$	12050.50	10508.30	9106.24
β	$[m^{-1}]$	1198.05	1198.05	760.26
α_r	$[m^{-1}]$	718.70	520.87	736.33
α_i	$[m^{-1}]$	1.85	3.01	-18.38

for the three modes considered; it can be observed that the 3^{rd} mode determines the



(a) Evolution of N-factor and envelope for all instability modes found from the stability analysis performed on the baseline geometry. The three modes considered in the analysis of this Section are highlighted.

(b) Comparison between the evolution of the N-factor for the three modes considered before and after the introduction of waviness in the profile to achieve $\Delta N \approx 2$.

Figure 10.15 Envelope of the N-factor and evolution of the N-factor corresponding to the chosen modes after reaching $\Delta N \approx 2$ using gradient ascent method. Conditions are $Re = 15 \times 10^6$, $M = 0.5$, $AoA = 1.25°$, and $k = 10$. Parameters are reported in Table 10.5.

minimum value of h_{tol} when compared with the others. It is interesting to compare the N-factor curves reported in Figure 10.15b with what was just observed; in particular, the growth of the N-factor for the 3^{rd} mode appears to be the most critical one. This is because the increase in the parameter occurs over a more extended region, particularly at the most advanced position where the boundary layer is thinner, potentially resulting in a more critical effect on the stability of the boundary layer. This reasoning can also be applied to the first and second modes; following the same logic, the second mode appears to be more critical than the first, which is confirmed by the calculated values of h_{tol}. Additionally, in this comparison, it is noted that the maximum value of the N-factor is practically the same. Therefore, when selecting the mode for calculating tolerances, not only the absolute value of the N-factor but also its growth rate should be considered. In summary, although the maximum N-factor is the same for either the first or second most amplified mode, the variation of the N-factor along the chord, which directly indicates the pressure distribution on the airfoil surface, is different and should be taken into account. When the second mode is used, the final optimized shape exhibits a large bump (indicating a large adverse pressure gradient) at the upstream

position of the chord, where the boundary layer is thinner, resulting in a lower tolerance compared to when the first mode is used. When the third mode is used, the final optimized shape exhibits a large bump compered to the second one.
In this section, the aim is to reiterate the importance of mode selection for tolerance calculation and emphasize the need to carefully consider each individual case, including the relationship between the increase in the N-factor and the boundary layer transition if sufficient data are available to accurately define the N-factor limit at which transition occurs (for a specific profile and flow conditions, refer to what already mentioned in Section 10.3.2) .

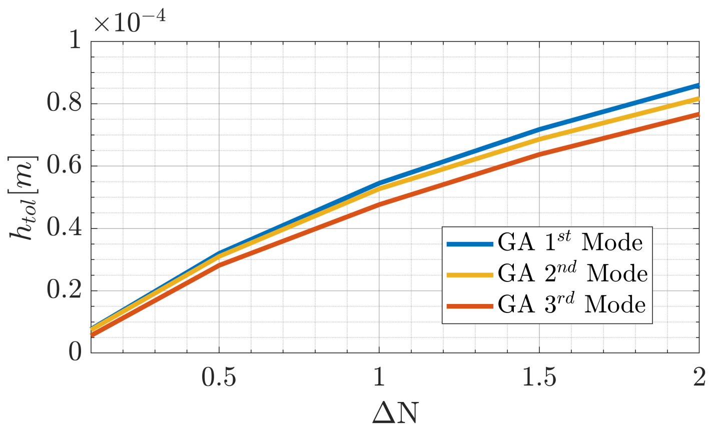


Figure 10.16 Evolution of h_{tol} with respect to ΔN for different instability modes considered. Flow conditions are: $Re = 15 \times 10^6$, $AoA = 1.25°$, $M = 0.5$, and $k = 10$.

Chapter 11

Results for transition delay study

In this chapter, some of the preliminary results obtained regarding the transition delay of boundary layer study are presented. During the development of the numerical implementation of this study, it became necessary to address various numerical and theoretical issues (some of which are presented here), significantly slowing down the progress of the work in order to achieve meaningful and conclusive results. Being a first attempt to use the developed calculation algorithm, it was not possible to set up a parametric study like the one proposed in the previous study, i.e., considering different values for the Reynolds, Mach, angle of attack flow parameters, and type of instability mode. Given the difficulties of the set study, the approach chosen was to select a particular flow condition and try to solve the various issues for the said case with the intention, subsequently, once a satisfactory result was achieved, to try to replicate the same methodology under different conditions. The following study draws inspiration from research conducted experimentally at TU Delft, where it was observed that, in the presence of cross-flow components, the introduction of a suitably defined bump (protuberance) can delay the boundary layer transition. Although the results of this study have not yet been published, it is known to have been conducted under conditions of incompressible flow, at a negative angle of attack, and in the presence of cross-flow velocity components. However, further details are not available; for this reason, it has been decided to consider a priori certain flow conditions, deferring to the future the possibility of setting up an optimization study for the conditions used by the Dutch colleagues who found that the solution provides improvements without optimizing its shape. Also, regarding the choice of the baseline wing profile to optimize, it was decided to follow this strategy, preferring to continue working with the

NLF(2)-0415 wing baseline profile with a 45° sweep angle, in order to leverage some of the information already obtained from the tolerance study previously presented.

11.1 Sensitivity information

Considering that the purpose of this study is to attempt to delay the transition of the boundary layer, which can be associated with the effort to reduce energy within the boundary layer, it is first necessary to determine whether, for the chosen flow conditions, it is possible to optimize the shape of the selected airfoil profile in order to delay the transition. This can be done by observing the trend of sensitivity with respect to the design variables, i.e., $\nabla_z E$, and verifying if it takes a negative value in a certain region. As explained in detail in Chapter 7, in order to minimize the energy of the system, with a gradient descent algorithm it is possible to make variations to the baseline geometry following the direction of maximum decrease. Therefore, if the sensitivity $\nabla_z E$ takes a negative value, it means that moving in the opposite direction of the gradient, i.e., introducing a convex bump to the baseline geometry of the profile, can achieve the desired result confirming the observation of the colleagues from TU Delft.
In Figure 11.1, the variation of the energy gradient computed with respect to the vertical coordinate of the baseline profile grid points is shown for the flow conditions $Re = 9 \times 10^6$, $AoA = 1.25°$, $M = 0.5$, and the instability mode whose parameters are listed in Table 11.1. The chosen instability mode corresponds to the CF type as it has been observed that these are the ones suitable for this type of study. It has already been observed that crossflow instabilities tend to dampen near APD; therefore, introducing a bump in a sufficiently advanced position on the profile limits the extent of the region where pressure decreases. It should be noted that for convenience, reference is made to flow conditions with positive AoA, although CF-type instabilities are dominant under negative incidence conditions. However, at this stage of the study, we are not particularly interested in referring to the most critical instability that triggers transition; rather, at this phase, we are interested in assessing whether the implemented procedure effectively results in benefits in terms of instability damping. Only later it will be appropriate to carefully choose the mode to effectively delay transition. Regarding this, it is worth noting once again the extreme difficulty in defining a threshold N-factor for transition, N_T, in a general manner; accurate boundary conditions are indeed necessary to set a reference value, considering turbulence intensity, etc..

Returning to Figure 11.1, it can be observed that over a sufficiently extended x/c

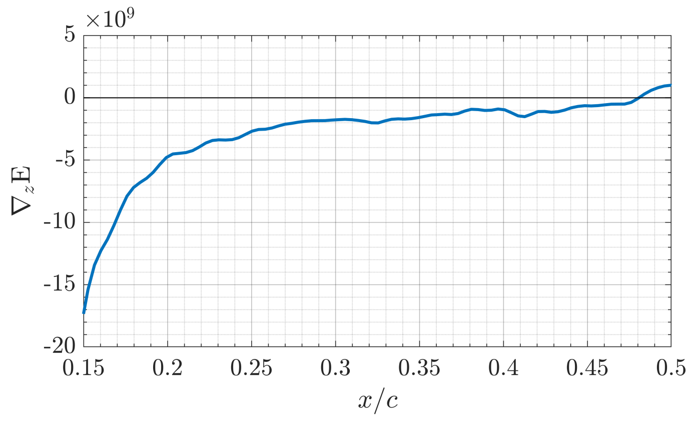


Figure 11.1 Sensitivity of energy with respect to design variables, $\nabla_z E$, for the baseline airfoil considering the instability mode whose parameters are reported in Table 11.1 is depicted. Flow conditions are: $Re = 9 \times 10^6$, $AoA = 1.25°$, $M = 0.5$.

interval ($\approx [0.15, 0.50]$), $\nabla_z E$ takes on negative values, suggesting that by introducing a properly shaped bump, there is indeed the possibility of reducing the objective function, E. The sensitivity trend displayed has been limited to a confined region because for x/c values less than 0.15, non-physical oscillations occur, which obviously must be disregarded. The issue is once again related to what was discussed in 9.3.1, along with the fact that this region is near the stagnation point, before which BLE and PSE are not solved along with their corresponding adjoint equations.

For the conditions considered here, there are good chances to optimize the profile

Table 11.1 Parameters of the instability mode used for transition delay analysis considered in the current section.

$Re = 9 \times 10^6$, $AoA = 1.25°$, $M = 0.5$		
f	$[Hz]$	4352.76
β	$[m^{-1}]$	682.69
α_r	$[m^{-1}]$	102.95
α_i	$[m^{-1}]$	-3.10

shape in order to delay transition. However, it is necessary to verify on a case-by-case basis whether this is actually the case or not.

11.2 Considerations on the positioning of bumps

As explained in Section 7.1.1, the sensitivity of the energy calculated with respect to the design variables, i.e., $\nabla_z E$, is projected into the space defined by a predefined number of Hicks-Henne bump functions. The projection aims to compute the amplitude of each bump function considered in order to find the optimal shape of the profile that allows for a reduction in kinetic energy. Once it has been established that sensitivity analysis highlights the possibility of optimizing the shape, as discussed in the previous section, it is necessary to consider some precautions in the selection of the bump functions to be used for the projection operation. It is worth noting that although the shape of the H-H bump functions is controlled by some parameters, users need to pay attention when selecting them to avoid encountering unexpected situations. Arbitrarily assigning values to the parameters of the H-H functions, i.e., not considering the shape of the bumps after projection for a specific case, in order to generalize the discussion as much as possible, allows for more prominent highlighting of the issues encountered in the various simulations conducted.

Figure 11.2 depicts a possible condition that may occur; it is observed that the combination of parameters defining the peak position and width of the function, i.e., h and t, as well as the boundaries of the interval x/c in which the gradient information is utilized, generate an abrupt step that undermines the stability of the boundary layer. In a condition of this kind, the boundary layer transition is almost certainly triggered, completely nullifying the benefits that such a study could bring. The problem arises from the fact that, unlike a preferred basis of functions such as the sine function of the Fourier basis used in tolerance studies, the H-H bump functions do not necessarily vanish at the extremes of the interval in which they are defined if their parameters are specified in terms of a characteristic length (such as the chord of the wing profile). Consequently, introducing constraints to systematically avoid such situations is not easy at all, and it is therefore advisable to directly assess at reasonable intervals whether the algorithm is actually performing as intended.

In addition, another aspect that was not taken into account in Figure 11.2 needs to be considered. To address this, consider Figures 11.3–11.4; in these figures, the more complex and interesting case of gradient projection is considered, where instead of considering a single bump function, a certain set of them is taken into account. In this case, it is possible to leverage more the information coming from the resolution of the adjoint equations by defining the amplitude of each bump function; simplistically,

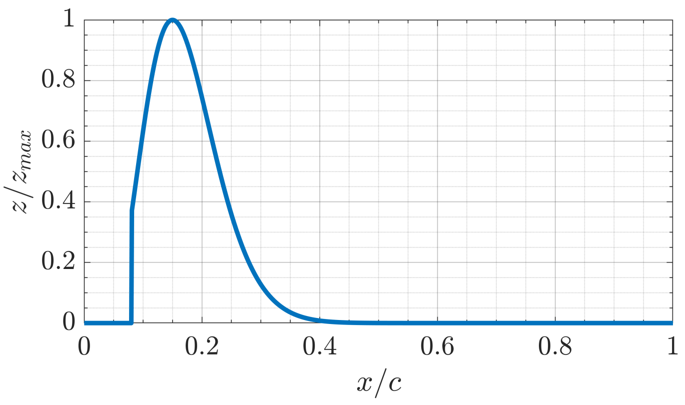
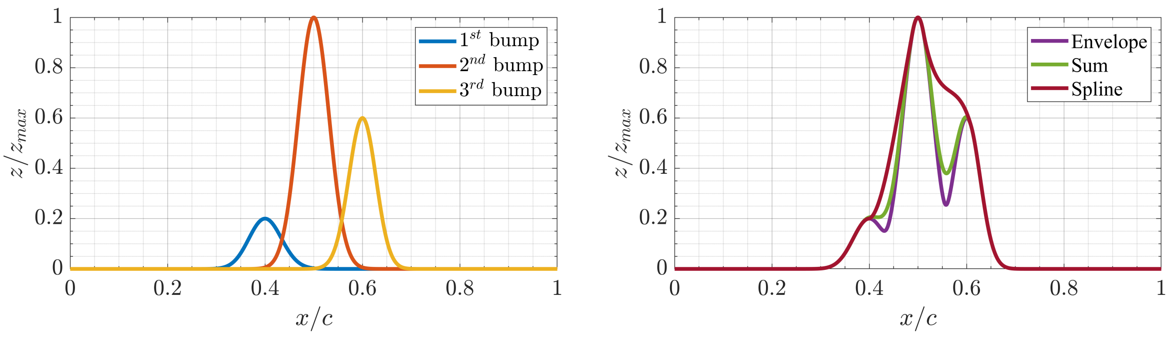


Figure 11.2 Situation arising from non-optimal positioning of bump functions: formation of an abrupt step near the leading edge. The peak position, i.e. h value, is set to 0.15, with a width parameter t of 0.40, and the region of the upper surface of the airfoil profile considered for modification is within the interval x/c=[0.08, 0.75].

one could say that in this case, it is possible to better replicate the shape of the gradient. Therefore, it is necessary to consider a methodology that allows deriving a reasonable and physically meaningful perturbation surface data from the superposition of the different shapes obtained. Once again, it is necessary to take into account what was presented in Figure 11.2; it could indeed occur that a single H-H function taken individually does not create problems regarding abrupt steps at the boundaries of the geometry modification interval, while the superposition of the different functions could cause it. Regarding the methodology, reference can be made to the right panels of the above-mentioned figures, which propose some of the solutions that were considered during the development of the code for gradient projection. In particular, three functions are considered: envelope, sum, and spline. The first considers the smooth curve that can be traced for an oscillating signal, such as $\nabla_z E$, outlining the extremes, the second is simply obtained as the sum of the different functions and subsequent normalization with the maximum value (in order to obtain a perturbation trend with unit peak), while the last one is obtained using a spline that connects the points of the maxima of the individual bump functions. Each of the three solutions presented has highlighted advantages and disadvantages: certainly, the first two functions are those that modify the original shape of the bump functions the least. However, it is not possible to overlook the fact that their composition leads to the formation of pronounced concave regions that can introduce negative effects on the stability of the boundary layer. In this regard, the results of the tolerance study come to our aid; it has been observed that moving towards APG conditions leads to an increase in the N-factor, i.e., an increase in the energy of the perturbations within the boundary layer. Furthermore, shapes of this type can certainly be responsible for the immediate transition of the boundary layer or even its separation. Given these observations, the third methodology is proposed, which traces a particularly smooth trend that approximates the general trend of the individual bump functions (to leverage the gradient information), while simultaneously avoiding the formation of concavities.

It is interesting to note at this point that it is particularly difficult to select in a general way which function is best suited for this purpose: the bump functions in Figure 11.3a are indeed very similar to those in Figure 11.4a; all parameters are unchanged except for the peak position of the 3^{rd} bump, which changes from x/c=0.6 to 0.55. It is then noted that a small variation like this can determine very different scenarios; in the first case, it is quite evident that, in order to try to delay the transition, the spline function is the most suitable, while in the second case, both the envelope and sum functions also provide good solutions.

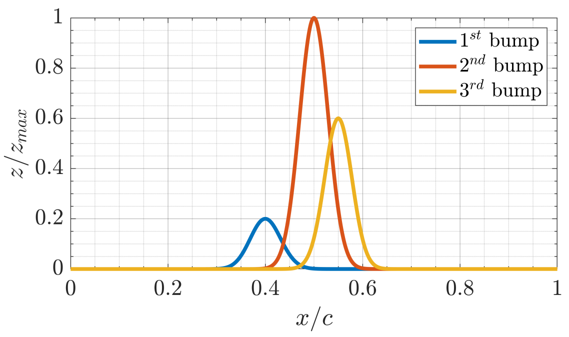


(a) Three different bump functions characterized by the following features, in the order of the legend: $h = [0.4, 0.5, 0.6]$, $t = [0.4, 0.5, 0.4]$, and amplitudes $[0.2, 1, 0.6]$.

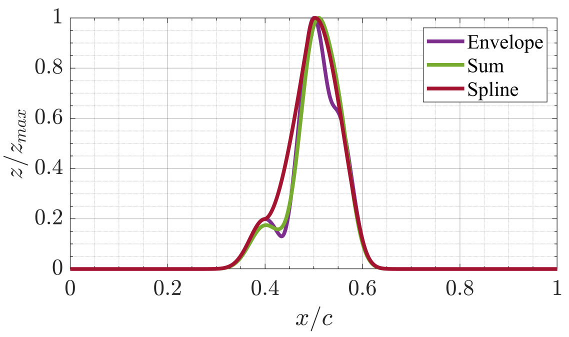
(b) Three different possible curves to consider as compositions of the selected bumps reported in the left panel for parameterizing the gradient are proposed: envelope, sum, and spline curve.

Figure 11.3 A possible situation that may arise during the sensitivity projection of the objective function with respect to surface deformation, i.e. $\nabla_z E$, into the space of chosen Hicks-Henne bump functions.

It can then be concluded that, in general, the method as developed requires some supervision from the user in order to consider, for the specific condition for which the optimization study is being performed, whether the setting of the input parameters regarding the positioning of the Hicks-Henne bump functions is reasonable or not. In this section, as already anticipated, bump functions chosen arbitrarily have been



(a) Three different bump functions characterized by the following features, in the order of the legend: $h = [0.4, 0.5, 0.55]$, $t = [0.4, 0.5, 0.4]$, and amplitudes $[0.2, 1, 0.6]$.



(b) Three different possible curves to consider as compositions of the selected bumps reported in the left panel for parameterizing the gradient are proposed: envelope, sum, and spline curve.

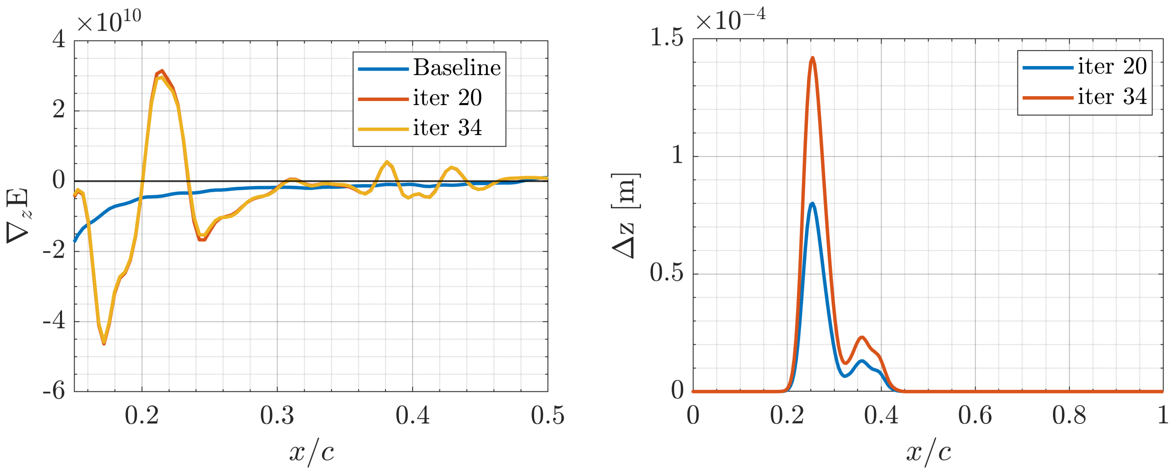
Figure 11.4 A possible situation that may arise during the sensitivity projection of the objective function with respect to surface deformation, i.e. $\nabla_z E$, into the space of chosen Hicks-Henne bump functions.

considered; in reality, the intention was precisely to consider the most critical scenarios that may arise to clearly highlight the issues for which a solution has been sought.

11.3 Example of optimisation cycles

Here are two examples among those considered during the testing phase of the developed algorithm. The results refer to the same flow condition, namely $Re = 9 \times 10^6$, $M = 0.5$, $AoA = 1.25°$, and the crossflow instability is characterized by the parameters reported in Table 11.1. The solutions obtained for the same flow condition are presented to highlight, among other aspects, the high sensitivity of the method to the position of the bumps along the upper surface of the airfoil, an aspect already mentioned in the previous section.

Consider now Figure 11.5–11.6, corresponding to the solution obtained after placing three bumps with parameters $h = [0.25, 0.36, 0.40]$ and $t = [0.30, 0.35, 0.20]$, which define respectively the position of the maximum of the individual function and their width. The positioning was based on the information from the $\nabla_z E$ trend shown in Figure 11.5a; it is observed that they were positioned at x/c coordinates for which the sensitivity with respect to the design variables is negative. In the same figure, the trend of $\nabla_z E$ after a certain number of iterations is also reported, specifically at the 20^{th} and 30^{th} iterations. At this point, the difficulties of implementing the method are observed; following the addition of bumps of extremely small sizes (note the scale of the ordinates in Figure 11.5b), the sensitivity changes from having a particularly smooth trend where it is clearly evident how the geometry should be modified, to a rather variable trend. The aspect that complicates all this the most is the fact that, following the introduction of bumps, in some positions the sensitivity changes from positive to negative values; therefore it could be possible that at a certain iteration, in the worst scenario after the second one, the selected location of a bump is, at all, no longer correct. In Figure 11.5b, the geometric perturbation introduced to the baseline geometry at the end of the iteration cycle can be seen.

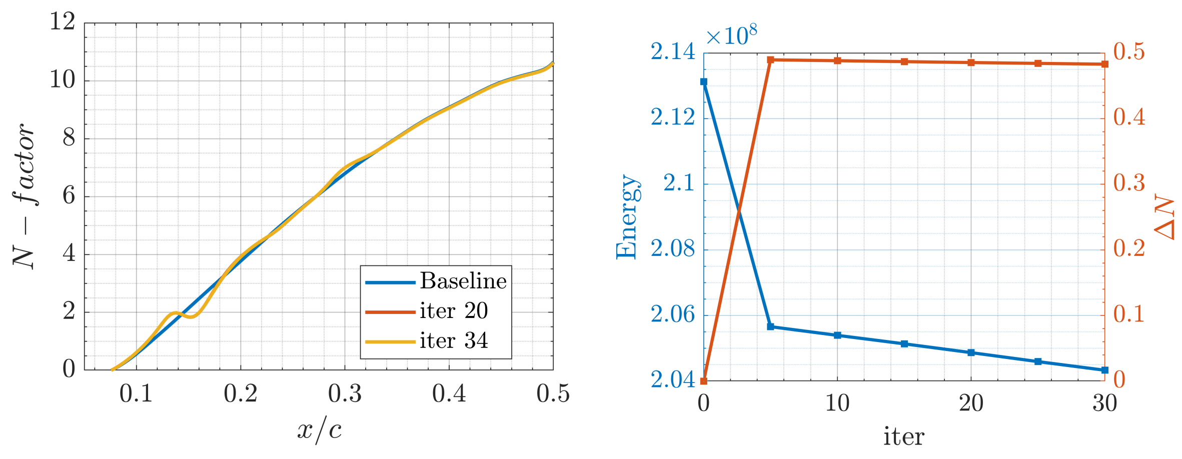


(a) $\nabla_z E$ as a function of the coordinate x/c for three different iterations.

(b) Shape of the perturbation of the baseline airfoil profile for three different iterations.

Figure 11.5 Flow conditions are $Re = 9 \times 10^6$, $M = 0.5$, $AoA = 1.25°$. The parameters of the HH bump functions employed are $h = [0.25, 0.36, 0.40]$ and $t = [0.30, 0.35, 0.20]$. The parameters of the instability mode considered are those reported in Table 11.1.

At this point, let us consider Figures 11.6, which allow us to understand the trend of the energy, or in particular to infer whether it is decreasing, indicating a damping of the instability chosen after the introduction of the bumps just presented. In Figure 11.5a, it is possible to observe a reduction in the N-factor at $x/c = 0.16$, which can be correlated in a certain way with the positioning of the bump closest to the LE. We observe a situation very similar to that commented on in the results of the tolerance study in the previous chapter. In particular, in the presence of APG, the crossflow instability tends to dampen; this damping can be evaluated, for example, through the value of ΔN between the current iteration and the initial iteration, as shown in Figure 11.6b. However, it is necessary to consider that this value of ΔN is calculated considering the position where there is the maximum reduction in the N-factor, namely for $x/c \approx 0.16$, and does not consider that at a more advanced position there is an increase in the same, which would result in a negative variation. Considering the purpose of this study, the fact just mentioned cannot be neglected, as such an increase in this parameter to higher absolute values could certainly lead to transition onset. It is also observed that the N-factor curves tend to overlap after a certain number of iterations.



(a) N-factor curve as a function of x/c for three different iterations.

(b) Trend of energy and ΔN as a function of the number of iterations.

Figure 11.6 Flow conditions are $Re = 9 \times 10^6$, $M = 0.5$, $AoA = 1.25°$. The parameters of the HH bump functions employed are $h = [0.25, 0.35, 0.40]$ and $t = [0.30, 0.35, 0.20]$. The parameters of the instability mode considered are those reported in Table 11.1.

In addition to considering the variation of the N-factor, essential in this calculation algorithm being the input condition and also its stopping criterion, it is useful to keep an eye on the energy value. It is observed that the latter decreases monotonically with the number of iterations, without reaching a plateau condition, which means that in general terms, the introduced bumps lead to a minimization of the objective function. However, the decrease is not significant as expected; to reach a satisfactory final condition multiple parameters have to be adjusted (bump position, overlap mode, etc.).

We can now consider the results shown in Figures 11.7–11.8, which as already mentioned refer to the same flow conditions but different positions of the bump functions. In particular, the parameters of the bumps are: $h = [0.20, 0.30, 0.50]$ and $t = [0.30, 0.30, 0.15]$; the characteristics of the functions chosen in this second case are therefore very similar to those of the previous case. Nevertheless, a quite different trend is observed as the number of iterations increases. In particular, observing Figure 11.8a, it can be noted that for this second setting, there is no increase in the N-factor before the introduction of the first bump, which was highlighted previously. Moreover, the N-factor curves are always scaled downwards as the iterations progress. However comparing the values of ΔN in Figures 11.8b–11.8b, in the first case, higher values are recorded; on the contrary, lower values for the energy are reached.

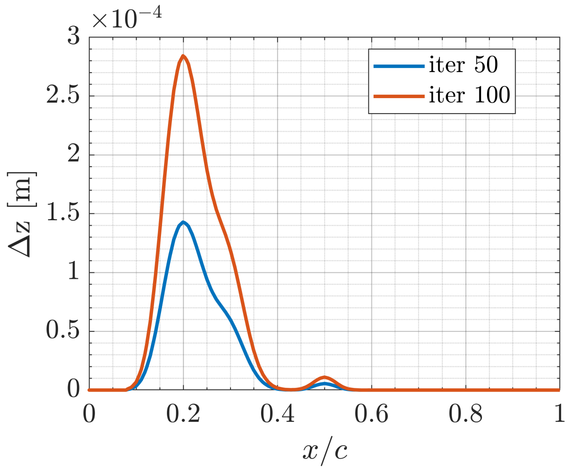
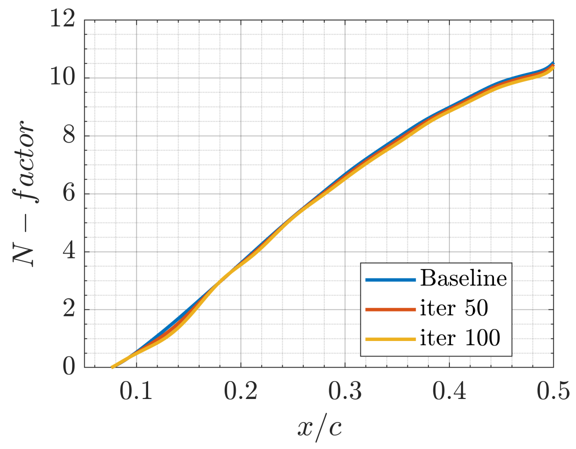
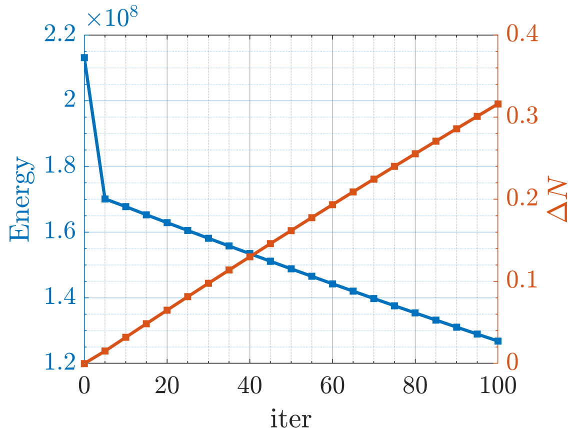


Figure 11.7 Shape of the perturbation of the baseline airfoil profile for three different iterations. Flow conditions are $Re = 9 \times 10^6$, $M = 0.5$, $AoA = 1.25°$. The parameters of the HH bump functions employed are $h = [0.20, 0.30, 0.50]$ and $t = [0.30, 0.30, 0.15]$. The parameters of the instability mode considered are those reported in Table 11.1.

As mentioned several times, the results presented in this section, although hinting at the possibility of optimizing the shape of an airfoil by introducing bumps described by Hicks-Henne functions, are not particularly satisfying. These brief comments, however, have been introduced in this work to hint at the numerous challenges that have been faced and that have required the search for solutions, as generalizable as possible in order to replicate a similar study under different flow conditions and baseline geometries. These preliminary results, however, are particularly useful in providing indications of aspects that require further attention and study. In particular, it will be necessary to understand why, in some conditions, there is an increase in the N-factor at positions advanced relative to the bumps introduced; this aspect could be challenging to understand. Furthermore, these results have highlighted the fact that, since $\nabla_z E$ in some cases changes from negative to positive values, the idea of fixing the position of the bump throughout the optimization cycle, as implemented in the developed algorithm, may not be the best solution. It may also be necessary to consider other solutions,



(a) N-factor curve as a function of x/c for three different iterations.



(b) Trend of energy and ΔN as a function of the number of iterations.

Figure 11.8 Flow conditions are $Re = 9 \times 10^6$, $M = 0.5$, $AoA = 1.25°$. The parameters of the HH bump functions employed are $h = [0.20, 0.30, 0.50]$ and $t = [0.30, 0.30, 0.15]$. The parameters of the instability mode considered are those reported in Table 11.1.

compared to that proposed here, for defining the shape of the bumps. In reference to this, the possibility of using a smoothing function to make the gradient of energy with respect to the z-coordinate physically meaningful and applying it directly to the baseline geometry has also been considered. Some simulations have been carried out, but once again, the oscillating behavior of the gradient makes it effectively impossible to make the optimization cycle completely automatizable; it is indeed required to verify at each iteration if there.

Certainly, one aspect that can greatly aid the development of the optimization algorithm is accessing the results of colleagues from TU Delft, thus being able to use their findings as a starting point for further consideration.

Conclusions

An approach based on the adjoint method was proposed to determine tolerances for surface waviness in the manufacturing of laminar flow (NLF) surfaces. One of the aims of this study was to extend the previous research conducted by Miniripiri *et al.*, shedding light on the case of instabilities arising in the presence of tapered aerodynamic surfaces. The mentioned study exclusively considers 2D instabilities, namely Tollmien-Schlichting type, as the problem referred to straight-wing configurations, therefore neglecting cross-flow velocity components. To properly extend the methodology to determine tolerances for a 3D wing, allowing deformations in both spanwise and chordwise directions, a comprehensive approach is required, necessitating the solution of the full 3D boundary layer and stability equations. However, while this approach may be feasible in some cases, it is significantly complex and computationally expensive. A simpler way to conduct such a study, albeit approximate compared to the aforementioned approach, is to assume that waviness occurs only in the chordwise direction; this is the solution which has been proposed here. To address the full three-dimensionality of the base flow, it can be handled in a similar manner to when designing and analyzing natural laminar flow wings. In this scenario, the flow at different wing sections along the span is treated separately, and a local infinite swept wing approximation is applied. This is feasible in the work frame already developed because part of the algorithm, namely codes for solving the boundary layer equations (BLE) and parabolized stability equations (PSE), are based on an infinite swept approximation and can accommodate both Tollmien-Schlichting and cross-flow perturbations. In this case, a manufacturing tolerance could be determined for each span section.
The intention to consider cross-flow velocity components using the open-source ADflow solver necessitated the implementation of the so colled 2.5D transformation to overcome the intrinsic limitation related to the absence of suitable boundary conditions. Specifically, to replicate the case of an infinite swept wing problem, periodic boundary conditions would have been required. However, due to the necessity of resorting to

symmetric boundary conditions, the introduction of a local reference system oriented in the normal-to-chord direction instead of the usual streamwise direction was necessary. The implementation of this treatment was proposed.
The objective of establishing manufacturing tolerances involves a computational process that includes obtaining the inviscid baseflow through the solution of the Euler equations, and the viscous meanflow by solving the boundary layer equations (BLE) for compressible flows. The stability of the boundary layer is assessed using the parabolized stability equations (PSE), and the gradient of the objective function concerning surface deformations is derived using the adjoint method through the solution of the adjoint equations corresponding to the Euler, BLE, and PSE.
The methodology for computing the surface waviness tolerance consists of an iterative gradient-based amplification of the kinetic energy of the boundary layer to find the wavy profile that causes a specified increase in the N-factor (ΔN). The NLF(2)-0415 airfoil from a 45° swept wing was chosen as the geometry considered in this study. However, it is important to underline that the methodology is applicable to any basic geometry, as the calculation algorithm has been appropriately modified to automate the mesh and geometry initialization process. This was one of the other objectives of the work carried out within the scope of this thesis. The results obtained using the gradient ascent (GA) method, show how tolerances scale with different Reynolds numbers, where the thickness of the boundary layer plays a significant role in the final tolerances together with the effect of different wavelengths of surface oscillations responsible for the level of adverse-favorable pressure gradients. The influence of other parameters was also considered: Mach number and angle of attack. To select the range of parameters to consider, typical conditions for aircraft with tapered wings were taken into account: Reynolds numbers between 9×10^6 and 15×10^6 based on common aerodynamic chords, Mach numbers between 0.45 and 0.60, and angles of attack ranging from $-1.00°$ to $1.75°$ in order to highlight how the characteristics of the instabilities vary.
The possibility of utilizing SLSQP has also been presented, which offers the advantage of being capable of finding the largest allowable deformation profile, with the minimum L^2-norm of surface deformation, albeit at the expense of additional computational time compared to the GA approach.
However, results regarding this alternative approach were not available at the time of submitting this work.
The outlined method aims to justify the efforts to constrain manufacturing tolerances

for NLF surfaces, with waviness shapes oriented along the chordwise direction, which are utilized in aerodynamic applications. The results from this study are believed to be particularly interesting during design process, as they allow for a direct link between the chosen, imposed (for various other reasons, ex. consider MDO[1] process), or required tolerance level, and the effect it has on the boundary layer development, namely the final ΔN resulting from the manufactured geometry. This will help engineers to be more confident about their design shape choices and aerodynamic performance.
Using a very similar approach to that proposed for the calculation of dimensional tolerances, an optimization method for optimizing the shape of a profile to delay boundary layer transition was then proposed. The methodology involves an iterative gradient-based reduction of the kinetic energy of the boundary layer to identify the shape that induces a specified decay of a certain perturbation, which correlates with the decrease in the N-factor. The approach to performing the flow solution and stability analysis remained the same as the one previously proposed; even the flow conditions were equal, once again considering the approximation of the infinite swept wing problem to avoid dealing with full three-dimensionality. Even the airfoil profile and the sweep angle of the wing used were the same, i.e. NLF(2)-0415 for 45° swept condition. The main difference lies in how the information from the sensitivity analysis of the adjoint solution is utilized. Given that the aim of this study is to find a local shape deformation advantageous for delaying transition, the focus was on introducing perturbations which reduce energy within the boundary layer, potentially directly linked to decreasing the N-factor. As the shape of sensitivity with respect to design variables is typically non-physical, i.e., not directly applicable to geometry, it was considered to parameterize its shape by projecting it into the space of a certain number of Hicks-Henne bump functions. The related implementation was proposed, along with the option of considering a simpler way to utilize information from the adjoint solution using an in-house smooth function.
Greater attention was paid to the theoretical definition and numerical implementation of the methodology rather than the analysis of the results, as obtaining promising results required a greater effort. Nevertheless, the method has shown promise, and it is planned to continue its implementation in the future to complete the work. In particular, greater clarity will be needed on the upstream effect of introducing surface perturbation, where an increase in the N-factor has been highlighted and representing a condition contrary to the desired outcome. It is known that the physics involved in

[1]Multidisciplinary design optimization

boundary layer transition is particularly complex, requiring a more focused approach and careful attention to all the factors involved. In any case, this work can certainly be seen as the starting point.

Acknowledgements

After completing the conclusions of a scientific work, it might seem like the most challenging part is over. For many, that may be the case, but for me, only partially so. Maybe it is the difficulty of putting my thoughts down on paper, the fear of forgetting someone, or, even worse, expressing gratitude insincerely. In conclusion, I believe that one of the most demanding parts is expressing thanks.

I would like to deeply thank the people who have made a significant contribution to the drafting of this work.
I thank Prof. Alessandro Bottaro for his support and professionality, but above all for giving me the opportunity to get in touch with Prof. Ardeshir Hanifi, a delightful person, always available and kind. I am very happy to have met you; I have perceived your indisputable scientific knowledge as well as the importance you give to personal relationships.

We always thank too little and too late!
Since I read this sentence, actually, writing acknowledgments has become a bit simpler. I hope I have done it enough and will be able to do it better in the future: thanking the people who have helped me, supported me, advised me, or simply kept me company at the very moment it happened, without necessarily waiting for the end of something, such as this thesis.
I hope you have (and will) appreciated and understood this way of thanking you.
So I will avoid listing here, name by name, the many people I have met in recent years, with whom I have shared moments and experiences that will stay with me for a long time.
Just few words.
I want to thank my friends from Genoa who welcomed me, encouraged and stimulated me. I hope that we all continue to carry this flame in the future despite the distance, which has shortened more and more over these two years.

I cannot forget my "nordic" friends from OSB18 and beyond, met during my stay in Stockholm. That Swedish experience was extraordinary: an educational, enjoyable, unique stay, and at times also a bit stressful.

A nod also to lifelong friends: us vuei ben! No stait a dismenteâsi di me!

And almost like closing a loop, I would like to thank those who provided me with the means, the tools, and even the support to embark on this journey that is now reaching a crossroads. My family: my mother, my father, and my brother Carlo. The effort I have devoted to this journey has been significant, necessary to express in the best way my sincere and immense gratitude for your palpable push.

That’s it!

"E quando penso che sia finita
È proprio allora che comincia la salita
Che fantastica storia è la vita..."
A. Venditti

References

[1] M. Moniripiri, P. C. Brito, A. V. G. Cavalieri, N. R. Sêcco, and A. Hanifi. An adjoint-based methodology for calculating manufacturing tolerances for natural laminar flow airfoils susceptible to smooth surface waviness. *Theoretical and Computational Fluid Dynamics*, 12 2023.

[2] O. Amoignon. Adjoint-based aerodynamic shape optimization, 2003.

[3] F. P. Bertolotti, Th. Herbert, and P. R. Spalart. Linear and nonlinear stability of the Blasius boundary layer. *Journal of Fluid Mechanics*, 242:441–474, September 1992.

[4] J. O. Pralits, C. Airiau, A. Hanifi, and D. S. Henningson. Sensitivity analysis using adjoint parabolized stability equations for compressible flows. *Flow Turbulence and Combustion*, 65(04-mar):321–346, 2000. QC 20100525.

[5] J. Van Ingen. The en method for transition prediction. historical review of work at tu delft. In *38th Fluid Dynamics Conference and Exhibit*, page 3830, 2008.

[6] A. Hanifi and D. S. Henningson. *Stability of Boundary Layer Flows*, pages 51–103. Springer Netherlands, Dordrecht, 1999.

[7] J. O. Pralits and A. Hanifi. Optimization of steady suction for disturbance control on infinite swept wings. *Physics of Fluids*, 15(9):2756–2772, 09 2003.

[8] J. O. Pralits, A. Hanifi, and D. S. Henningson. Adjoint-based optimization of steady suction for disturbance control in incompressible flows. *Journal of Fluid Mechanics*, 467:129–161, 09 2002.

[9] G. W. Kenway, C. A. Mader, H. Ping, and J. R. R. A. Martins. Effective adjoint approaches for computational fluid dynamics. *Progress in Aerospace Sciences*, 110:100542, October 2019.

[10] O. Amoignon, J. O. Pralits, A. Hanifi, M. Berggren, and D. S. Henningson. Shape optimization for delay of laminar-turbulent transition. *AIAA Journal*, 44(5):1009–1024, 2006. QC 20120511.

[11] C. A. Mader, K. W. Gaetan, Anil Y., and J. R. R. A. Martins. ADflow—an open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, 2020.

[12] A. Yildirim, Gaetan K. W., C. A. Mader, and J. R. R. A. Martins. A jacobian-free approximate newton–krylov startup strategy for rans simulations. *Journal of Computational Physics*, 397:108741, nov 2019.

[13] S. Hein, F.P. Bertolotti, M. Simen, A. Hanifi, and D. S. Henningson. Linear nonlocal instability analysis - the linear nolot code -. Technical report, 1995. LIDO-Berichtsjahr=1995,.

[14] United States. National Oceanic, Atmospheric Administration, and United States Committee on Extension to the Standard Atmosphere. *U.S. Standard Atmosphere, 1976*. NOAA - SIT 76-1562. National Oceanic and Amospheric [sic] Administration, 1976.

[15] Construct 2D. https://sourceforge.net/projects/construct2d/. Accessed: September, 2023.

[16] Z. Tong, Y. Zhang, C. Haixin, C. Yingchun, and Z. Miao. Supercritical wing design based on airfoil optimization and 2.75d transformation. *Aerospace Science and Technology*, 56:168–182, 2016.

[17] Ney Secco, Gaetan Kenway, Ping He, Charles Mader, and Joaquim Martins. Efficient mesh generation and deformation for aerodynamic shape optimization. *AIAA Journal*, 59:2020, 10 2020.

[18] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[19] X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. B$a \ldots b$ck, V. Dunjko, and T. E. O'Brien. Performance comparison of optimization methods on variational quantum algorithms. *Physical Review A*, 107(3):032407, 2023.

[20] E. Sharifi, A. Akhavan Taheri Borojeni, and M. H. Hekmat. Investigation of the adjoint method in aerodynamic optimization using various shape parameterization techniques. *Journal of the Brazilian Society of Mechanical Sciences*, XXXII:176–186, 06 2010.

[21] The MathWorks Inc. Filtering and smoothing data, 2023.

[22] M. Drela. An analysis and design system for low reynolds number airfoils. In Thomas J. M., editor, *Low Reynolds Number Aerodynamics*, pages 1–12, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[23] Hossein Haj-Hariri. Characteristics analysis of the parabolized stability equations. *Studies in Applied Mathematics*, 92(1):41–53, 1994.

[24] Fei Li and Mujeeb R Malik. On the nature of pse approximation. *Theoretical and Computational Fluid Dynamics*, 8(4):253–273, 1996.

[25] P. Andersson, D. S. Henningson, and A. Hanifi. On a stabilization procedure for the parabolic stability equations. *Journal of Engineering Mathematics*, 33(3):311–332, 1998.

[26] D. Arnal and Olivier Vermeersch. Compressibility effects on laminar-turbulent boundary layer transition. *Int. J. of Engineering Systems Modelling and Simulation*, 3, 03 2011.

[27] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Appendix A

Code A.1 Bash script for the automatic generation of the mesh with optional 2.5D transformation of the geometry using the Construct2D code, starting from the profile geometry in .dat or .cgns file format.

```bash
# ====================================================================
# From UNswept to swept, mesh generation (INIT)
# ====================================================================
echo "**********************************************"
echo "----------------------------------------------"
echo "Init Transformation2.5D/Meshing from streamwise to normal-to-chord
    direction"
echo "----------------------------------------------"
echo "----------------------------------------------"

# Extract the file extension
extension="${InputFileName##*.}"
# Conditions for file extension
if [[ "$extension" == "dat" ]]; then
    PythonScript="coord_from_dat.py"
elif [[ "$extension" == "cgns" ]]; then
    PythonScript="coord_from_cgns.py"
else
    echo "Error: Unknown file extension."
    exit 1  # Terminate the script with a non-zero exit code
fi
current_dir=$(pwd)
cd mesh || exit 1

# Run Python script to generate modified coordinates
python3 "$PythonScript" "$InputFileName" "$Sweptangle"
file_path="coords_swept_${Sweptangle}deg.dat"
```

```
num_lines=$(awk 'NR > 1 {count++} END {print count}' "$file_path")
nsfr=$num_lines
# Overwrite values in .in file
sed -i "s/^\s*nsrf\s*=\s*.*/  nsrf = $nsfr/" "grid_options.in"
# Read nwke value from grid_options.in and calculate new value for b
nwke=$(awk '/^ *nwke *=/ {print $3}' "grid_options.in")
index=$((nsfr+nwake))
# If command line arguments are provided, use them for num_lines and b
if [ $# -eq 2 ]; then
    num_lines="$1"
    index="$2"
fi
nsfr=$(($nsfr + $nwke ))
# Increment the value of nwke by 1
nwke=$((nwke + 1))
# Update values in info_split.txt using awk
awk -v new_nwke="$nwke" -v new_nsfr="$nsfr" 'NR==1{$3=new_nwke} NR==2{$3
    =new_nsfr} 1' info_split.txt > temp.txt && mv temp.txt info_split.txt

construct2d <<EOF
coords_swept_${Sweptangle}deg.dat
grid
buff
quit
EOF

# Convert plot3d to CGNS format
plot3d_to_cgns -f -d naca.p3d NLF_N600_3D.cgns
# Split CGNS file and overwrite boundary conditions
cgns_utils split --splitFile info_split.txt NLF_N600_3D.cgns
    NLF_N600_3D_splited.cgns
cgns_utils overwriteBC NLF_N600_3D_splited.cgns info_bc.txt NLF_N600.
    cgns
# ======================================================================
# From UNswept to swept, mesh generation (END)
# ======================================================================
```

Code A.2 Python script for performing the 2.5D coordinate transformation starting from the airfoil geometry into mesh files in .cgns format.

```
# ====================================================================
# Import modules
# ====================================================================
import numpy as np
from mpi4py import MPI
from idwarp import *
import sys
import matplotlib.pyplot as plt
from math import cos, radians
import glob
# ====================================================================
# Input Information
# ====================================================================
#Load file name
file_name=sys.argv[1]
#---------------------------------------------------------------------
# Define the swept angle in degrees
swept_angle = float(sys.argv[2])
# ====================================================================
matching_files = glob.glob(file_name)
# Check if there are matching files
if matching_files:
    name_file = matching_files[0]
    outputDirectory = './'
    # Create a mesh object
    mesh = USMesh(options={'gridfile': file_name})

    print("*" * 40)
    print(f"File {file_name} loaded successfully")
    print("*" * 40)
    # Get surface coordinates
    original_coords = mesh.getSurfaceCoordinates()
    print(max(original_coords[:,0]))
    print(max(original_coords[:,1]))
    print(max(original_coords[:,2]))
    # Save surface coordinates to a .dat file
    output_file_path = 'original_coords.dat'
    np.savetxt(output_file_path, original_coords, fmt='%.16e', delimiter
   =' ')

```

```python
    print("*" * 40)
    print(f"Init 2.5d coordinates transformation")
    print("*" * 40)

    # Convert the swept angle to radians and calculate the cosine
    scaling_factor_y = cos(radians(swept_angle))
    new_coords = original_coords.copy()
    new_coords[:, 2] /= scaling_factor_y

    # Find the minimum value in the second column to consider only one airfoil, doing this only root (in .cgns tip and root airfoils)
    min_y_value = np.min(new_coords[:, 1])
    # Filter rows where the value in the second column is equal to the minimum
    filtered_data = new_coords[new_coords[:, 1] == min_y_value]
    # Swap the second and third columns (y and z)
    filtered_data[:, 1], filtered_data[:, 2] = filtered_data[:, 2].copy(), filtered_data[:, 1].copy()
    # Save surface coordinates to a .dat file with header
    output_file_path = f'coords_swept_{swept_angle:.2f}deg.dat'
    header = f"x y_modified z sweep_angle={swept_angle:.2f}"
    np.savetxt(output_file_path, filtered_data, fmt='%.16e', delimiter=' ', header=header)

    print("*" * 40)
    print(f"End 2.5d coordinates transformation. File saved")
    print("*" * 40)
else:
    print("*" * 40)
    print(f"No files matching name {file_name} found.")
    print("*" * 40)
```

Code A.3 Python script for performing the 2.5D coordinate transformation starting from the airfoil geometry into files in .dat format (x, y, z).

```python
# ======================================================================
# Import modules
# ======================================================================
import sys
import numpy as np
from math import cos, radians
import glob
# ======================================================================
# Input Information
# ======================================================================
# Load file name
file_name = sys.argv[1]
# ----------------------------------------------------------------------
# Define the swept angle in degrees
swept_angle = float(sys.argv[2])
# ======================================================================
def read_airfoil_coordinates(file_path):
    x, y, z = np.loadtxt(file_path, unpack=True, skiprows=1)
    return x, y, z
def save_modified_coordinates(file_path, x, y_scaled, z_scaled):
    with open(file_path, 'w') as file:
        for xi, yi_scaled, zi_scaled in zip(x, y_scaled, z_scaled):
            file.write(f"{xi:.15f} {yi_scaled:.15f} {zi_scaled:.15f}\n")
matching_files = glob.glob(file_name)
# Check if there are matching files
if matching_files:
    name_file = matching_files[0]
    print("*" * 40)
    print(f"File {file_name} loaded successfully. Init 2.5d coordinates
   transformation")
    print("*" * 40)
    # Read baseline airfoil coordinates
    x, y, z = read_airfoil_coordinates(name_file)
    # y is scaled considering the sweep angle (2.5D transformation)
    y_scaled = y / np.cos(np.radians(swept_angle))
output_file_path = f'coords_swept_{swept_angle:.2f}deg.dat'
with open(output_file_path, 'w') as file:
    file.write("# x y_modified z sweep_angle={:.2f}\n".format(
   swept_angle))
    for xi, yi_scaled, zi in zip(x, y_scaled, z):
        file.write(f"{xi:.16e} {yi_scaled:.16e} {zi:.16e}\n")
print("*" * 40)
print(f"End 2.5d coordinates transformation. File saved")
print("*" * 40)
```

Code A.4 Python script for smoothing the desired quantity into the geo.dat_1001 file. Users can choose between smoothing the C_p and the geometry of the airfoil.

```python
import numpy as np
import matplotlib.pyplot as plt
import sys

Sweptangle = float(sys.argv[1])
smooth_quantity = float(sys.argv[2])

# Load data from 'geo.dat_1001'
geo_data = np.loadtxt('geo.dat_1001', dtype=np.float64, skiprows=3)
# Copy the first three lines
header_lines = []
with open('geo.dat_1001', 'r') as file:
    for i in range(3):
        header_lines.append(next(file))
# Read data from 'header.txt' and insert it as the second row
with open('header.txt', 'r') as header_file:
    header_data = header_file.readline().strip()

# Calculate x and y
x = geo_data[:, 0] * np.cos(np.radians(Sweptangle))
y = geo_data[:, 1]
# Reshape x and y as column vectors
x = x.reshape(-1, 1)
y = y.reshape(-1, 1)

def myfilter(s, y, n):
    s = s.reshape(-1, 1)
    imax = len(s)
    mmax = y.shape[1]
    yout = y.copy()

    for j in range(n):
        for m in range(mmax):
            for k in range(1, imax-1):
                ds1 = (s[k] - s[k-1]) / (s[k+1] - s[k-1]) * 0.5
                ds2 = (s[k+1] - s[k]) / (s[k+1] - s[k-1]) * 0.5
                yout[k, m] = (yout[k-1, m] * ds2 + 4 * yout[k, m] + yout
    [k+1, m] * ds1) / (4.0 + ds2 + ds1)

    return yout
```

```

dx = np.diff(x[:, 0])
dy = np.diff(y[:, 0])
s = np.concatenate([[0], np.cumsum(np.sqrt(dx**2 + dy**2))])

# Apply myfilter to smooth the respective column of smooth_quantity
smth = myfilter(s, geo_data[:, int(smooth_quantity)].reshape(-1, 1), 20)
# Create a copy of the original data
geo_smoothed = geo_data.copy()
# Overwrite the third column of the copy with the smoothed values
geo_smoothed[:, int(smooth_quantity)] = smth.flatten()
# Save the smoothed data into 'geo.dat_1001' with headers and inserted
    data
with open('geo.dat_1001', 'w') as f:
    f.writelines(header_lines[0])
    f.writelines(header_data + '\n')
    f.writelines(header_lines[2])
    np.savetxt(f, geo_smoothed, delimiter=' ', fmt='%.16f')
```

Code A.5 Python script for re-organizing the output file of ADflow, geo.dat_1001, to prepare it as input for BL3D code. It identifies the coordinates of the stagnation point on the airfoil and separates the upper and lower surface flow solutions.

```python
# ======================================================================
# Import modules
# ======================================================================
import numpy as np
import sys
# ======================================================================
# Input Information
# ======================================================================
iter=float(sys.argv[1])
Re=float(sys.argv[2])
Mode=float(sys.argv[3])

header=np.loadtxt('header.txt',dtype=np.float64)
apreynoldsLength=header[0]
apreynolds=header[1]
apmach=header[2]
apT=header[3]
apSwept=header[4]
# ----------------------------------------------------------------------
# Open output file
f=np.loadtxt('geo_all.dat', dtype=np.float64)
f=f[0:-1,:] #exclude duplicated rows  (first and last lines)
# ----------------------------------------------------------------------
# Find stagnation point and trailing edge
X=f[:,0]
Cp=f[:,2]
istag = np.argmax(Cp)
imax= np.argmax(X)
# ----------------------------------------------------------------------
# Extract upper and lower surface data. Assumes data in geo_all.dayt is
    sorted.
f1=[]
f2=[]
if istag<imax:
 f1=f[istag:imax+1,:]
 f2=f[istag:-1:-1,:]
 f2=np.concatenate((f2,f[imax+1:,:]), axis=0)
else:
 f1=f[istag:imax:-1,:]
 f2=f[istag:,:]
 f2=np.concatenate((f2,f[0:imax+1,:]), axis=0)
# ------------------------------------------------------------------
# Make sure f1 corresponds to data on uppersurface.
# It assumes that y-coordinate on upper side increases when moving from
    stagnation point.
if f2[1,1]-f2[0,1]>0:
 f0=f1
 f1=f2
 f2=f0
# ------------------------------------------------------------------
# Writing BL file upper surface
f=open('geo.dat_1001','w')
f.write("#  Lref                RE                     MACH
      Tinf                 Sweep \n")
f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (apreynoldsLength,
     apreynolds, apmach, apT, apSwept))
f.write("#   x/Lref          y/Lref          Cp            Cq          Tw
     \n")
for i in range(np.size(f1,0)):
    f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (f1[i,0],f1[i
     ,1],f1[i,2], 0.0, 0.0))
f.close()

f=open('geo.dat_Nf%d_Re%2.8d_I%2.3d' %(Mode,Re,iter),'w')
f.write("#  Lref                RE                     MACH
      Tinf                 Sweep \n")
f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (apreynoldsLength,
     apreynolds, apmach, apT, apSwept))
f.write("#   x/Lref          y/Lref          Cp            Cq          Tw
     \n")
for i in range(np.size(f1,0)):
    f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (f1[i,0],f1[i
     ,1],f1[i,2], 0.0, 0.0))
f.close()

f=open('geo.dat','w')
for i in range(np.size(f1,0)):
    f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (f1[i,0],f1[i
     ,1],f1[i,2], 0.0, 0.0))
f.close()
```

```
# ---------------------------------------------------------------------
# Writing BL file lower surface
#f=open('geo.dat_1002','w')
#f.write("#  Lref               RE                     MACH
      Tinf                 Sweep \n")
#f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (apreynoldsLength
   , apreynolds, apmach, apT, apSwept))
#f.write("#   x/Lref          y/Lref          Cp            Cq
   Tw \n")
#for i in range(np.size(f2,0)):
#   f.write("%5.16f %20.16f %20.16f %20.16f %20.16f \n" % (f2[i,0],f2[i
   ,1],f2[i,2], 0.0, 0.0))
#f.close()

print('---------------------------------------')
print('Cp max:',f1[0,2])
print('---------------------------------------')
```

Code A.6 Python script for projecting the sensitivity output of ADflow into the space of selected Hicks-Henne bump functions for the optimization study regarding the delay of the transition of the boundary layer over a NLF airfoil.

```
# ====================================================================
#          Import modules
# ====================================================================
import numpy
import scipy
import sys
import os
import matplotlib.pyplot as plt

from subprocess import call
zeroLE=float(sys.argv[1])
normTE=float(sys.argv[2])
Nmode=float(sys.argv[3])
iteration=float(sys.argv[4])

iter = iteration
zero_LE = zeroLE    #position until which we zero the LE sensitivity
norm_TE = normTE    #position until which we normalize the sensitivity
    -> we normalize by the |max_val| between (zero_LE) & (norm_TE)

sens=numpy.loadtxt('sens_cut.dat',dtype=numpy.float64)
x = sens[:,0]
sens_raw = sens[:,1]
#-------------------------------------------------------------------
# Definition of HH-bump functions
def hickshenne(n, a, h, t, SF):
    x_new = numpy.linspace(0, 1, n)
    m = numpy.log(0.5) / numpy.log(h)
    b = numpy.zeros_like(x_new)

    for i in range(len(x_new)):
        for j in range(len(a)):
            b[i] += SF * a[j] * numpy.sin(numpy.pi * x_new[i]**m)**t

    return b

# Input data
dx_new = 0.01
```

```python
h = [0.15, 0.20, 0.25]
target_width = [0.2, 0.3, 0.2]
threshold = 1e-5
x_new = numpy.arange(0, 1 + dx_new, dx_new)
t_calc = [] # Initialize an empty list to store calculated t values
HH = [] # Initialize an empty list to store desired bump funtions
HH_interp = []  # Initialize an empty list to store interpolated bump funtions

# Span over lots of t different values to find a good set of parameters for the bump function desired
t_values = numpy.arange(1, 501, 1)

for j in range(len(h)):
    h_value = h[j]
    delta_values = []

    for i in range(len(t_values)):
        t = t_values[i]
        y = hickshenne(len(x_new), [1], h_value, t, 1)
        # Threshold operation
        y[y < threshold] = 0
        non_zero_indices = numpy.where(y != 0)[0]
        first_index, last_index = non_zero_indices[0], non_zero_indices[-1]
        delta_values.append(abs(x_new[last_index] - x_new[first_index]))

    index = numpy.argmin(numpy.abs(numpy.array(delta_values) - target_width[j]))
    closest_delta_x = delta_values[index]
#   print(f'The value of delta x closest to {target_width[j]} is: {closest_delta_x}')
#   print(f'Percentage error is: {(abs(closest_delta_x - target_width[j]) / target_width[j] * 100)}')
    t = t_values[index]
    t_calc.append(t)    # Store the calculated t value in the list
    # Create the selected bump function considering the position of the max of the function and width
    HH_single = hickshenne(len(x_new), [1], h_value, t, 1)
    HH.append(HH_single)
    # Create the selected bump function considering the position of the max of the function and width
    HH_interp.append(numpy.interp(x, x_new, HH_single))
# Transpose HH_interp
HH_interp_transposed = numpy.transpose(HH_interp)

# End of definition of HH-bump functions
#---------------------------------------------------------------------
#Interpolating data
x_init = x[0]
x_end  = x[-1]
N = len(x)

pos_init = next(k for k, val in enumerate(x) if val>zero_LE)
pos_end = next(k for k, val in enumerate(x) if val>norm_TE)

HH_interp_transposed_f = HH_interp_transposed[pos_init:pos_end+1, :]
xf = (x[pos_init:pos_end+1]-x[pos_init])/(x[pos_end]-x[pos_init])
sens_raw_f = sens_raw[pos_init:pos_end+1]
#---------------------------------------------------------------------
# Applying H-H bump filter
s_full = numpy.zeros(N)
s_full_single = numpy.zeros((N, len(h)))
s_serie = numpy.zeros_like(HH_interp_transposed_f)

bn_values = []
for i in range(len(h)):
    bn = numpy.trapz(sens_raw_f * HH_interp_transposed_f[:,i], xf)
    # Append bn to the array
    bn_values.append(bn)
bn_values = numpy.array(bn_values)
print("bn_values")
print(bn_values)

filename = 'HH_bn_coeff.txt'
if iter == 0:
    bn_values_with_iter = numpy.insert(bn_values, 0, iter)
    bn_values_with_iter = numpy.reshape(bn_values_with_iter, (1, -1))
    # Ensure it's a row vector
    # Generate header with dynamic column names
    num_coeffs = bn_values_with_iter.shape[1] - 1
    header = '\t'.join(['iter'] + [f'coeff{i}' for i in range(1,
    num_coeffs + 1)])
    fmt_string = '%d\t' + '\t%.16e' * num_coeffs
    numpy.savetxt(filename, bn_values_with_iter, delimiter='\t', fmt=fmt_string, header=header)
else:
    # Load existing data from file
    existing_data = numpy.loadtxt(filename)
    # Add the iteration number as the first column
    bn_values_with_iter = numpy.insert(bn_values, 0, iter)
    bn_values_with_iter = numpy.reshape(bn_values_with_iter, (1, -1))  # Ensure it's a row vector
    # Add the new values as the last row
    all_data = numpy.vstack((existing_data, bn_values_with_iter))
    # Generate header with dynamic column names
    num_coeffs = bn_values_with_iter.shape[1] - 1
    header = '\t'.join(['iter'] + [f'coeff{i}' for i in range(1, num_coeffs + 1)])
    fmt_string = '%d\t' + '\t%.16e' * num_coeffs
    # Save the combined data back to the file
    numpy.savetxt(filename, all_data, delimiter='\t', fmt=fmt_string, header=header)

for i in range(len(h)):
    s_serie[:, i] = HH_interp_transposed_f[:, i] * bn_values[i]

sum_bumps = numpy.sum(s_serie, axis=1)

#-----------------------------------------------------------------------
# Create the sum_bumps of bump functions vector with zeros out of the modify region
s_full[pos_init:pos_end+1] = sum_bumps
#-----------------------------------------------------------------------
# Normalizing the sum_bumps of bump functions
s_full_norm = s_full/max(abs(s_full))
#-----------------------------------------------------------------------
#Writting Normalized Filtered Sensitivity
f=open('sens_filt_HH_norm.dat','w')
for i in range(len(x)):
    f.write("%5.16f %5.16f %5.16f %5.16f\n" % (x[i], sens_raw[i], s_full[i], s_full_norm[i]))
f.close()

f=open('sens.dat'+str(iter),'w')
for i in range(len(x)):
    f.write("%5.16f %5.16f %5.16f %5.16f\n" % (x[i], sens_raw[i], s_full[
     i], s_full_norm[i]))
f.close()
```

Appendix B

Code B.1 grid_options.in

```
&SOPT
  nsrf = 600
  lesp =     5.0000000000000002E-003
  tesp =     5.0000000000000001E-003
  radi =     150.00000000000000
  nwke =              50
  fdst =     1.0000000000000000
  fwkl =     1.0000000000000000
  fwki =     10.000000000000000
/
&VOPT
  name = 'naca'
  jmax =            100
  slvr = 'HYPR'
  topo = 'CGRD'
  ypls =     5.000000000000000
  recd =     100000.0000000000
  stp1 =            1000
  stp2 =              20
  nrmt =               1
  nrmb =               1
  alfa =     1.0000000000000000
  epsi =     15.000000000000000
  epse =     0.0000000000000000
  funi =     0.0000000000000000
  asmt =              20
/
```

```
&OOPT
   gdim =                    3
   npln =                    2
   dpln =      1.0000000000000000
/
```

Code B.2 info_bc.txt

```
1 ilow BCFarfield farfield
1 jhigh BCSymmetryPlane sym
1 jlow BCSymmetryPlane sym
1 khigh BCFarfield farfield
2 klow BCWallviscous wall
2 khigh BCFarfield farfield
2 jhigh BCSymmetryPlane sym
2 jlow BCSymmetryPlane sym
3 jhigh BCSymmetryPlane sym
3 jlow BCSymmetryPlane sym
3 ihigh BCFarfield farfield
3 khigh BCFarfield farfield
```

Code B.3 info_split.txt

```
1 1 41
1 1 640
```

Appendix C

In the following, we provide an overview of the settings used for the ADflow and NOLOT codes (for direct and adjoint solution).

Code C.1 Settings used in the ADflow solver for performing the simulations (direct and adjoint) presented in this work. These settings are specified in the codes named Coupling_run_0.py and Coupling_Script_C2D_adj.py.

```
# ADflow, multiblock structured flow solver
#
# This code solves the 3D RANS, laminar NS or Euler equations
# on multiblock structured hexahedral grids.
# This is a parallel executable running on 2 processors.
# It has been compiled with the following options:
# - Optimized mode.
# - Size of standard integers: 4 bytes.
# - Size of standard floating point types: 8 bytes.
# - With cgns support
# - With support for signals.
#
+----------------------------------------------+
|            All ADFLOW Options:               |
+----------------------------------------------+
{'ADPC': False,
 'ANKADPC': False,
 'ANKAMGLevels': 2,
 'ANKAMGNSmooth': 1,
 'ANKASMOverlap': 1,
 'ANKCFL0': 5.0,
 'ANKCFLCutback': 0.5,
 'ANKCFLExponent': 0.5,
 'ANKCFLFactor': 10.0,
 'ANKCFLLimit': 100000.0,
 'ANKCFLMin': 1.0,
 'ANKCFLReset': True,
 'ANKCharTimeStepType': 'None',
 'ANKConstCFLStep': 0.4,
 'ANKCoupledSwitchTol': 1e-16,
 'ANKGlobalPreconditioner': 'additive Schwarz',
 'ANKInnerPreconIts': 1,
 'ANKJacobianLag': 10,
 'ANKLinResMax': 0.1,
 'ANKLinearSolveBuffer': 0.01,
 'ANKLinearSolveTol': 0.05,
 'ANKMaxIter': 40,
 'ANKNSubiterTurb': 1,
 'ANKOuterPreconIts': 1,
 'ANKPCILUFill': 2,
 'ANKPCUpdateCutoff': 1e-16,
 'ANKPCUpdateTol': 0.5,
 'ANKPCUpdateTolAfterCutoff': 0.0001,
 'ANKPhysicalLSTol': 0.2,
 'ANKPhysicalLSTolTurb': 0.99,
 'ANKSecondOrdSwitchTol': 1e-16,
 'ANKStepFactor': 1.0,
 'ANKStepMin': 0.01,
 'ANKSubspaceSize': -1,
 'ANKSwitchTol': 1000.0,
 'ANKTurbCFLScale': 1.0,
 'ANKTurbKSPDebug': False,
 'ANKUnsteadyLSTol': 1.0,
 'ANKUseApproxSA': False,
 'ANKUseFullVisc': True,
 'ANKUseMatrixFree': True,
 'ANKUseTurbDADI': True,
 'ASMOverlap': 1,
 'CFL': 1.5,
 'CFLCoarse': 0.75,
 'CFLLimit': 1.5,
 'GMRESOrthogonalizationType': 'modified Gram-Schmidt',
 'ILUFill': 2,
 'L2Convergence': 1e-14,
 'L2ConvergenceCoarse': 1e-14,
 'L2ConvergenceRel': 1e-16,
 'MGCycle': 'sg',
'MGStartLevel': -1,
'NKADPC': True,
'NKAMGLevels': 2,
'NKAMGNSmooth': 1,
'NKASMOverlap': 1,
'NKFixedStep': 0.25,
'NKGlobalPreconditioner': 'additive Schwarz',
'NKInnerPreconIts': 1,
'NKJacobianLag': 10,
'NKLS': 'cubic',
'NKLinearSolveTol': 0.3,
'NKOuterPreconIts': 3,
'NKPCILUFill': 1,
'NKSubspaceSize': 100,
'NKSwitchTol': 1e-07,
'NKUseEW': True,
'NKViscPC': False,
'RKReset': False,
'TSStability': False,
'acousticScaleFactor': 1.0,
'adjointAMGLevels': 2,
'adjointAMGNSmooth': 1,
'adjointDivTol': 100000.0,
'adjointL2Convergence': 1e-07,
'adjointL2ConvergenceAbs': 1e-16,
'adjointL2ConvergenceRel': 1e-16,
'adjointMaxIter': 10000,
'adjointMaxL2DeviationFactor': 1.0,
'adjointMonitorStep': 10,
'adjointSolver': 'GMRES',
'adjointSubspaceSize': 100,
'alphaFollowing': True,
'alphaMode': False,
'altitudeMode': False,
'applyAdjointPCSubspaceSize': 20,
'applyPCSubspaceSize': 10,
'approxPC': True,
'backgroundVolScale': 1.0,
'betaMode': False,
'blockSplitting': True,
'cavExponent': 0,
'cavSensorOffset': 0.0,
'cavSensorSharpness': 10.0,
'cavitationNumber': 1.4,
'closedSurfaceFamilies': None,
'coarseDiscretization': 'central plus scalar dissipation',
'computeCavitation': False,
'coupledSolution': False,
'cpMinRho': 100.0,
'cutCallback': None,
'debugZipper': False,
'deltaT': 0.01,
'designSurfaceFamily': None,
'discretization': 'central plus scalar dissipation',
'dissContMagnitude': 1.0,
'dissContMidpoint': 3.0,
'dissContSharpness': 3.0,
'dissipationLumpingParameter': 6.0,
'dissipationScalingExponent': 0.67,
'eddyVisInfRatio': 0.009,
'equationMode': 'steady',
'equationType': 'Euler',
'eulerWallTreatment': 'linear pressure extrapolation',
'explicitSurfaceCallback': None,
'firstRun': True,
'flowType': 'external',
'forcesAsTractions': True,
'frozenTurbulence': False,
'globalPreconditioner': 'additive Schwarz',
'gridFile': 'NLF_N600.cgns',
'gridPrecision': 'double',
'gridPrecisionSurface': 'double',
'infChangeCorrection': True,
'infChangeCorrectionTol': 1e-12,
'infChangeCorrectionType ': 'offset ',
'innerPreconIts ': 1,
'isoVariables ': [],
'isosurface ': {},
'liftIndex ': 3,
'limiter ': 'van Albada',
'loadBalanceIter ': 10,
'loadImbalance ': 0.1,
'localPreconditioner ': 'ILU',
'lowSpeedPreconditioner ': False,
'machMode': False,
'matrixOrdering ': 'RCM',
'maxL2DeviationFactor ': 1.0,
'meshSurfaceFamily ': None,
'monitorVariables ': ['resrho ', 'cd ', 'cl '],
'nCycles ': 200000,
'nCyclesCoarse ': 250,
'nFloodIter ': -1,
'nRKReset ': 5,
'nRefine ': 10,
'nSaveSurface ': 1,
'nSaveVolume ': 1,
'nSubiter ': 1,
'nSubiterTurb ': 10,
'nTimeStepsCoarse ': 48,
'nTimeStepsFine ': 400,
'nearWallDist ': 0.1,
'numberSolutions ': True,
'outerPreconIts ': 3,
'outputDirectory ': './',
'outputSurfaceFamily ': 'allSurfaces ',
'overlapFactor ': 0.9,
'oversetDebugPrint ': False,
'oversetLoadBalance ': True,
'oversetPriority ': {},
'oversetProjTol ': 1e-12,
'oversetUpdateMode ': 'frozen ',
'pMode': False,
'partitionLikeNProc ': -1,
'partitionOnly ': False,
'preconditionerSide ': 'right ',
'printAllOptions ': True,
'printIntro ': True,
'printIterations ': True,
'printNegativeVolumes ': False,
'printTiming ': True,
'printWarnings ': True,
'qMode': False,
'rMode': False,
'resAveraging ': 'never ',
'restartAdjoint ': True,
'restartFile ': None,
'restrictionRelaxation ': 0.8,
'selfZipCutoff ': 120.0,
'sepSensorOffset ': 0.0,
'sepSensorSharpness ': 10.0,
'setMonitor ': True,
'skipAfterFailedAdjoint ': False,
'smoothParameter ': 1.5,
'smoother ': 'DADI',
'solutionPrecision ': 'double ',
'solutionPrecisionSurface ': 'double ',
'storeConvHist ': True,
'storeRindLayer ': True,
'surfaceVariables ': ['cp', 'vx', 'vy', 'vz', 'mach', 'cf'],
'timeAccuracy ': 2,
'timeIntegrationScheme ': 'BDF',
'timeIntervals ': 1,
'timeLimit ': -1.0,
'turbResScale ': 10000.0,
'turbulenceModel ': 'SA',
'turbulenceOrder ': 'first order ',
'turbulenceProduction ': 'strain ',
'useALE': True,
 'useANKSolver': True,
 'useApproxWallDistance': True,
 'useBlockettes': True,
 'useDiagTSPC': True,
 'useDissContinuation': False,
 'useExternalDynamicMesh': False,
 'useGridMotion': False,
 'useLinResMonitor': False,
 'useMatrixFreedrdw': True,
 'useNKSolver': True,
 'useOversetWallScaling': False,
 'useQCR': False,
 'useRotationSA': False,
 'useTSInterpolatedGridVelocity': False,
 'useWallFunctions': False,
 'useZipperMesh': True,
 'useft2SA': True,
 'verifyExtra': True,
 'verifySpatial': True,
 'verifyState': True,
 'vis2': 0.25,
 'vis2Coarse': 0.5,
 'vis4': 0.0156,
 'viscPC': False,
 'viscWallTreatment': 'constant pressure extrapolation',
 'viscousSurfaceVelocities': True,
 'volumeVariables': ['resrho', 'mach', 'cp', 'blank'],
 'wallDistCutoff': 1e+20,
 'windAxis': False,
 'writeSolutionEachIter': False,
 'writeSurfaceSolution': True,
 'writeTecplotSurfaceSolution': False,
 'writeVolumeSolution': True,
 'zipperSurfaceFamily': None}
-> Alpha... 0.000000
#
# Specified load imbalance tolerance 0.100 achieved.
# Continuing with 0.001 load imbalance for the cells and
   0.002 for the faces
#
#
# Grid level: 1, Total number of cells: 69201
#
+------------------------------------------------------+
 CGNS Surface Families by Boundary Condition Type
+------------------------------------------------------+
| Wall Types          : wall
| Inflow Types        :
| Outflow Types       :
| Symmetry Types      : sym
| Farfield Types      : farfield
| Overset Types       :
+------------------------------------------------------+

+------------------------------------------------+
| pyADFLOW Warning:                              |
| 'zipperSurfaceFamily' option was not given.    |
| Using all wall                                 |
| boundary conditions for the zipper mesh.       |
+------------------------------------------------+

+---------------------------------------+
|          All IDWarp Options:          |
+---------------------------------------+
{'LdefFact': 1.0,
 'aExp': 3.0,
 'alpha': 0.25,
 'bExp': 5.0,
 'bucketSize': 8,
 'cornerAngle': 30.0,
 'errTol': 0.0005,
 'evalMode': 'fast',
 'fileType': 'CGNS',
 'gridFile': 'NLF_N600.cgns',
  'restartFile': None,
  'specifiedSurfaces': None,
  'symmTol': 1e-06,
  'symmetryPlanes': None,
  'symmetrySurfaces': None,
  'useRotations': True,
  'zeroCornerRotations': True}
  -> Reading CGNS File: NLF_N600.cgns
    -> Number of Zones:           3
#--------------------------------#
 Total Volume Nodes :    140400
#--------------------------------#
+------------------- Symmetry Planes -------------------+
|            Point                          Normal       |
| (   0.000    1.000    0.000)    (   0.000    1.000    0.000) |
| (   0.000    0.000    0.000)    (   0.000   -1.000    0.000) |
+-------------------------------------------------------+
#----------------------------------#
 Unique Surface Nodes :      1198
#----------------------------------#
 Computing Denominator Estimate...
 Load Balancing...
 Finished Mesh Initialization.
+----------------------------------------+
|  Switching to Aero Problem: FIELD      |
+----------------------------------------+
-> Alpha... 1.767487
+----------------------------------------+
|      All Modified ADFLOW Options:      |
+----------------------------------------+
{'CFL': 1.5,
 'CFLCoarse': 0.75,
 'L2Convergence': 1e-14,
 'L2ConvergenceCoarse': 1e-14,
 'NKADPC': True,
 'NKJacobianLag': 10,
 'NKOuterPreconIts': 3,
 'NKPCILUFill': 1,
 'NKSubspaceSize': 100,
 'NKSwitchTol': 1e-07,
 'adjointL2Convergence': 1e-07,
 'adjointMaxIter': 10000,
 'equationType': 'Euler',
 'gridFile': 'NLF_N600.cgns',
 'gridPrecisionSurface': 'double',
 'liftIndex': 3,
 'monitorVariables': ['resrho', 'cd', 'cl'],
 'nCycles': 200000,
 'nCyclesCoarse': 250,
 'nSubiterTurb': 10,
 'resAveraging': 'never',
 'skipAfterFailedAdjoint': False,
 'solutionPrecision': 'double',
 'solutionPrecisionSurface': 'double',
 'surfaceVariables': ['cp', 'vx', 'vy', 'vz', 'mach', 'cf'],
 'turbResScale': 10000.0,
 'useNKSolver': True,
 'volumeVariables': ['resrho', 'mach', 'cp', 'blank']}
#
# Grid 1: Performing 200000 iterations, unless converged
    earlier.
# Minimum required iteration before NK switch:
# 5. Switch to NK at totalR of:   2.94E-03
#
```

Code C.2 Settings used in the NOLOT code for performing the stability analysis (direct and adjoint) presented in this work. These settings are specified in the code named psecomp_init.in."

```
meanflow.dat
20 100            N, ymax
3                 ifdim : (0:nodim., 1:dim. input, 2 & 3 automatic choice of
    wavenumbers)
1                 mutask(1:sutherland, 2:polynomial)
1                 kptask(1:keye, 2:polynomial)
1                 prtask(1:constant, 2:variable prandtl number)
1                 cptask(1:constant, 2:variable cp)
0 500 20          k0, delta_k and number of k (k is total wavenumber)
0 90 45             first and last waveangle and number of steps
25 2e-4            number of frequency and cut of reduced frequency (for ifdim>1)
2 10              istart, istep
2500000           Step lenght for frequancy to save (not used in this version)
1                 metask (0=without curvature, 1=with curvacure)
1                 Stask (1=parallel, 2=nonparallel)
```

Code C.3 Settings used in the NOLOT code for performing the stability analysis (direct and adjoint) presented in this work. These settings are specified in the code named psecomp.in."

```


1                                    1: Dimensional frequency
1                                    2: Dimensional wavenumbers


2                                    3: Malik's point distribution
  100.00000                             Coordinate of last point
  70.000000                             Percentage of points in region [0:yp]
  20.000000                             Coordinate of yp


 500                                 4: Number of stations in streamwise dir.
   1                                    Every i'th station used only
4                                   5: Nonlocal, nonparallel stability analysis
2                                       I.c. from local, nonparallel theory
1                                       First order backward Euler


1                                   6: Constant Prandtl number
1                                   7: Constant specific heat
1                                   8: Keyes' formula for kappa
1                                   9: One part Sutherland's formula for viscosity


0                                   10: Amplitude functions not saved
1                                   11: Given metric
  10                                12: Maximum number of iterations
 0.10000000E-06                  13: Convergence criterion
meanflow.dat                     14: Meanflow file (max 30 char):
```