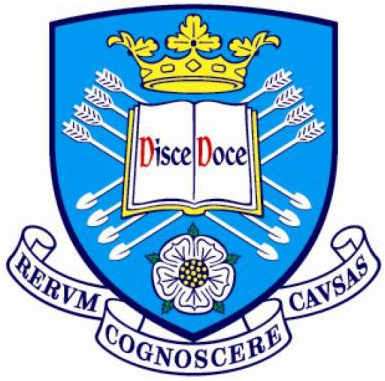


The University of Sheffield

Department of Mechanical Engineering

Mesh Sensitivity Investigation in the Discrete Adjoint Framework

Gabriele Luigi Mura

Supervised by Prof. Ning Qin

This thesis is submitted to University of Sheffield in partial fulfilment of the requirement for the degree of Doctor of Philosophy

March 2017

Abstract

Aerodynamic optimisation using gradient-based methods has found a wide range of academic applications in the last 30 years. This framework is also becoming more and more popular in the industrial world where, most of the time, unstructured grids are largely used. In this framework, apart from the need to solve the flow field, there is the need to quickly map the aerodynamic surface in terms of some aerodynamic figure of merits such as the drag coefficient, without being limited by the computational expense related to the grid size. This is a concrete industrial need which requires the efficient computation of the grid sensitivity.

A novel method based on the DGM (Delaunay Graph Mapping) mesh movement is proposed to efficiently compute the grid sensitivity required in the discrete adjoint optimisation framework. The method makes use of a one-to-one explicit algebraic mapping between the volume mesh and the solid boundary nodes. This procedure results in a straightforward computation of the gradient without the need to invert a large, sparse and stiff matrix generally associated with implicit mesh movements such as the spring or LE (Linear Elastic) analogy. The method is verified using FDs (Finite Difference) and a thorough comparison in terms of CPU time, formulation against the LE-based mesh movement and adjoint gradient is presented.

The DGM-based gradient chain allows to comfortably obtain the gradient with respect to each surface mesh point. Unfortunately, these gradients cannot be used directly because of their inherent poor smoothness feature. In order to address this issue one has to use a parameterisation technique which inevitably sacrifices the design space explorablity. To bridge the gap between the free-nodes and the parameterisation approaches, a novel formulation of the CST (Class Shape Transformation) was developed and termed *l*-CST (*local*-CST). The method is based on a simple trigonometric function which works as a cut-off filter on the BPs (Bernstein Polynomials) which are used to enforce a strong on-demand local control. The method is tested on an inverse geometric fitting and its effect on the resulting aerodynamic coefficients and the pressure distribution is also analysed.

The DGM-based chain allows the efficient mapping of the entire surface while the *l*-CST allows the combination of excellent explorablity and surface smoothness. The former is tested within the non-consistent mesh movement and sensitivity framework because there are situations

where one method may be preferred over the other based on the grounds that mesh movement is a very different task than mesh sensitivity although strongly related to each other. The latter is instead tested against the free-nodes approach which offers a similar advantage in terms of discrete control without maintaining a C^2 curve unless properly smoothed.

Publications

Journal Articles

1. **Mura**, **G. L.**, Hinchliffe, B. L., Qin, N., and Brezillon, J., “Efficient Method to Eliminate Mesh Sensitivity in Adjoint Based Optimization,” *AIAA Journal*, Vol. 55, No. 4, 2017, pp. 1140-1151..

Conference Papers

1. **Mura**, **G. L.**, Hinchliffe, B. L., Qin, N., and Brezillon, J., "Using a Fast and Explicit Mesh Movement Method to Efficiently Compute Mesh Sensitivity," *54th AIAA Aerospace Sciences Meeting and Exhibit*, San Diego, USA, AIAA 2016-0291, January 2016.
2. **Mura**, **G. L.**, Hinchliffe, B. L., and Qin, N., “Robust Delaunay Graph Based Mesh Movement for Adjoint Mesh Sensitivity,” *RAeS Applied Aerodynamics Conference*, Bristol, UK, July 2016.
3. **Mura**, **G. L.**, and Qin, N., “Local Class Shape Transformation Parameterization (*l*-CST) for Airfoils,” *55th AIAA Aerospace Sciences Meeting and Exhibit*, Grapevine, USA, AIAA 2017-0237, January 2017.
4. **Mura**, **G. L.**, Hinchliffe, B. L., Qin, N., and Brezillon, J., “Effect on Non-Consistent Mesh Movements and Sensitivities on a Discrete Adjoint Based Aerodynamic Optimization,” *55th AIAA Aerospace Sciences Meeting and Exhibit*, Grapevine, USA, AIAA 2017-0461, January 2017.

Chie dormit a pizzinnu pianghet a bezzu

(Loosely translated as: Who sleeps when he is young cries when he is old)

Old Sardinian dialect saying.

Acknowledgements

My first acknowledgment goes to Prof. Ning Qin for giving me the opportunity to start this Ph.D. and for his guidance, patience and attention to the details throughout these three years.

I acknowledge Bruno Cerrato from SATIZ and Antonio Morena from FIAT Chrysler Automobile for giving me the opportunity to leave my much loved job and start this Ph.D.

From Airbus U.K., Dr. Kasidit Leoviriyakit for serving as my initial industrial supervisor during my time in Filton. His initial support and technical knowledge was deeply appreciated. From Airbus France, Joël Brezillon for taking up the position as my second industrial supervisor. His knowledge of the adjoint method together with his attention to the details has most definitely shaped this work. Murray Cross for setting up the collaboration with The University of Sheffield and for the help received during my placement in Filton. The University of Sheffield and Airbus U.K. financial support is deeply appreciated.

A special thanks is directed to all my colleagues from The University of Sheffield Aerodynamic Research Group. In particular, Dr. Benjamin Lee Hinchliffe for the initial implementation and the verification of the DGM code, Dr. Feng Zhu for his help and knowledge of the CFD tools and optimisation methods during my time in Filton and Dr. Sheng Liu for the fruitful discussion on the turbulence models applied to laminar aerodynamic flows. A special thanks is also directed to my colleagues in Filton: Jean Demange and Spyridon Kontogiannis (University of Cranfield), Simone Simeone, Dr. Christian Agostinelli (University of Bristol) and Hamza Ali (University of Surrey).

From the German Aerospace Centre (DLR), Caslav Ilic and Andrei Merle for their advice and help during the implementation of the DGM method in the DLR TAU-code.

Finally, family and friends, without them I would not be here. A special thanks is also due to Pasquale Zito and Marco Corongiu for introducing me, from an early age, to the engineering world.

Gabriele Luigi Mura

The University of Sheffield, Mechanical Engineering Department,

November 1st, 2016

Table of Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$\{\boldsymbol{A}\}$	CST coefficients vector
$[\boldsymbol{A}]$	Kinematic matrix
B	Approximation of the Hessian
$\{\boldsymbol{B}\}$	Vector of all the Delaunay vertices
$\{\boldsymbol{B}_{ff}\}$	Delaunay vertices that coincide with the farfield surface mesh nodes
$\{\boldsymbol{B}_{sb}\}$	Delaunay vertices that coincide with the supporting box nodes
$\{\boldsymbol{B}_{sw}\}$	Delaunay vertices that coincide with the solid wall surface mesh nodes
c	Chord
c_p	Specific heat at constant pressure
C_d,C_l, C_{my}	Drag, lift and longitudinal moment coefficient for a 2D geometry
C_D,C_L, C_{MY}	Drag, lift and longitudinal moment coefficient for a 3D geometry
C_f	Skin friction coefficient
C_p	Pressure coefficient
$C^{1,2}$	First and second order continuity, i.e. slope and curvature continuity
d	Total derivative or distance from the wall
dV	Discretised volume boundary
dS	Discretised surface boundary
D	Drag
$\{\boldsymbol{D}\}$	Design variables vector
D/Dt	Material derivative
e_i	Volume ratio coefficient

E	Energy or Young's modulus
$[\boldsymbol{E}]$	Delaunay volume coefficient ratios matrix
f_i	Hicks-Henne bump function
$\{\boldsymbol{F}^i, \boldsymbol{G}^i, \boldsymbol{H}^i\}$	Inviscid flux vector along x, y, z respectively
$\{\boldsymbol{F}^v, \boldsymbol{G}^v, \boldsymbol{H}^v\}$	Viscous flux vector along x, y, z respectively
H	Total enthalpy or inverse of the approximated Hessian
k	Turbulent kinetic energy or iteration index
$\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$	Unit vectors along x, y, z respectively
I	Objective function
$[\boldsymbol{I}]$	Identity matrix
$[\boldsymbol{K}]$	Linear elasticity mesh deformation matrix
l	*l*-CST local function exponent
L	Lift
log_{10}	Logarithm with base 10
$\mathcal{L}$	Lagrangian (augmented objective function)
L_2	Norm of the least-squares error
M	Mach number
n	Bernstein polynomial's degree
$\boldsymbol{n}$	Boundary normal pointing vector
N_{CP}	Number of parametric control point
N_{DV}	Number of design variables
N_I	Number of objective functions
N_f	Number of dual-cell control volume faces
N_{VM}	Number of volume mesh nodes

N_{SB}	Number of supporting box points
N_{SM}	Number of surface mesh nodes
$N_{1,2}$	CST class shape parameters
p	Pressure or descendent direction
P	Quadratic approximation of the objective function
$\{\boldsymbol{P}\}$	Vector containing the Cartesian coordinates
$\{\boldsymbol{P}_l\}$	Vector containing the Cartesian coordinates with respect to the local system
Pr	Prandtl's number
q_x, q_y, q_z	Heat fluxes along x, y, z respectively
$\bar{q}_x, \bar{q}_y, \bar{q}_z$	Turbulent heat fluxes along x, y, z respectively
$\{\boldsymbol{Q}\}$	Flux tensor
R	Gas constant
$\{\boldsymbol{R}\}$	Discretised flow residual vector
$R_{adjoint}$	Residual associated to the flow-adjoint system
R_{flow}	Residual associated to the flow system
s	Second
$sign\ ()$	Sign of a real number
S	Sutherland's temperature constant
$\{\boldsymbol{S}\}$	Surface (solid wall) mesh nodes vector
$[\boldsymbol{S}]$	Sobolev smoothing matrix
S_P	Production source term
S_D	Destruction source term
t	Time

$t_{1_i,2_i}$	Hicks-Henne bump control coefficients
T	Temperature
$\{\boldsymbol{T}\}$	Mesh movement residual vector
T_0	Temperature at ambient condition
Tr	Trace
u	Non-dimensionalised chord length
u_{CP}	Non-dimensionalised position of the local control peak
u_{TE}	Trailing edge thickness
u, v, w	Cartesian component velocities along x, y, z respectively or FFD local parametric coordinates
v_i	Delaunay sub-volume
V	Dual cell volume or Delaunay volume element
x, y, z	Cartesian direction or x_i
y^+	Dimensionless wall distance
$\{\boldsymbol{X}\}$	Volume mesh nodes vector
$\{\boldsymbol{W}\}$	State variables vector
$\cdot$	Inner product
$\times$	Outer product

Greek Symbols

α	Angle of attack or step size in the search direction
δ_{ij}	Kronecker's delta
δ_i	Local function coefficient
ε	FD step size or smoothing factor when used in the Sobolev gradient
θ_i	Dihedral angle
κ	SA turbulence model constant
λ	Second coefficient of viscosity

$\{\boldsymbol{\Lambda}_F\}$	Flow-adjoint vector
$\{\boldsymbol{\Lambda}_G\}$	Mesh adjoint vector
$\{\boldsymbol{\Lambda}_G\}_{DGM}$	Mesh adjoint vector w.r.t. the Delaunay graph method
$\{\boldsymbol{\Lambda}_G\}_{LE}$	Mesh adjoint vector w.r.t. the linear elasticity
μ	Dynamic viscosity
μ_0	Dynamic viscosity at ambient condition
μ_{eff}	Effective dynamic viscosity
μ_l	Laminar dynamic viscosity
μ_t	Eddy or turbulent dynamic viscosity
υ	Poisson's ratio
$\tilde{\nu}$	Modified kinematic viscosity
ρ	Fluid density
σ	Spalart-Allmaras turbulence model's coefficient
π	pi
$\boldsymbol{\sigma}$	Linear elasticity stress tensor
τ_{ij}	Reynolds viscous stresses
$\bar{\tau}_{ij}$	Turbulent Reynolds viscous stresses
Γ	Computational domain boundaries
Φ_{ij}	Angle formed by two tetrahedron edges
Ω	Computational volume
φ	General time and space dependent quantity
∂	Partial derivative
∇	First derivative
∇^2	Second derivative

Subscripts

D	Drag
DGM	Delaunay Graph Method
F	Flow
FF	Far Field
G	Grid

i	Inviscid
IP	Interpolating Points
L	Lift
LE	Linear Elasticity
$n.d.r.$	normal decay rate
p	pressure
SB	Supporting Box
SM	Surface Mesh
SW	Solid Wall
v	viscous
VM	Volume Mesh
∞	free-stream condition

Superscripts

T	Transpose operator
-1	Inverse operator
$'$	Fluctuating term (Reynolds)
$''$	Fluctuating term (Favre)
$\bar{}$	Reynolds averaged term
$\tilde{}$	Favre averaged term

Acronyms

AoA	Angle of Attack
AD	Automatic Differentiation
ADIFOR	Automatic DIfferentiation for FOrtran
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BP	Bernstein Polynomial
CAD	Computer Aided Drawing
CAPRI	Computational Analysis PRogramming Interface
CFD	Computational Fluid Dynamics
CF	Class shape Function

CP	Control Point
CPU	Central Processing Unit
CST	Class Shape Transformation
DC	Drag Count
DPW	Drag Prediction Workshop
DGM	Delaunay Graph Mapping
DLR	Deutsches Zentrum für Luftund Raumfahrt (German Aerospace Centre)
DNS	Direct Numerical Simulation
DV	Design Variable
FANS	Favre Averaged Navier-Stokes
FD	Finite Difference
FDM	Finite Difference Method
FEM	Finite Element Method
FF	Far Field
FFD	Free-Form Deformation
FVM	Finite Volume Method
GD	Gradient Destruction
GMRES	Generalised Minimal RESidual
H-H	Hicks-Henne
IDW	Inverse Distance Weighting
ILU	Incomplete Lower Upper
I/O	Input/Output
JST	Jameson-Schmidt-Turkel
l-CST	*local*-Class Shape Transformation
L-BFGS	Limited (memory) Broyden-Fletcher-Goldfarb-Shanno
LE	Linear Elasticity
LES	Large Eddy Simulation
LHS	Left Hand Side
LU-SGS	Lower-Upper Symmetric-Gauss-Seidel
mDGM	modified Delaunay Graph Mapping
MDO	Multidisciplinary Design Optimisation
MINRES	MINimal RESidual

NACA	National Advisory Committee for Aeronautics
NFA	Near Field Approach
NLF	Natural Laminar Flow
NURBS	Non-Uniform Rational Basis Spline
oDGM	original Delaunay Graph Mapping
ONERA	Office National d'Etudes et Recherches Aérospatiales
PARSEC	PARAmetric SECtion
PDE	Partial Differential Equation
RAE	Royal Aeronautical Establishment
RAM	Random Access Memory
RANS	Reynolds Averaged Navier-Stokes
RBF	Radial Basis Function
RHS	Right Hand Side
RSM	Reynolds Stress Model
SA	Spalart Allmaras
SB	Supporting Box
SM	Surface Mesh
SL-SQP	Sequential Least-Squares Quadratic Programming
SST	Shear Stress Transport
SW	Solid Wall
TFI	Trans Finite Interpolation
TSL	Thin-Shear Layer
VOLGRAD	VOLume (approach for) GRADient (calculation)
VM	Volume Mesh

Chapter 1 Introduction

1.1 Background

Civil commercial aircraft have been flying for more than a century. Their shape evolved significantly since the first commercial aviation flight in 1914, and has since then inevitably settled on a design generally referred to as tube-and-wing configuration. More or less at the same time, aerodynamic numerical methods started to become available [1]. These activities can be roughly split into three areas, namely mesh generation, numerical schemes for CFD (Computational Fluid Dynamics) solutions and optimisation methods. In 1978, after the seminal paper by Hicks and Henne [2] another area of research started to emerge which focused on combining these three branches together. Since then, this subject has been known as aerodynamic optimisation.

There are many types of aircraft made for different flow regimes. This work focuses primarily on the transonic regime where commercial civil aircraft typically operate. It is interesting to see how the aircraft history compares to the development history of CFD. Of particular interest is the ratio between wind tunnel and CFD tests where the latter is becoming more and more predominant. As can be seen from Fig. 1.1, each period of revolutionary design is then followed by a period of evolutionary design and vice versa. These terms both describe technology advances, but are not quite the same. In a few words, where the former concentrates on radical changes, such as the introduction of sweeping and turbofan, the latter focuses on making them perfect via minor improvements. In this two types of transitions CFD is becoming more and more important.

In a broad sense, CFD and optimisation techniques together seek to facilitate the transition from revolutionary to evolutionary and vice versa. CFD is defined as the art of discretising the Navier-Stokes equations, converting them into a system of algebraic equations and solving them in order to obtain an approximated solution of real world phenomena. Therefore, it seeks to make a bridge between pure theory and wind tunnel experiments. On the other hand, optimisation is the art of finding through numerical methods, the best optimum. Thus, it seeks to make a bridge between the designer’s intuition and the classical trade studies. However, as one tries to combine these two, new challenges arise as Drela [3] has underlined.

There are many optimisation branches and techniques currently being investigated. Of particular interest is the distinction between those that focus on finding the best solution and those that focus on finding the most robust solution. The latter inherently describes the real challenge of a truly useful aerodynamic optimisation. In fact, finding the best solution is already difficult from a practical and also numerical point of view, thus making the entire process robust creates a whole set of new problems. This thesis finds its place within the first type of classification because even before asking how the best and most robust optimum can be found, one has to have the capability to see what the design space has to offer without being limited by the size of the mesh. In other words, the designer has to see how the objective function (e.g. the drag coefficient) changes in response to a change made anywhere in the aerodynamic surface. One way to do this is to compute the gradient, however, it is recognised that other strategies such as gradient-free methods could be used. These two strategies are by definition different and their pros and cons need to be evaluated on a case-by-case basis. Nevertheless, some general comments can still be made. Gradient-based methods are generally preferred when a fast convergence of the chosen merit function is needed. These methods suffer from getting stuck in the closest local minimum and the gradient calculation is not always an easy step to perform. Thus, in situations where time is a restriction, fast convergence is an advantage. On the other hand, when the time is not a constraint and a wider design space exploration is necessary, gradient-free methods are instead preferred.

There are many types of sensitivity available such as flow and structure sensitivity. Specifically, this work focuses on the sensitivity of integrated quantities such as the drag and lift coefficients. These should not be considered as a standalone part of the gradient-based chain, but as a part of a highly integrated system, something that was recognised since the 1980's where the so-called "*Sobieski's plea*" [4] was made for methods able to provide a robust and interdisciplinary sensitivities.

Sensitivity derivatives are one of the most important part of any gradient-based algorithm. There are many techniques to obtain them, ranging from not accurate to analytically accurate methods. This research area has seen many developments in the last 30 years. Among them, the adjoint method [5, 6] has been widely recognised as one of the most promising and reliable. Nevertheless, limitations were uncovered and some of them are addressed in the current work.

The adjoint method allows one to compute the gradient at a cost nearly independent from the N_{DV} (number of DVs). This has naturally lead researchers to explore the pros and cons of using the maximum N_{DV} possible. This case is represented by the entire surface of a lifting body where $N_{DV} = N_{SM}$ (number of surface mesh) which leads to a greatly improved exploration capability. Of course, as will be made clear in this work, maximum explorability does not equate maximum usability. Many techniques were developed to bridge this gap and they generally go under the umbrella name of parameterisation. Although these are very effective, they consistently lack in either intuitiveness or explorability.

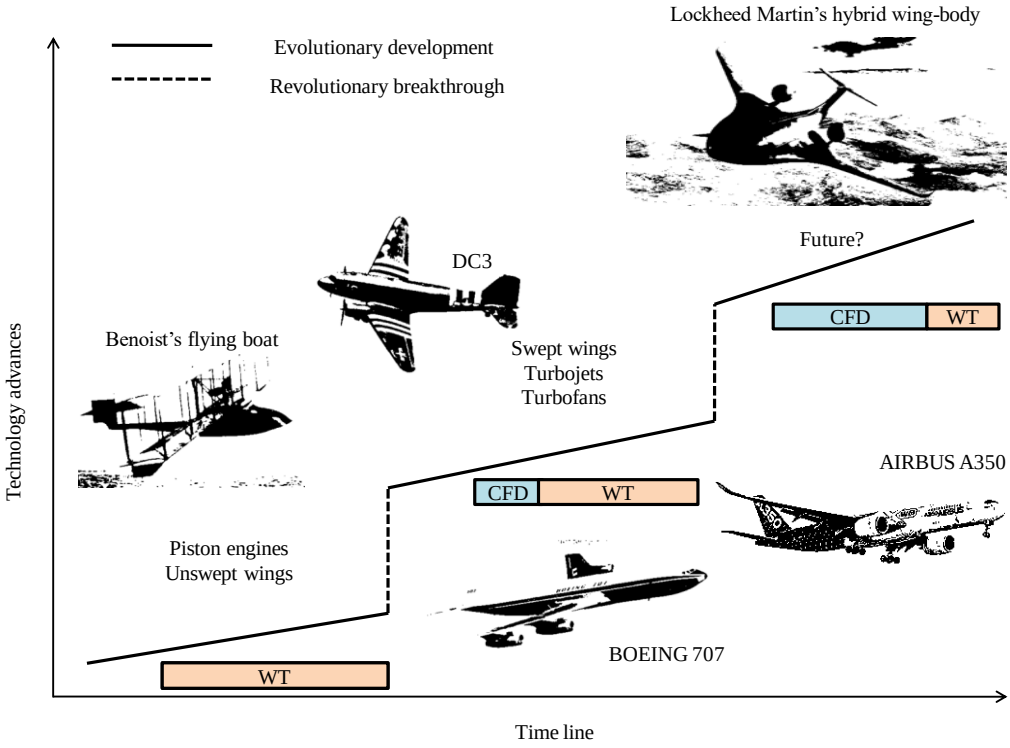


Figure 1.1 Revolutionary and evolutionary milestones for civil commercial aircraft.

This is important because, as Fig. 1.1 shows, the current time period is characterised by a constant evolution where aircraft manufactures are trying to improve their products by exploiting even the smallest detail. To this end, it is felt that having access to the entire design space, i.e. explorablity, and more importantly how it varies, i.e. gradient, without incurring in

memory and computation time issues while being able to impose small tweaks, i.e. usability, is of fundamental importance.

Having said that, the adjoint method is the most appropriate approach currently available. In this highly complex and integrated framework, Reuther *et al.* [7] highlighted five *salient issues of concern*:

- Discrete versus continuous adjoint-based gradient
- Choice of the optimisation procedure
- Treatment (and influence) of the constraints
- Level of coupling between CFD software and optimisation routine
- Parameterisation of the shape

This work investigates a method, based on the DGM mesh movement, to efficiently compute the grid sensitivity within the discrete adjoint approach applied to a lift-constrained optimisation problem coupled with a quasi-Newton search direction optimiser.

1.2 Aims and Objectives

The aims and objectives of this thesis were established based on the surveyed literature. For each research question there is one main focus accompanied by different specific research questions defined as the investigation was advancing. Gradient-based optimisation comprises of four main steps: mesh movement, gradient calculation, parameterisation and optimisation framework. The research questions addressed in this thesis can be assigned to each one of these four steps.

The first is to find a low memory and low CPU (Central Processing Unit) time consuming mesh movement method. The DGM mesh movement [8] is chosen as the mean investigation tool and associated with this choice there are a series of specific research questions:

- Assessment of the cost in terms of memory and CPU time
- Assessment of the effect of the supporting box necessary to make the DGM more robust
- Comparison against the LE mesh movement

The second step addresses the grid sensitivity, an issue peculiar to the discrete adjoint framework. The main aim is the development and verification of a newly proposed and efficient method which is able to comfortably handle the grid sensitivity computation. To this end, the following actions are investigated:

- Differentiation of the DGM mesh movement
- Verification of the DGM-based adjoint chain
- Assessment of the DGM cost both in memory and CPU time
- Comparison against the LE-based gradient
- Assessment of the grid sensitivity differentiation extent
- Assessment of the effect of the supporting box (i.e. a series of carefully selected volume mesh points) on the gradient.

Even if a large design space becomes accessible (thanks to the DGM) this does not automatically mean it can be effectively used. In fact, the gradient and shape smoothness are two very important drawbacks when each surface point is used as a DV (Design Variable). Smoothing via Sobolev gradient [9, 10] is one of the technique used to solve this problem as an alternative to parameterisation. With this in mind, the novel *l*-CST parameterisation method [11] was developed. The following research questions are investigated:

- Effect of the *l*-CST correction on each BP
- Sensitivity analysis
- Comparison against the CST method
- Assessment of the method capabilities

Finally, the last step is to test the DGM-based chain in a gradient-based optimisation loop. With this in mind, the following research questions were addressed:

- Assessment of the influence of the supporting box in an optimisation loop
- Comparison between different consistent (i.e. same mesh movement and grid sensitivity) approaches
- Assessment of non-consistent (i.e. different) mesh movements
- Assessment of non-consistent differentiated mesh movements

- Assessment of the *l*-CST parameterisation against the free-nodes approach
- Assessment of the Sobolev smoothing operator applied either on the shape or on the gradient

The connections between the above bullet points can be seen in a concise way in Fig. 1.2.

1.3 Thesis Outline

This thesis comprises of eight chapters which are briefly described here.

Chapter 2 presents the literature review focused on the mesh sensitivity computation in the discrete adjoint framework. This offers an overview of all the methods known in the literature to address the linearisation of the mesh movement. For each one of them, the pros and cons are analysed. In here, the main research question is addressed in the details and the chosen investigation path is outlined.

Chapter 3 gives an introduction on the governing equations and the solution methods used. The Navier-Stokes equations are introduced along with their averaging formulation, turbulence model, discretisation and numerical scheme used. The choices made are justified in relation to the flow regime investigated in the present work.

Chapter 4 briefly addresses the volume mesh deformation techniques available. The main investigation tool, i.e. DGM-based mesh movement, is introduced along with a modification which improves its robustness. This is then compared against the LE mesh movement in order to assess its advantages and disadvantages.

Chapter 5 is the core of this thesis. This chapter addresses the differentiation of the DGM and its use in the discrete adjoint sensitivity chain. Its derivation and verification against FDs is discussed and a comparison against the LE in terms of formulation and performance is also presented. Furthermore, a theoretical discussion is proposed which addresses, within the discrete adjoint approach, the evaluation of the grid sensitivity and its distribution in the computational domain.

Chapter 6 addresses the surface mesh deformation techniques used in the current work. Two techniques are discussed, namely the newly formulated *l*-CST and the FFD (Free-Form Deformation) [12]. The former introduces a modification of the original CST [11] which allows to add a strong on-demand local control without modifying the degree of the parameterised curve. On the other hand, the latter is used for the study of the non-consistent approaches as it can be easily applied to 3D geometries. In this framework, the reasons behind the lack of gradient smoothness are discussed and the solution offered by the Sobolev operator is also presented.

Chapter 7 is where the general performances of the DGM-based sensitivity chain and *l*-CST are assessed by considering two types of optimisation studies. At first, the influence of the supporting box is studied in conjunction with the volumetric grid sensitivity extent differentiation. Subsequently, the questions of non-consistent mesh movements and grid sensitivities are addressed. To conclude, a series of optimisations in 2D, where the *l*-CST is tested against the free-nodes approach, are presented. In this case, the effect of smoothing directly either the gradient or the shape is investigated and the results compared against the *l*-CST.

Chapter 8 describes the conclusions and outlines the future works by providing a detailed research path for the future studies.

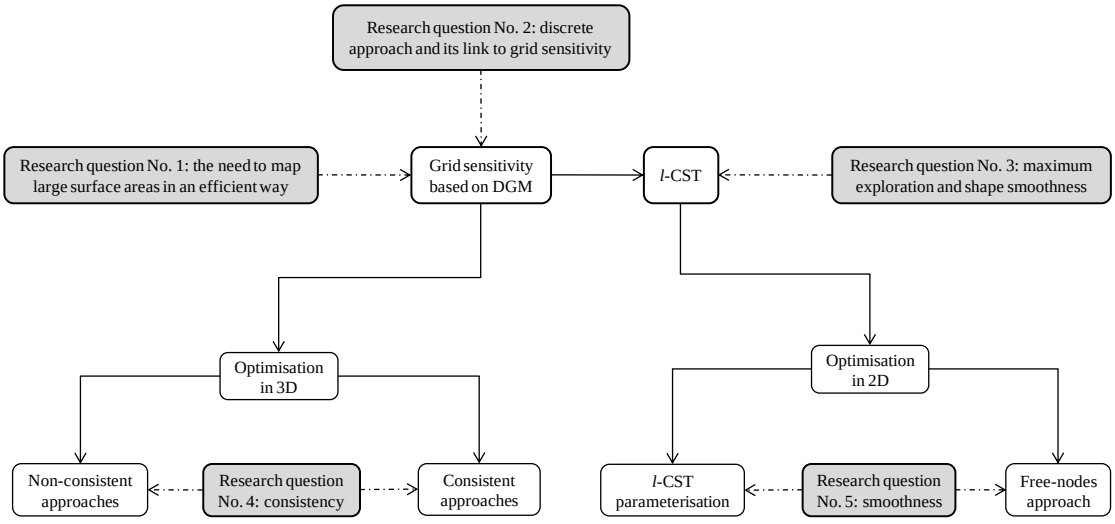


Figure 1.2 Schematic view of the addressed research questions.

Chapter 2 Literature Review

2.1 Introduction

This literature review starts with a broad introduction of the gradient-based optimisation framework where the flow and grid sensitivity are introduced.

The analysis first focuses on the flow-adjoint gradient computation where a discussion on the differences between the continuous and the discrete approach is presented. The reasons why the discrete approach has been preferred over the continuous approach are analysed. From here, a list of the issues related to the discrete adjoint framework along with some possible solutions is presented. This naturally leads to the discussion of the core of this work, namely grid sensitivity computation and its bottleneck.

The second part focuses on the methods that lead to the current mesh adjoint method so as to present the research path followed in the literature, i.e. from FDs to the adjoint method. Each method is analysed along with its pros and cons. Here, what needs to be done is outlined and the reasons why this thesis's goal is worth further study are discussed.

The chapter concludes with a summary of the most important findings.

2.2 Gradient-based Optimisation Framework

Before CFD and optimisation routines became available, aerodynamic design improvements were the result of painstaking cut-and-try solutions. These were based on the evaluation of the impact of a change in a single DV on the figure of merit while maintaining the other DVs constant. This routine is generally referred to as trade study. It is easy to see that the expense of performing this analysis scales linearly with the N_{DV}. It was just a matter of time to see the automation, i.e. aerodynamic optimisation, of this manual process.

In the aerodynamic optimisation literature, one of the first classification that can be made is between *direct* and *inverse* methods as shown in Fig. 2.1. Direct design methods are used to solve the flow field which is then coupled to an optimiser which provides the DV updates. The final product of this process is an optimised pressure distribution. On the other hand, the inverse

design method encompasses all those strategies for which the aerodynamic shape is the output when an already optimised pressure is given as an input.

For both design strategies there are two important steps. The first one is the flow solution for which many strategies have been published spanning from multigrid, pre-conditioning to variations of the Newton-type algorithm. The second one is the computation of the direction of improvement which can be obtained using either a gradient-free [13, 14], gradient-based algorithm or an hybridisation of the previous two [15, 16].

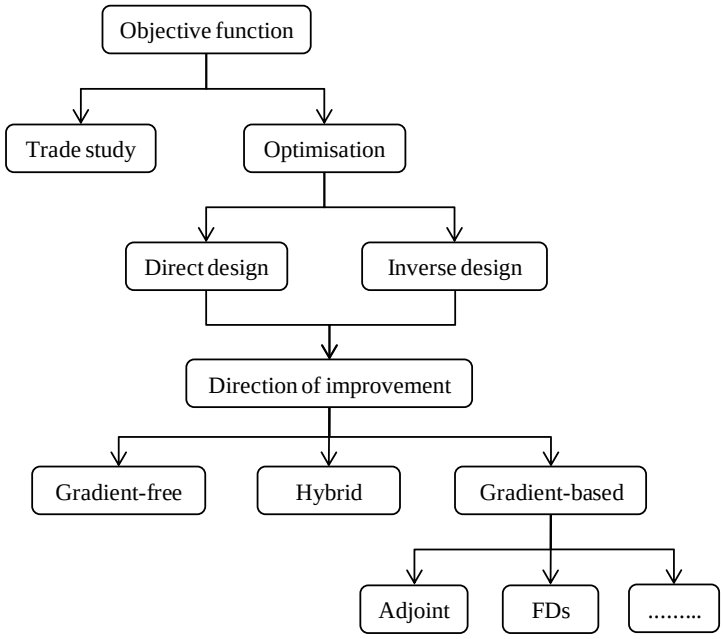


Figure 2.1 Initial classification of the available optimisation frameworks.

The focus of this work is on the direct gradient-based optimisation framework. To do so, it is necessary to know how the system varies in response to a change in the DV. There are different terms in the literature to refer to this process as noted by Martins and Hwang [17]. For instance, terms like *sensitivity analysis*, *sensitivity derivatives* and *design sensitivity* are all common. However, in this work the term *gradient* is preferred as it describes the true nature of what is being computed, i.e. a vector of partial derivatives.

To justify the choice made in this thesis, several metrics, as listed in Tab. 2.1, need to be considered. It is constructive to build the comparison by making a parallel with the gradient-free algorithms. Exploration denotes the capability of the optimiser to avoid being trapped by the nearest optimum and gradient-free methods do it best, whereas exploitation denotes how quickly the nearest optimal point is found and gradient-based methods are known to successfully accomplish this task in a reasonable amount of time. The DV type refers to whether is discrete such engines' number, or continuous such as the increment of the shape parametric DVs. It is known that gradient-based algorithms are best suited for the latter, whereas gradient-free methods perform best for the former. Discontinuity refers, as the name suggests, to the number of continuous derivatives and gradient-based algorithms, contrary to the gradient-free framework, require to maintain derivatives continuity. In terms of convergence criteria, there is a solid theory behind the criteria used to terminate a gradient-based algorithm, whereas non-gradient based may show a series of iterations with no improvement at all, making difficult to establish when the process needs to be terminated. The cost is considered in relation to how quickly it scales with the N_{DV} and gradient-free methods seem to suffer more when compared to gradient-based methods. The development time is also important from an implementation stand point of view especially for industrial applications because, for instance, the development of the adjoint solver requires a considerable effort. Lastly, the problem of the choice of the initial condition is much more important for gradient-based than for gradient-free algorithms. This is because the former, if placed closed to a local minimum, will definitely converged to this local minimum, whereas for the latter this is likely not to happen thanks to their improved explorability features. For a further discussion of the respective merits of the two optimiser approaches the reader is referred to Pulliam *et al.* [18] and Zingg *et al.* [19].

Considering this thesis's goal which is to map a large continuously varying DVs, the choice of a gradient-based optimiser seems to be the most appropriate [20]. Firstly, although it is true that gradient-based algorithms show poor exploration capability, meaning they are susceptible of getting stuck in a local minimum, it is also true that gradient-free algorithms have a cost proportional to the N_{DV} with poor exploitation capability. Therefore, it is felt that the time required to provide a direction of improvement is a much bigger limitation than the risk of getting stuck in a local minimum. This means that, by employing a gradient-based algorithm, although it is not guaranteed that the global minimum will be found, at least an improvement in a reasonable amount of time can be achieved.

Table 2.1 Comparison between gradient-based and gradient-free approaches.

Metrics	Gradient-free	Gradient-based
Exploration	Very good (global)	Poor (local)
Exploitation	Slow	Fast
DV types (variation)	Discrete (No. of engines etc.)	Continuous (parametric DVs)
Discontinuity	Easily treated	Problematic
Convergence criteria	Difficult to choose	Solid theory behind
Computational cost	Expensive in terms of function evaluations	Expensive (FDs) Inexpensive (adjoint)
Development time	Not time consuming	Not time consuming (FDs) Time consuming (adjoint)
Initial conditions	No much influence	Very important

2.3 Flow Sensitivity

Derivatives analysis is a branch of aerodynamic optimisation that deals with the calculus of variations. In the specific, there are two basic and expensive steps, namely derivative of the state variable w.r.t. the DVs and derivative of the grid w.r.t. the geometric DVs. In this chapter the focus is on the first type of derivative for which a classification of the method employed to obtain them is depicted in Fig. 2.2, whereas the second type is described at length in Chap. 2.4.

In 1978, Hicks and Henne [2], published in the *Journal of Aircraft* a paper titled *"Wing Design by Numerical Optimization*". This is considered the first paper to describe a fully automated design optimisation loop. This article along with many others that followed based the search direction on derivatives computed using FDs. This technique has three main drawbacks: they are inaccurate, problem dependent and require an additional flow-call for each DV which makes the cost of the method linearly scalable with the N_{DV}. Furthermore, they need a parametric study to choose the appropriate step size.

After Hicks and Henne [2] seminal paper, many other techniques were published. Methods such as complex-step, direct differentiation method, automatic (algorithm) differentiation and adjoint

(also known as variational or optimal control) all try to address a specific issue by their own. For each one of them the most important features are briefly summarised. However, most of the techniques listed here are then re-discussed for the computation of the grid sensitivity. Therefore, in order to avoid repetitions, the discussion is restricted to the direct and the adjoint method only. This comparison is appropriate because the direct approach provides the same sensitivity from an accuracy stand point of view but is delivered at a different cost.

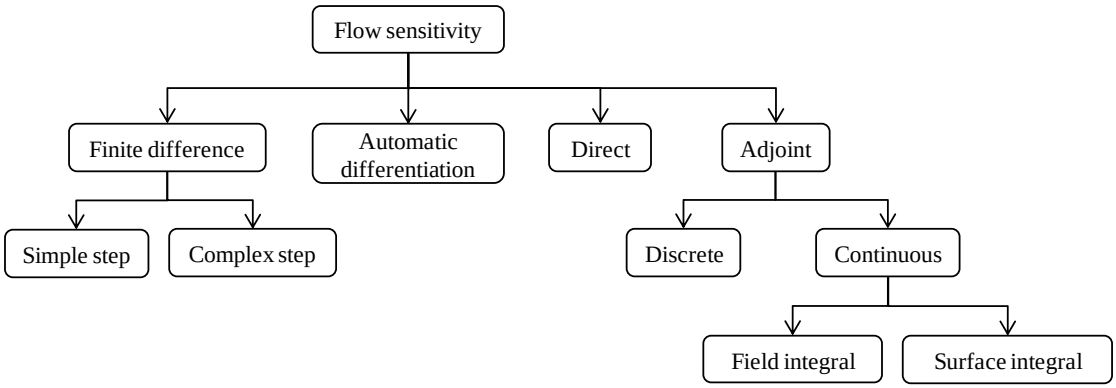


Figure 2.2 Classification of the known methods to compute the flow sensitivity.

The direct approach is based on the direct differentiation of the equations. This technique is equivalent to the adjoint in that it yields identical results for the derivatives and was first investigated by Sobieszczanski-Sobieski [4]. As noted by Narayanan and Chattopadhyay [21], the direct differentiation is generally used in problems with a large N_I (number of objective functions) and a small N_{DV} because the cost of solving the resulting linear system of equations is effectively proportional to the N_{DV}.

The breakthrough came in 1988, when Jameson [6] published an article in the *Journal of Scientific Computing* titled "*Aerodynamic Design via Control Theory*", where he explained, using the adjoint state, how to obtain the aerodynamic derivatives for transonic potential flow without incurring in issues related neither to the cost nor the accuracy of the gradient generally associated to the use of FDs. The article heavily drew from Lions [22] partial differential optimal control theory and Pironneau [5] who first applied it to fluid mechanics. Tab. 2.2 offers

a summary of the milestones related to the development of the adjoint method for aerodynamic optimisation.

Table 2.2 Adjoint method development milestones.

Year	Author(s)	Contributions
1971	Lion [22]	A theory for optimal control of systems of partial differential equations
1973	Pironneau [5]	First application applied to fluid mechanics
1988	Jameson [6]	First application applied to transonic potential flow
1994	Jameson and Reuther [23]	First application applied to the Euler equations
1998	Jameson, Pierce and Martinelli [24]	First application applied to the Navier-Stokes equations

The flow-adjoint method is based on simple differentiation chain rules. Although conceptually simple, efficient implementation of this technique requires an in-depth knowledge of the method itself. In order to understand, at least conceptually, what the main advantages of the adjoint technique are, let us consider the cost associated to the direct approach. The most computationally intensive step in both approaches is the solution of the respective resulting linear system. In the case of the direct method, as noted by Mader *et al.* [25], a linear system of N_I equations needs to be solved N_{DV} times, whereas using the adjoint method the same linear system needs to be solved only N_I times. Thus, the choice between these two methods depends largely on how the N_{DV} compares to the N_I. Therefore, if $N_{DV} \gg N_I$ the adjoint method is the preferred strategy, whereas if $N_{DV} \ll N_I$ the direct method would be instead preferred.

In order to justify the choice made for this work, it is necessary to recall the requirements: firstly, the maximum N_I is equal to two (drag and lift coefficients), and secondly it is necessary to compute the derivative of these two objective functions w.r.t. the entire surface. It is evident that the N_{DV} is very large because the $N_{DV} = N_{SM}$ ($N_{DV} \gg N_I$), therefore the adjoint is the most appropriate method for this work. However, it is recognised that when a parameterisation tool is used, the N_{DV} is lowered significantly. In this case, the benefits offered by the adjoint method

becomes relatively less appealing. Nevertheless, there is a another point in favour of the adjoint method even when the N_{DV} is reduced by employing a parameterisation tool, and this is its ability to provide accurate derivatives.

The above discussion has briefly covered the main techniques used to compute the flow sensitivity. Having in mind the focus of this work, i.e. efficient computation of the grid sensitivity in the discrete adjoint framework, Chap. 2.3.1 focuses on the differences and similarities between the discrete and the continuous formulation in much more details. The analysis is appropriate because it helps to introduce why these two approaches differ in the way the grid sensitivity is treated.

2.3.1 Discrete versus Continuous Adjoint Formulation

From a mathematical point of view, the adjoint equations are derived based on the linear perturbation of the governing flow equations (PDEs (Partial Differential Equations)). However, these equations need to be discretised from a numerical point of view as well. This can be done according to two formulations which are generally referred to as the discrete and continuous adjoint approach. For the continuous approach the governing partial differential equations are first linearised then their adjoint is constructed, discretised and then solved. For the discrete approach, the process is inverted as depicted in Fig. 2.3. To assess which method is more appropriate for this work the analysis is based accordingly to the metrics listed in Tab. 2.3.

In the case of large 3D simulations, the discrete approach requires a high memory to be available. What matters is the way the discrete flux Jacobian is stored because the discrete approach, to maintain consistency, requires the storage of the full flux-Jacobian [26].

One problem is the level of convergence that must be reached to obtain accurate derivatives, another is the condition such that the convergence is assured for the adjoint equation. In particular, it was shown by Giles *et al.* [27] that the dual-problem can have the same flow solver asymptotic convergence rate. Furthermore, Nadarajah and Jameson [28] established that the accuracy of the gradients obtained solving the adjoint system depends on the flow solution level of convergence only.

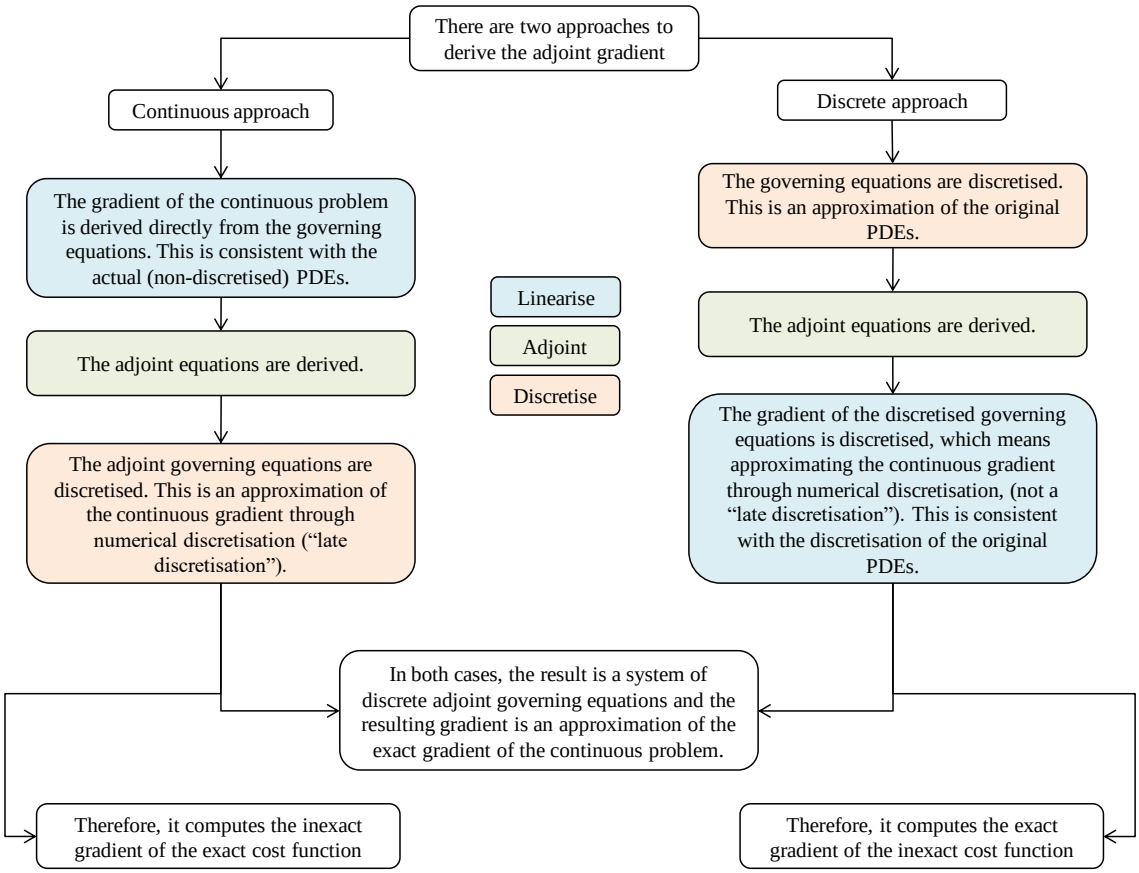


Figure 2.3 Discrete and continuous adjoint approach in their fundamental differences.

In terms of gradient accuracy, the order to which the equations are linearised and discretised is important. The process does not commute in general and some differences are expected depending on the order in which the linearisation is performed. As reported in Fig. 2.3, the discrete adjoint computes the exact gradient of the inexact cost function, whereas the continuous approach computes the inexact gradient of the exact cost function [28, 29]. As a result, the discrete adjoint gradient is consistent with that computed using FDs independently from the mesh refinement. However, Nadarajah and Jameson [28] proved that for both Euler and Navier Stokes solutions, to the limit of a refined mesh, the derivatives computed by the two methods converge to the same value. This means that for the continuous approach the gradient accuracy is governed by the flow residual and the mesh refinement, whereas for the discrete approach is governed only by the flow residual.

Lozano [30] noted that the continuous approach offers the choice to choose any discretisation scheme for the adjoint equations, whereas the discrete approach needs to keep numerical consistency w.r.t. the flow solver. For example, if a numerical scheme is used to compute the turbulence viscous terms in the flow solver, the flow Jacobian matrix should include these terms. This is translated in an advantage for the discrete approach from an implementation effort stand point of view, because the same flow solution numerical scheme can be used for the solution of the flow-adjoint equations. On the other hand, the continuous approach is not bounded to use the same numerical scheme.

It is not always easy to find the corresponding adjoint problem. Meaning that neither objective nor boundary conditions are easily converted into the proper adjoint equivalent. This happens because in the continuous adjoint the objective function is appearing in the boundary conditions, which requires a new derivation for each target function. An example is represented by the necessary adjoint boundary condition along the shock wave, which is very difficult to implement. However, in practice this is never used and as noted by Giles and Pierce [31] this choice does not cause much problem. The discrete adjoint can treat arbitrary cost function without implementing any special treatment.

The link with the physics is not always easy to make. To this regard, the continuous approach offers a better description of the physics for both the equations and the boundary conditions [20]. For example, the behaviour across shock and stagnation point can be more easily understood. Furthermore, regardless to the approach chosen, Giles and Pierce [20] noted that after the adjoint problem has been solved, the adjoint variables tell how a source term function influence the chosen function of interest. An application of this definition is offered by Jacques *et al.* [32] where the adjoint variables are shown to provide the sensitivity of the objective function w.r.t. a change in the residual. An example of this is also presented in Chap. 7.4.6.

Having analysed the main metrics reported in Tab. 2.3, it is time to draw some conclusions. The chief reason why the discrete approach was chosen over the continuous approach can be justified by the fact that, the discrete approach delivers accurate derivatives regardless to the mesh refinement.

The fact that the discrete adjoint provides the exact gradient of the inexact objective function is an important advantage, but it can also be regarded as a disadvantage because it requires the

computation of the grid sensitivity which is troublesome for large unstructured grids. This is the focus of the next chapter.

Table 2.3 Comparison between discrete and continuous adjoint approaches.

Metrics	Discrete approach	Continuous approach
Memory requirement	Demanding especially for large 3D cases	Does not suffer in terms of memory when large 3D cases are used as long as the surface integral is used
Convergence	In both cases, the accuracy of the gradient depends only on the flow-solution convergence	
Accuracy	Governed only by the flow-solution residual Computes the exact gradient of the inexact cost function	Governed by the flow-solution residual and ultimately by the mesh refinement Computes the inexact gradient of the exact cost function
Discretisation	Not flexible because it needs to use exactly the same flow solver discretisation scheme	More flexible because it does not need to use exactly the same flow solver discretisation scheme
Boundary conditions	Easier to obtain in a discrete fashion	Not all the time is possible to obtain them
Physical meaning	Difficult to understand	Easy to understand

2.3.2 Discrete Adjoint Approach

When state, costate and design space are all combined together, Ta'asan [33] suggested that the solution of such a problem can be visualised as the intersection of three hyperplanes as shown in the Fig. 2.4. With this in mind, the adjoint solution is represented by all those points that lay at the intersection between the state and costate hyperplane.

To derive the discrete adjoint formulation it is first necessary to discretise the governing equations followed by their linearisation. This second step can be done in two ways, by using the concept of Lagrangian multipliers or by direct differentiation of the objective function dependency. The Lagrangian approach (also known as costate) is generally used because the

problem is constrained (i.e. flow and mesh constraint). However, other applications, such as in error estimation, do not require the adoption of the Lagrangian variables because they are not constrained [34].

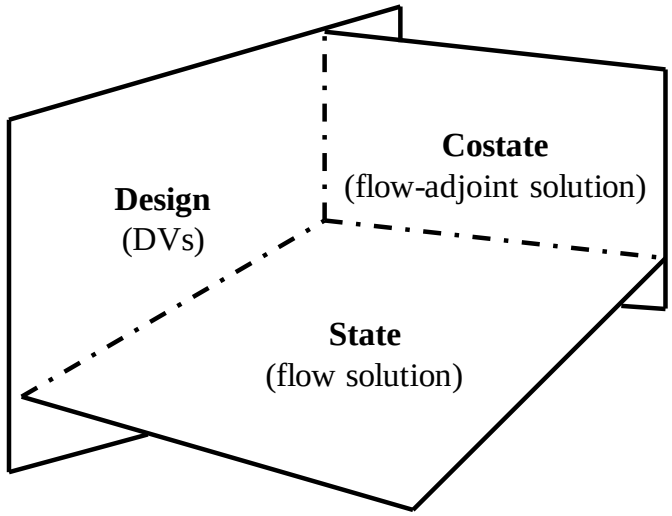


Figure 2.4 Representation of state, costate and design space.

The objective function depends on the mesh, $\{\boldsymbol{X}\}$, DVs, $\{\boldsymbol{D}\}$, and flow variables, $\{\boldsymbol{W}\}$. The flow-field solution is obtained by solving a set of highly non-linear equations and, as a consequence, the objective function is typically nonlinear as it depends on the flow state vector. The scalar objective function serves as a measure of the fitness of the design process and is assumed to be continuously differentiable. This is an assumption often made but rarely verified. However, there are cases where this assumption is not valid (shocks are a valid examples). A smoothing procedure could be employed to deal with this issue, but most numerical methods always show a certain amount of dissipation which smears discontinuity over the neighbouring cells mitigating the non differentiability of the objective function. In this work, no smoothing is performed and as noted by Anderson and Venkatakrishnan [35], this is in line with the discrete approach where discontinuity is not directly taken into account.

Consider the Lagrangian of the objective function (augmented objective function) w.r.t. the flow residual constraint, $\{\boldsymbol{R}\}$:

$$\mathcal{L}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X},\boldsymbol{\Lambda}) = I(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X}) + \{\boldsymbol{\Lambda}_F\}^T\{\boldsymbol{R}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X})\}, \quad \forall\,\{\boldsymbol{\Lambda}\},\{\boldsymbol{D}\} \tag{2.1}$$

where I is the objective function and $\{\boldsymbol{\Lambda}_F\}$ is the flow-adjoint vector. Since Eq. (2.1) is valid for all the DVs, the Lagrangian is identical to the objective function, i.e. $\mathcal{L} = I$ and, so for the derivatives, i.e. $\{d\mathcal{L}/d\boldsymbol{D}\} = \{dI/d\boldsymbol{D}\}$. Differentiation of the Lagrangian w.r.t. the DVs via chain rule yields:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{W}}\frac{d\boldsymbol{W}}{d\boldsymbol{D}} + \frac{\partial I}{\partial \boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial I}{\partial \boldsymbol{D}}\right\} + \{\boldsymbol{\Lambda}_F\}^T\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\frac{d\boldsymbol{W}}{d\boldsymbol{D}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}}\right] \tag{2.2}$$

The notation for a total derivative, d, was used, to distinguish it from the partial derivative, ∂. As noted by Mader *et al.* [25] this is done because generally the objective function is a function of multiple independent DVs, some of which have an explicit dependence and do not need the convergence of the residual and others which have not an explicit dependence and therefore need the convergence of the residual, i.e. another flow solution.

For a pure aerodynamic shape optimisation, where only the solid wall is updated, the DVs influence only the flow-field solution and objective function through the grid variations. As a consequence $\{\partial I/\partial \boldsymbol{D}\} = \{\boldsymbol{0}\}$ and $[\partial \boldsymbol{R}/\partial \boldsymbol{D}] = [\boldsymbol{0}]$. Therefore, rearranging Eq. (2.2) for the common terms yields:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{W}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right\}\left[\frac{d\boldsymbol{W}}{d\boldsymbol{D}}\right] + \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] \tag{2.3}$$

By doing so, the two most important and expensive components are highlighted, namely the sensitivity of the state variables w.r.t. the DVs, i.e. $[d\boldsymbol{W}/d\boldsymbol{D}]$, and the sensitivity of the volume grid nodes w.r.t. the DVs, i.e. $[d\boldsymbol{X}/d\boldsymbol{D}]$. The first expensive term can be eliminated by solving the following linear system:

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right]^T\{\boldsymbol{\Lambda}_F\} = -\left\{\frac{\partial I}{\partial \boldsymbol{W}}\right\}^T \tag{2.4}$$

Note that, although the original flow system is non-linear, the flow-adjoint system is linear. The flow-adjoint vector can be chosen such that it satisfies Eq. (2.4) and by doing so the matrix $[d\boldsymbol{W}/d\boldsymbol{D}]$ no longer appears in the total derivative. To be more precise, what is being

eliminated is not the sensitivity, but the dependency of the cost from the N_{DV}. The final sensitivity is then expressed as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\} \left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] \tag{2.5}$$

In conclusion the flow-adjoint method allows one to efficiently compute the gradient when the $N_I \ll N_{DV}$ at a cost of a single linear system solution and a subsequent product, thus making the method nearly independent from the N_{DV}. This leads to Eq. (2.5), where the second expensive term, i.e. $[d\boldsymbol{X}/d\boldsymbol{D}]$, needs to be addressed. This matter is discussed in Chap. 2.4.

2.3.3 Solving the Flow Adjoint System

The strategy implemented in the DLR TAU-Code to solve the adjoint linear system of equations is to re-use the same numerical scheme of the flow solver. In fact, the flow Jacobian can be taken as it is from the flow solver and reused for the adjoint system. By doing so, one fully exploits the fine tuning already available for the flow solver.

In theory, Eq. (2.4) can be solved by direct inversion of the flow Jacobian. However, this requires it to be constructed exactly which results in a very large and sparse matrix, thus the cost associated with it can become prohibitive. The matrix is never inverted directly, instead an implicit marching method is used to drive the residual to zero. The flow Jacobian is a very large and sparse matrix because it involves an extended neighbouring stencil which is higher on unstructured grids.

To maintain consistency the flow Jacobian needs to be exact. Therefore, the boundary conditions and all the laminar and turbulent contributions need to be taken into account. However, if all the elements of the Jacobian matrix are stored in memory, the computation time can be drastically reduced, but the memory requirement would be prohibitively large for 3D problems. One option is to store a reduced Jacobian where only the immediate neighbour information is considered. The remaining entries can then be computed on-the-fly each time the adjoint residual is required. In this way, the memory requirement can be remarkably reduced, however the computation cost is increased [36].

Different techniques have been used to solve the resulting adjoint linear system of equations. Nielsen and Anderson [37] employed a GMRES (Generalised Minimal RESidual) solver, an approach favoured also by Hicken and Zingg [38], whereas Mavriplis [39] used an explicit multigrid method taken directly from the flow solver. Although the adjoint system is linear while the flow system is non-linear, Dwight *et al.* [40] noted that sometimes it is as hard as the flow system to solve. It was proven experimentally by Giles *et al.* [27] that, if the non-linear flow problem converges asymptotically, the same is guaranteed also for the flow-adjoint problem. However, as noted by Dwight *et al.* [40] this is rarely the case. To solve this issue they proposed a method based on the recursive projection method.

The strategy used in this work is based on the restarted preconditioned ILU (Incomplete Lower Upper) GMRES algorithm developed by Saad and Schultz [41]. The method works by approximating the solution vector, constructed in a Krylov subspace using the minimal residual technique. As noted by Widhalm *et al.* [36], the Krylov loop helps to stabilise the linear adjoint solver especially when the non-linear flow solution is not fully converged.

In general, the adjoint solution needs less convergence than the primal solution, generally estimated in two-to-three orders of magnitude. If the primal (flow) solution is not sufficiently converged, the flow-adjoint solution will likely not converge at all. However, having the flow adjoint solution converged is not an assurance that the flow solution is sufficiently converged. As a general guideline, it is good practice to start from relaxed convergence values (e.g. $R_{flow} = 10^{-5}$ and $R_{adjoint} = 10^{-4}$) and decrease them until the gradient stops changing. As noted by Nadarajah and Jameson [28] and Kim *et al.* [42], this strategy is based on the knowledge that the adjoint gradient accuracy is sensitive to the convergence level of the flow solver only and insensitive to that of the flow-adjoint solver. The flow and flow adjoint convergence used in this work were set to $R_{flow} = 10^{-8}$ and $R_{adjoint} = 10^{-6}$ respectively.

2.3.4 Issues and Solutions

Dwight and Brezillon [26] identified three main difficulties associated with the discrete adjoint method, namely differentiation of the flow solver, memory requirement for the flow Jacobian and computation of the grid sensitivity for large meshes.

The discrete approach requires the discretisation of the flow solver before the linearisation is performed. This does not involve only the flow solver but boundaries, turbulence model and many other parts of the chain must be linearised accordingly. Failure to do so, would result into inaccuracies in the computation of the resulting derivatives. One solution would be to use hand differentiation which is not difficult by itself, but time consuming and prone to errors. An example of the complexity of the differentiation process is presented by Zymaris [29] where the discretisation of the SA turbulence model was manually derived. A solution to this problem is AD (Automatic Differentiation), but its direct use, brings other problem such as the memory overhead while used in reverse mode. To this regard, an interesting solution was proposed by Mader *et al.* [25] where they used AD just for some of the most difficult partial derivatives. The second problem is related to the memory requirement needed to store the discrete flow Jacobian. Unfortunately, this is acceptable for 2D simulations only as the requirement for 3D simulations is not practical. Dwight and Brezillon [26] offer a comparison of the memory requirement needed.

In the literature, there are two widely used approximations to tackle the aforementioned drawbacks, namely first-order accurate discretisation and constant eddy-viscosity assumption.

The goal of the first-order accurate discretisation is to reduce the memory by reducing the spatial stencil discretisation, hence the extension of the information stored to a fraction of the immediate neighbouring nodes. This significantly reduces the sparsity of the Jacobian for unstructured mesh and secondly reduces the stiffness of the non-linear system. A reduction of 50% in memory requirement can be achieved [26].

While solving a turbulent flow using time averaging, the mean-flow equations are coupled with the turbulence model. Consequently a rigorous analysis should consider them both as a state variables. However, most of the time, the adjoint equations take into consideration only the variation in the mean flow quantities and keep the variation in the turbulent (eddy) viscosity constant. This is done because an accurate linearisation of the turbulence model can be difficult as it exhibits complex dependency on both the flow variables and the distance from the wall. If the variation of the eddy viscosity turbulence model is considered *frozen* [24, 43], a reduction in the development time effort is obtained. Kim *et al.* [44] offered a comparison between constant and non constant eddy viscosity at transonic speed. The study concluded that the gradients are

very different as compared to the ones obtained by FDs and not all of them have the correct sign. It should be appreciated that the sign is of paramount importance if a direction of improvement must be provided. Another comparison for incompressible flow is offered by Zymaris *et al.* [29].

Lastly, the third problem is related to the CPU time and memory requirements associated with the computation of the grid sensitivity. This is strictly related to the mesh movement adopted and is the focus of the following literature review.

2.4 Mesh Sensitivity

2.4.1 Importance of Mesh Sensitivity

An adjoint-based aerodynamic optimisation involves the linearisation of the entire chain. By doing so, and assuming the shape is parameterised, three types of sensitivities are needed, namely flow, grid and shape sensitivity. The flow sensitivity derivation was extensively described in Chap. 2.3.2, whereas in this section the focus is on the mesh sensitivity. The cost associated with this computation can become very expensive, especially for large unstructured meshes.

It is interesting to see what are the reasons that lead to this conclusion with the support of Fig. 2.5. The link between the interior volume meshes, surface meshes and the geometric DVs can be expressed as:

$$\{\boldsymbol{X}\} = \{\boldsymbol{X}\left(\boldsymbol{S}(\boldsymbol{D})\right)\} \tag{2.6}$$

Direct differentiation w.r.t. the geometric DVs provides the complete grid sensitivity expression:

$$\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right] = \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.7}$$

The first RHS term is termed volumetric mesh sensitivity and this matrix has dimensions $[N_{VM} \times N_{SM}]$. It is related to the mesh movement or to the grid generator used depending on the

strategy chosen to update the mesh. The second RHS term is a smaller matrix with dimensions $[N_{SM} \times N_{DV}]$ and represents the surface mesh sensitivity, i.e. how the surface change w.r.t. a change in the geometric DVs. The analytical availability of this derivative depends on the choice of the shape parameterisation which is generally easily differentiated.

There is one special case, namely $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$ where the $[\partial \boldsymbol{S}/\partial \boldsymbol{D}]$ reduces to the identity matrix and no longer plays a role. This is the case where it is desired to visualise the gradient of the objective function w.r.t. the entire lifting surface, i.e. $\{dI/d\boldsymbol{S}\}$. Examples of such an application can be found in Palacios *et al.* [45] for supersonic flow regime using the continuous adjoint approach and Hinchliffe and Qin [46] for transonic flow regime using the discrete adjoint approach.

The discrete and continuous approach differ also from the way the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is treated. The continuous approach can be formulated (the so-called reduced gradient) in such a way that the need to evaluate the volumetric mesh sensitivity is eliminated [30]. The original version of the continuous approach [9] was based on the evaluation of the volumetric mesh sensitivity for structured grids which did not pose an issue in terms of CPU time and memory. However, unstructured grids feature a $N_{VM,SM}$ which is one or even two orders greater than structured grids. Following these difficulties, Weinerfelt and Enoksson [47] formulated a surface integral version of the continuous approach which was also derived independently by Jameson and Kim [9]. This formulation avoids the direct computation of the volumetric term and provides the gradient based on surface information only.

On the other hand, while working with the discrete approach the volumetric mesh term cannot be eliminated, and the approach shares the same issues in terms of CPU and memory, when unstructured large grids are used. To address this issue the approaches used to compute the linearised mesh movement need to be analysed. There are two possibilities, namely construct a mesh adjoint for any implicit (iterative) mesh movement or use any explicit (non iterative) mesh movement such as the DGM. The former strategy was proposed by Nielsen and Park [37] (see Chap. 2.5.6) which allows to decouple the cost of computing the volumetric mesh sensitivity from the N_{DV}. This approach is similar in its concept to what was done for the flow sensitivity. The latter option is investigated in the present work and a detailed analysis is presented in Chap. 5.3.

The differences registered in the way the grid sensitivity is treated, was the chief reason why Anderson and Venkatakrishnan [35] proposed an hybrid approach between the discrete and the continuous approach. This strategy dictates the calculation of the flow-adjoint vector with the continuous approach, which is then plug into the final discrete adjoint formulation in order to account for the effect induced by the grid sensitivity.

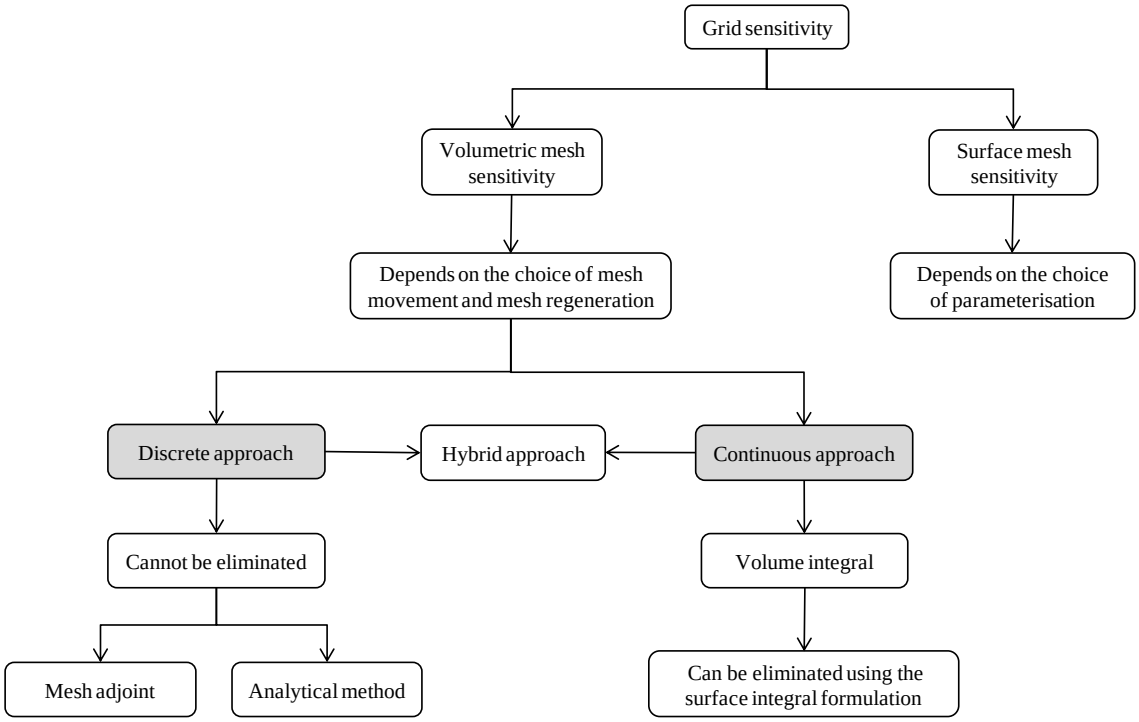


Figure 2.5 Discrete versus continuous adjoint approach on the grid sensitivity computation.

2.4.2 Extent of Mesh Sensitivity

It was highlighted why in Eq. (2.7) the volumetric mesh sensitivity is an expensive part of the chain. If the mesh is structured, analytical methods are available and the N_{VM} does not represent a problem. However, if no analytical mesh movement is known, an iterative method must be employed and the procedure quickly becomes very expensive as the N_{VM} grows. This has encouraged researchers to study a way to reduce the expense associated to the computation of the volumetric mesh sensitivity.

It can be highlighted that away from the wall, the internal volume mesh nodes experience a different change as compared to those close to the solid wall. Therefore, it seems reasonable to neglect the information stored in the internal volume mesh nodes far away from the wall. In fact, Nielsen and Anderson [48] proved that it is possible to reduce the differentiation of the volume mesh domain significantly. This is translated in a substantial saving while evaluation Eq. (2.7).

2.4.3 Simplification Regarding Flow and Grid Sensitivity

There are two major simplifications that applies to Eq. (2.5). They consist in dropping either the flow sensitivity, i.e. $\{\boldsymbol{\Lambda}_F\}^T[\partial \boldsymbol{R}/\partial \boldsymbol{X}]$ or the grid sensitivity, i.e. $[d\boldsymbol{X}/d\boldsymbol{D}]$.

The first approximation was proposed by Mohammadi [49] who suggested that there are particular cases where the grid sensitivity dominates. In this case, there is no need to construct the flow adjoint as the term for which the adjoint is sought is deleted as part of the simplification. This could be particularly useful when the objective function is an integral of a boundary quantity (e.g. drag and lift coefficients), and the DVs control only the solid wall shape. This approximation is based on the observation that the dominant part in the gradient is the derivative w.r.t. the geometry and not to the flow variables when a small change in the geometry causes very slight variations in the flow variables. The computation cost could be drastically reduced because the flow derivatives are not required. Subsequently, Kim *et al.* [50] compared this approximation with a full (flow and grid sensitivity) gradient for the design of supersonic high-lift devices. They concluded that for the case under analysis, this hypothesis gave reasonable results at the leading edge, but performed poorly at the trailing edge. A similar study was recently repeated by Wang *et al.* [51]. Their analysis confirmed the results reported by Kim *et al.* [50], i.e. the volume mesh sensitivity should not be neglected otherwise a large deviation in the gradient will be registered.

On the other hand, there are particular cases where the grid sensitivity can be ignored. This was dictated by the need to save computation time and memory while evaluating the grid sensitivity. In fact, as Kim *et al.* [50, 52] explained, the cost of solving the differentiated mesh movement is as much expensive as the mesh movement itself. Interestingly, this simplification works well

both at the leading and trailing edges. Compared to the case where the flow sensitivity was neglected, it seems that this simplification is more robust.

Another important point to mention is that both Mohammadi [49] and Kim *et al.* [50, 52] employed an Euler solver which, although carefully corrected for the boundary layer thickness, is not as accurate as a full Navier-Stokes solution. In fact, Anderson and Bonhaus [53] proved that for a full Navier-Stokes solution, if the grid sensitivity is not included, large deviations and incorrect signs in the gradients should be expected.

2.5 Existing Methods to Compute Mesh Sensitivity

There are many methods known in the literature to compute grid sensitivity. Fig. 2.6 offers a brief and concise classification. This is based on the fundamental difference between mesh regeneration and deformation which will be also discussed in the details in Chap. 4.3.

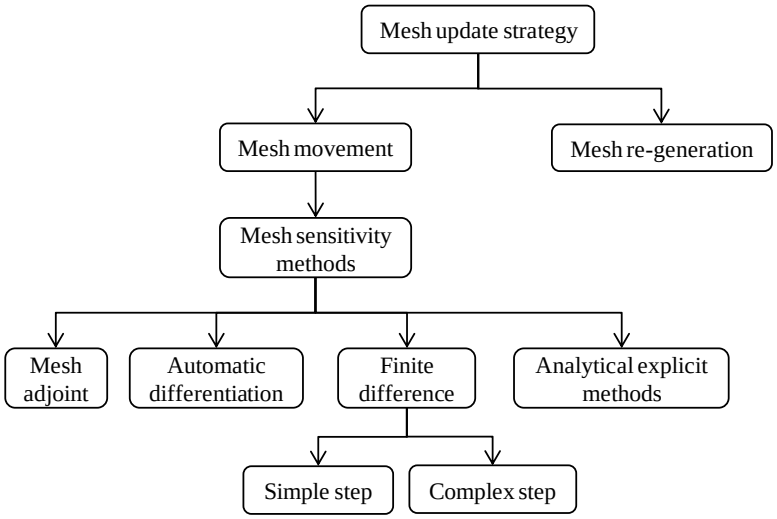


Figure 2.6 Classification of the methods to compute the mesh sensitivity.

2.5.1 Finite Step Differences

One of the first method used to evaluate the grid sensitivity was FD. The method has always been used to approximate the value of differentials either because no solution is known or because differentiation is so tedious that an approximation is considered qualitatively enough.

There are a series of issues with this approach. Accuracy is the main drawback of the method especially for non linear problems such as those of aerodynamic nature. Cost is also something that should not be underestimated. For instance, considering the case where the grid sensitivity needs to be computed, the number of operations that need to be performed is $N_{DV} \times m$, where m is the number of step size tested which is unknown a priori and is specific for each DV. It is obvious that the cost quickly becomes unmanageable.

In conclusion, most of the time the choice of this approach is motivated also by the cost of the mesh movement. If the volume mesh deformation is not expensive the cost of the FDs is not impeding, but becomes unpractical for more costly mesh movements such as the LE or spring analogy. In fact, Martins *et al*. [54] and Nemec and Zingg [55] used FDs thanks to the fact that their algebraic mesh movement was not expensive to use. Furthermore, both references employed structured grids for which the N_{VM} and the memory requirement is considerably low as compared to unstructured grids. Earlier examples can also be found in Hicks *et al.* [56] and Hicks and Vanderplaats [57].

2.5.2 Complex Step Differences

The method was originally investigated by Lyness and Moler [58], and Squire and Trapp [59], but was made popular in the aerodynamic optimisation community by Martins *et al.* [60]. This method is primarily used to avoid the additional time spent on the parametric study and the accuracy issue associated with FDs. This approach, by constructing a complex rather than a finite variable, avoids the cancellation error associated with FDs because the method is effectively insensitive to the choice of the step size. An arbitrarily small step can be used without incurring in the subtractive cancellation error associated to FDs. In fact, this allows one to use a much smaller step which is able to capture even the slightest deviation in the function

being evaluated. Additionally, it requires only one function evaluation and not two compared to the second order FDs.

On the downside, the method assumes that the mesh movement code can handle complex step which roughly doubles the memory requirements [61]. More importantly, the cost of computing the gradient remains proportional to the N_{DV}. Therefore as the per the case where FDs were used, the cost is effectively bounded by the N_{DV}.

Sturdza [62] applied this method to an optimisation of a supersonic aircraft, Newman *et al.* [63] used the method for the aero-structural design of a wing and Anderson *et al.* [64] used it for a transonic flow application. The complex-step method is often used as a verification tool to check the gradient accuracy as used by Martins *et al.* [65] who pioneered the method for MDO (Multidisciplinary Design Optimisation) applications.

2.5.3 Explicit versus Implicit Methods

In Chap. 2.3.2, it was discussed how the total final sensitivity can be expressed as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] \tag{2.8}$$

In Chap. 2.4.1, it has been discussed how to split the second RHS, leading to:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{2.9}$$

If $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$, Eq. (2.9) reduces to:

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right] \tag{2.10}$$

where the bottom line is that, within the discrete adjoint the evaluation of the large Jacobian represented by the second RHS in Eq. (2.10) needs to be addressed. From the surveyed literature, it has emerged the necessity to make a distinction between those methods which provide this Jacobian directly, i.e. explicit methods, and those that cannot, i.e. implicit methods.

To be more precise, the term *explicit* is used to refer to the availability of the volume mesh update (and therefore also of the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$) without having to solve through an iterative procedure a linear system of equations. If an iterative procedure is required to solve the resulting system of equations, the method is then referred to as *implicit*.

Most of the time, analytical mesh movements are easier to formulate for structured meshes and their applications have been largely confined to this type of mesh in the literature. In fact, the way the mesh node information is stored effectively follows the grid lines. This simple method cannot be used directly for unstructured meshes where this ordered structure does not exist. This should not exclude the existence of analytical methods also for unstructured meshes. In fact, for unstructured grids, the link between the spatial location of each node and its indices is no longer ordered and therefore more difficult to retrieve. Examples of the differentiation of explicit methods are given by Burgreen *et al.* [66] and Le Moigne and Qin [67] where they both differentiated manually the arch length algebraic mesh movement. Another example has been published by Fillola *et al.* [68] where the differentiation of a mesh movement based on the integral length was applied to a spoiler deflection for a large wing-body configuration.

On the other hand, examples of the differentiation of implicit methods can be found in Mavriplis [69] for the spring analogy mesh movement and Nielsen and Park [37] for the LE mesh movement. For all these methods the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is not available explicitly, meaning that an iteration procedure is required to compute its entries. Nielsen and Anderson [70] proved that, if the LE is used, a simple mesh movement can cost as much as 30% of the primal solution. In a subsequent paper, Nielsen and Park [37] showed that the differentiated version has a convergence similar to the mesh movement. Therefore, even the differentiated version is expected to take up to 30% of the flow solution for each DV.

2.5.4 Automatic Differentiation

AD refers to the automation of the hand derivation process. Method such as ADIFOR (Automatic DIFferentiation for FOrtran) [71, 72] and TAPENADE provide analytical results. One of the most common use of this method is to compute time-consuming hand-derivable partial derivatives. This greatly improves the development time especially for the adjoint code. One example of this application is the so-called ADjoint approach introduced by Mader *et al.*

[25] where the authors suggested to use AD only for some difficult-to-derive derivatives. When AD is used in the forward mode, to obtain the derivative w.r.t. the DVs a single evaluation is needed. However, when AD is used in reverse mode, a single evaluation is needed for each objective function, making it effectively independent from the N_{DV}. This last advantage is offset by the large memory requirement needed because the intermediate values of each DV have to be stored on memory.

Example of such a procedure are offered by Newman *et al.* [73] where AD was applied to a chain based on the spring analogy mesh movement, Newman and Taylor [74] where AD was applied to a mesh adaptation technique for an unstructured Euler flow solver code and Bischof *et al.* [75] where AD was applied to an algebraic mesh movement.

2.5.5 Differentiation of the Mesh Generation Routine

Although it has been emphasised that differentiating the mesh generation tool is somehow tricky and lengthy, there are cases where this has been nevertheless implemented for structured grids. The advantage is that accurate sensitivities can be obtained while reducing the chance to break the mesh and obtaining the best mesh quality possible. On the down side, all the time a mesh is regenerated, consistency in the discretisation error is no longer guaranteed because the connectivity is most likely going to be different (see also Chap. 4.3).

Sadrehaghighi *et al.* [76] and Korivi *et al.* [77] differentiated an algebraic grid generator, whereas Pagaldipti and Chattopadhyay [78] performed the same kind of differentiation by using an elliptic and hyperbolic grid generator. Another example is offered by Bischof *et al.* [75] where the authors used ADIFOR AD to differentiate a multi-block, three-dimension structured grid generator.

2.5.6 Mesh Adjoint

The motivation behind the article of Nielsen and Park [37] was “to eliminate”, meaning provide the sensitivity without incurring in a cost proportional to the N_{DV} by constructing a second adjoint linear system termed mesh adjoint. Having this in mind, two Lagrangian multipliers are

constructed here, one for the non-linear flow residual and one for the linear mesh movement residual, $\{\boldsymbol{T}(\boldsymbol{X},\boldsymbol{D})\}$:

$$\mathcal{L}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X},\boldsymbol{\Lambda}_F,\boldsymbol{\Lambda}_G) = I(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X}) + \{\boldsymbol{\Lambda}_F\}^T\{\boldsymbol{R}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X})\} + \{\boldsymbol{\Lambda}_G\}^T\{\boldsymbol{T}(\boldsymbol{X},\boldsymbol{D})\} \tag{2.11}$$

where $\{\boldsymbol{\Lambda}_G\}$ is the mesh adjoint vector. The differentiation of Eq. (2.11) follows exactly that reported in Chap. 2.3.2 with the only difference represented by the differentiation of the mesh movement w.r.t. the DVs which takes the following form:

$$\left[\frac{d\boldsymbol{T}}{d\boldsymbol{D}}\right] = \left[\frac{\partial\boldsymbol{T}}{\partial\boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial\boldsymbol{T}}{\partial\boldsymbol{D}}\right] \tag{2.12}$$

It follows that, under the assumption of pure geometric changes, the linearisation w.r.t. the DVs yields:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial\boldsymbol{W}} + \boldsymbol{\Lambda}_F^T\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{W}}\right\}\left[\frac{d\boldsymbol{W}}{d\boldsymbol{D}}\right] + \left\{\frac{\partial I}{\partial\boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{X}} + \boldsymbol{\Lambda}_G^T\frac{\partial\boldsymbol{T}}{\partial\boldsymbol{X}}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] + \left\{\boldsymbol{\Lambda}_G^T\frac{\partial\boldsymbol{T}}{\partial\boldsymbol{D}}\right\} \tag{2.13}$$

The flow-adjoint linear system of equations is constructed and solved as reported in Chaps. 2.3.2 and 2.3.3 respectively. Very much like what it was done for the flow derivatives, to eliminate the cost associated to the grid sensitivity derivative, i.e. $[d\boldsymbol{X}/d\boldsymbol{D}]$, the following mesh adjoint linear system of equations needs to be addressed:

$$\left[\frac{\partial\boldsymbol{T}}{\partial\boldsymbol{X}}\right]^T\{\boldsymbol{\Lambda}_G\} = -\left\{\frac{\partial I}{\partial\boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{X}}\right\}^T \tag{2.14}$$

which needs to be solved for the mesh adjoint vector. At this point, it is necessary to analyse the mesh movement technique because Eq. (2.14) effectively requires the linearisation of the mesh movement w.r.t. the volume mesh nodes. There are several methods used to update the mesh. For instance, Nielsen and Park [37] used the LE, whereas Mavriplis [69] used the spring analogy. A further detailed derivation is given for the LE in Chap. 5.5.

It has been discussed that the flow adjoint effectively decouples the cost of the gradient expense from the N_{DV} at the cost of solving an additional system of linear equations. However, the larger the N_{VM}, the more expensive the linear system solution becomes. Similarly, although the mesh adjoint method effectively makes the expense associated with the evaluation of the grid

sensitivity independent from the (geometric) N_{DV}, it does it at the cost of solving an additional linear system of equations that could be as hard to converge as the flow adjoint.

2.5.7 Integrated Geometry and Mesh Movement Approach

In the literature, there is also another less known method where both the surface and volume mesh nodes are updated using the same method. The advantage of this approach is the availability of an integrated fully consistent surface and volume mesh deformation.

There are two important applications of this approach. The first one is based on the RBF method and was proposed by Morris *et al.* [79], whereas the other is based on the B-splines method and was proposed by Hicken and Zingg [38]. The gradient computation of the surface w.r.t. the DVs, i.e. $[\partial \boldsymbol{S}/\partial \boldsymbol{D}]$, can be easily obtained. However, the sensitivity w.r.t. the volume mesh was computed using FDs by Morris *et al.* [79] whereas Hicken and Zingg [38] opted for the more accurate mesh adjoint approach.

To be more precise, these two approaches seems to be very similar to the FFD proposed by Sederberg and Parry [12]. Indeed, they share the similarity of considering both surface and volume mesh points as a cloud of independent points. However, where the original FFD performs no inverse fitting on the original geometry thanks to its local parametric coordinates transformation (see Chap. 6.3) the B-spline and RBF require to retrofit the original surface. This discussion is covered again in Chap. 7.4 from a consistency point of view.

2.6 Summary

The literature review discussed in this chapter introduced the reasons why a gradient-based optimisation framework was preferred over other search direction methods. The pros and cons were analysed and it was explained how the advantages outweighed the disadvantages for the test cases investigated in this work.

The advantages of the adjoint method for the computation of the flow sensitivity were described. Within this framework, the discrete version was chosen over the continuous method on the grounds that the gradients accuracy depends only on the flow solution and not on the

mesh refinement level. However, by doing so, care must be exercised while computing the grid sensitivity. For this last one, the methods available were surveyed and a summary of this analysis is offered in Tab. 2.4 to help the reader understand the sometimes subtle differences.

It was highlighted how important is to consider how the cost scales with the mesh size. In this regard, the last known method published to address the computation of the grid sensitivity is the mesh adjoint proposed by Nielsen and Park [37]. Its formulation, advantages and its current bottleneck were identified in the solution of a large and stiff linear system of equations. This served as an introduction and justification for the new method presented in the current work which is based on the DGM mesh movement.

Table 2.4 Pros and cons of the methods known to evaluate the grid sensitivity.

Methods	**Pros**	**Cons**
Finite step differences	Easy implementation	Prones to cancellation and round-off error Cost scales linearly with the N_{DV} Poor accuracy and consistency
Complex step differences	No cancellation error	Assumes the code can handle complex variable The cost is proportional to the N_{DV}
Analytical	No convergence issue Low memory and CPU requirements	The pre-processing cost is proportional to the N_{DV}
Automatic differentiation	Easy to implement Largely automatic	Increase in memory when used in reverse mode
Mesh generation routine	Large deformation can be enforced Good quality mesh is maintained	Limited to simple differentiable routine Generally limited to structured mesh Non-consistent gradient
Mesh adjoint	Independent from the N_{DV}	Needs the solution of a large linear system

Chapter 3 Governing Equations and Numerical Methods

3.1 Introduction

In this chapter different types of governing equations are briefly surveyed. This is then followed by a detailed description of the appropriate set of equations used to describe the transonic flow regime studied in this work.

The chapter is divided in two sections. The first part explains why the Navier-Stokes equations are preferred over other systems of equations, such as the Euler equations. The time and density averaging of the Navier-Stokes equations are introduced, along with the turbulence model needed to close the equations. The second part focuses on the actual spatial, temporal discretisation methods and the acceleration techniques used. A discussion of why certain algorithms are chosen is provided.

The chapter concludes with a summary of the most important points.

3.2 Transonic Flight

The transonic regime referrers to that local (note that not always local coincide with the free-stream condition) velocities window which can be split into three distinct domains, namely subsonic, sonic and supersonic domain. Thus, a Mach number generally considered in the range from 0.6 to 1.2. In 2D, the region where the local sonic speed is met is a line, whereas in 3D is a surface. This separates the subsonic from the supersonic region.

Commercial civil aircraft fly close to the transonic regime because the Mach number times the lift over drag ratio, ($M \cdot L/D$), which is one of the metric used to approximate the aircraft cruising efficiency, stays constant up to a certain value of the Mach number. Therefore, it pays off cruising close to the upper limit of this range which is the transonic regime [1]. As the aircraft operates at the transonic regime the wings experience local sonic flows and shocks are likely to appear. The shock is nothing more than the way the flow is recompressed after it has locally exceed the local sonic speed.

The question now becomes how to properly model the physics peculiar of the transonic flow. There are different set of equations used in CFD. These were all derived from the full Navier-Stokes set of equations under specific hypothesis and are intrinsically limited in their applications. There are many hypotheses that can be used, such as dropping viscosity or/and vorticity etc.... Their original goal was to find a way to reduce the solution time, however the downside was a reduction in the applicability spectrum. In order to give a flavour of the many approximations available and justify why the Navier-Stokes equations were chosen, only the most common set of equations are depicted in Fig. 3.1 (adapted after [1]).

Shocks interact with the boundary layer right where modelling viscous phenomena is important. This becomes more and more important if only small changes (tweaks) are sought from an already optimised geometry. From here, it is clear that viscosity, vorticity and change in density are all important factors and for these reasons the Navier-Stokes equations are deemed appropriate to model the transonic flow physics.

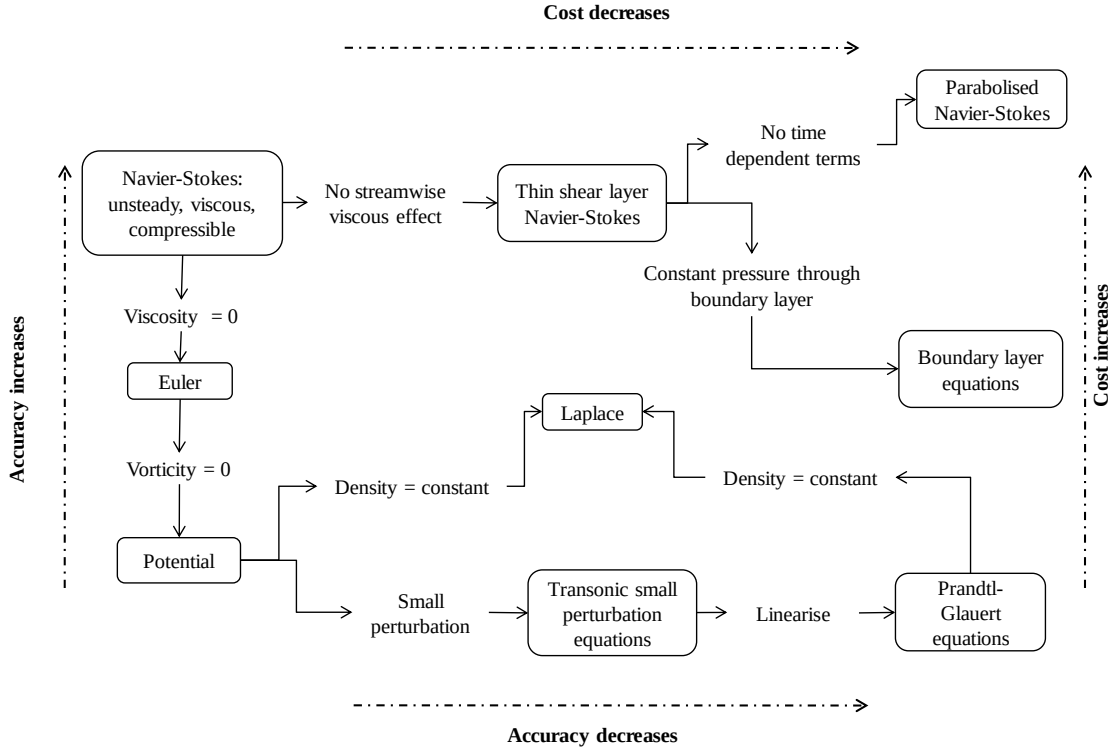


Figure 3.1 Navier-Stokes equations and some possible simplifications.

3.3 Navier-Stokes Equations

The compressible Navier-Stokes equations are a coupled nonlinear mixed (hyperbolic, parabolic and elliptic) PDEs governing the unsteady compressible viscous flow. These represent the conservation of mass, momentum and energy. The vast majority of today CFD codes express the Navier-Stokes PDEs in integral form:

$$\frac{\partial}{\partial t}\iiint_{\Omega}\{\boldsymbol{W}\}\,dV+\iint_{\Gamma}\{\mathbf{Q}\}\cdot\mathbf{n}\,dS=0 \tag{3.1}$$

where Ω is the computation volume, Γ is its boundary, $\{\boldsymbol{Q}\}$ is the flux tensor, $\boldsymbol{n}$ is the boundary normal pointing vector, dV and dS are the discretised volume and surface respectively. The time derivative can be ignored if a steady-state solution is considered. This is always the case for all the simulations presented in this work. The state variables vector is defined as:

$$\{\boldsymbol{W}\}=\{\rho,\rho u,\rho v,\rho w,\rho E\}^{T} \tag{3.2}$$

where u, v, w are the velocity components along the Cartesian x, y, z directions, ρ is the density and E is the total energy. To facilitate the numerical solution of the equations, the flux tensor, is decomposed along the three Cartesian coordinates:

$$\{\mathbf{Q}\}=\left\{\boldsymbol{F}^{i}+\boldsymbol{F}^{v}\right\}\cdot\boldsymbol{i}+\left\{\boldsymbol{G}^{i}+\boldsymbol{G}^{v}\right\}\cdot\boldsymbol{j}+\left\{\boldsymbol{H}^{i}+\boldsymbol{H}^{v}\right\}\cdot\boldsymbol{k} \tag{3.3}$$

where $\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}$ represent the unit vectors along x, y, z. $\boldsymbol{F}^{i}$, $\boldsymbol{G}^{i}$, $\boldsymbol{H}^{i}$ and $\boldsymbol{F}^{v}$, $\boldsymbol{G}^{v}$, $\boldsymbol{H}^{v}$ are the inviscid and viscous flux vectors along x, y, z defined respectively as:

$$\left\{\boldsymbol{F}^{i}\right\}=\begin{Bmatrix}\rho u\\ \rho u^{2}+p\\ \rho uv\\ \rho uw\\ \rho Hu\end{Bmatrix}\qquad\left\{\boldsymbol{F}^{v}\right\}=\begin{Bmatrix}0\\ \tau_{xx}\\ \tau_{xy}\\ \tau_{xz}\\ u\tau_{xx}+v\tau_{xy}+w\tau_{xz}+\kappa\dfrac{\partial T}{\partial x}\end{Bmatrix}$$

$$\left\{\boldsymbol{G}^{i}\right\}=\begin{Bmatrix}\rho v\\ \rho uv\\ \rho v^{2}+p\\ \rho vw\\ \rho Hv\end{Bmatrix}\qquad\left\{\boldsymbol{G}^{v}\right\}=\begin{Bmatrix}0\\ \tau_{yx}\\ \tau_{yy}\\ \tau_{yz}\\ u\tau_{yx}+v\tau_{yy}+w\tau_{yz}+\kappa\dfrac{\partial T}{\partial y}\end{Bmatrix} \tag{3.4}$$

$$\{\boldsymbol{H}^i\} = \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho Hw \end{Bmatrix} \qquad \{\boldsymbol{H}^v\} = \begin{Bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + \kappa \dfrac{\partial T}{\partial z} \end{Bmatrix}$$

H is the enthalpy, p the pressure, κ is the thermal conductivity coefficient, T is the temperature and lastly τ_{ij} are the viscous stresses defined as:

$$\begin{aligned} \tau_{xx} &= 2\mu \frac{\partial u}{\partial x} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \\ \tau_{yy} &= 2\mu \frac{\partial v}{\partial y} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \\ \tau_{zz} &= 2\mu \frac{\partial w}{\partial z} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \\ \tau_{xy} &= \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \tau_{xz} &= \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \tau_{yz} &= \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \end{aligned} \tag{3.5}$$

where λ is the second coefficient of viscosity which can be computed accordingly to the Stokes' hypothesis:

$$\lambda = -\frac{2}{3}\mu \tag{3.6}$$

where μ is the dynamic viscosity computed as a function of the Sutherland's law:

$$\mu = \mu_0 \left(\frac{T}{T_0} \right)^{3/2} \frac{T_0 + S}{T + S} \tag{3.7}$$

where, for air, constants μ_0, T_0, S (Sutherland's temperature constant) take the following values: $1.7894 \times 10^{-5}\ [kg/(m \cdot s)]$, $288.15\ [K]$, $110.4\ [K]$. The thermal conductivity allows to compute the heat fluxes as:

$$q_x = -\kappa \frac{\partial T}{\partial x}$$
$$q_y = -\kappa \frac{\partial T}{\partial y} \tag{3.8}$$
$$q_z = -\kappa \frac{\partial T}{\partial z}$$

where the thermal conductivity coefficient is defined as:

$$\kappa = \mu \frac{c_p}{Pr} \tag{3.9}$$

where c_p and Pr are the specific heat at constant pressure and the Prandtl's number with value equal to $1{,}006.43\ [J/(kg \cdot K)]$ and 0.72 respectively. Finally, the total enthalpy is defined as:

$$H = E + \frac{p}{\rho} \tag{3.10}$$

where the relation between pressure and density is established by the ideal gas equation:

$$p = \rho RT \tag{3.11}$$

where R is the air gas constant taking value equal to $287.05\ [J/(kg \cdot K)]$.

3.4 Navier-Stokes Equations Averaging

If the discretised Navier-Stokes equations were to be solved by numerical brute force, a technique known as DNS (Direct Numerical Solution), the entire range of both spatial and time turbulence scales would be resolved. To do so, the volume mesh used to discretise the domain, would be very dense and a very small time step would be necessary to properly capture small scale turbulence fluctuations. According to Tennekes and Lumley [80], such a computation (in terms of floating point operations) is equal to the total number of discrete mesh points times the time step used. This can be shown to be proportional to the third power of the Reynolds number. Based on that, Parviz and Kim [81] estimated that a DNS simulation for a commercial aircraft employing a teraflop computer will take something in the range of thousands of years.

To lower such a high demanding computation power, the volume mesh density and the time step must be reduced. This can be done adopting (relative) less expensive methods, such as RANS (Reynolds-Averaged Navier-Stokes) or LES (Large Eddy Simulation). The idea behind these techniques is not to resolve the entire turbulence scales, but instead to model them as briefly shown in Tab. 3.1.

Table 3.1 Comparison between DNS, LES and RANS.

Method	**Modelling of turbulent scales**	**Mesh resolution**		**Computation time**		**Applications**
		N_{SM}	N_{VM}	**Time steps**	**Total operations**	
DNS	Resolves all the scales hence no modelling	10^{16}	10^{17}	10^{8}	10^{25}	Small scale turbulence structures
LES	Models sub-grid scales but resolves large scales	10^{9}	10^{12}	10^{8}	10^{20}	Large separation such as vortex shedding from the nacelles
RANS	Models all the small and big scales	10^{6}	10^{7}	10^{4}	10^{11}	Steady-state attached or mildly separated flows

This chapter concentrates on the RANS method for reasons that can be reduced to the specific flight condition (transonic flow) being simulated and the available amount of time. The Reynolds' method consists in averaging in time. Given a time and space dependent quantity, ϕ, its value is defined as:

$$\varphi = \bar{\varphi} + \varphi' \tag{3.12}$$

where φ' is the fluctuating part and the Reynolds time averaged is defined as:

$$\bar{\varphi} = \frac{1}{\Delta t}\int_{\Delta t} \varphi \, dt \tag{3.13}$$

However, to properly consider compressibility effects, the flow variables are instead Favre-averaged as:

$$\varphi = \tilde{\varphi} + \varphi'' \tag{3.14}$$

where, $\tilde{\varphi}$ is the mean part whereas φ'' is the fluctuating part. The former is the defined as:

$$\tilde{\varphi} = \frac{1}{\bar{\rho}\Delta t}\int_{\Delta t} \rho\varphi \, dt \tag{3.15}$$

where $\bar{\rho}$ is the Reynolds-averaged density. By doing so the equations are then termed FANS (Favre Averaged Navier-Stokes). Therefore, the following holds:

$$\begin{aligned} u &= \tilde{u} + u'' \\ v &= \tilde{v} + v'' \\ w &= \tilde{w} + w'' \\ p &= \bar{p} + p' \\ \rho &= \bar{\rho} + \rho' \\ T &= \tilde{T} + T'' \end{aligned} \tag{3.16}$$

This process saves computation time as compared to a full-scale DNS analysis, but there is a loss of fidelity in terms of captured physics. In fact, the approximation can be assumed to be valid and accurate only for attached or mildly detached flow where the flow streamlines curvature is generally small. The substitution of the newly defined state variables in Eq. (3.16) back in Eq. (3.1) creates two new unknowns. The first is generally referred to as the turbulent Reynolds stresses:

$$\bar{\tau}_{ij} = -\overline{\rho u_i'' u_j''} \tag{3.17}$$

where subscript i, j correspond to the three Cartesian directions x, y, z. On the other hand, the second newly created quantity is the turbulent heat fluxes:

$$\bar{q}_i = -c_p \overline{\rho u_i'' T''} \tag{3.18}$$

The newly created turbulent Reynolds stresses and the turbulent heat fluxes, coming from the time averaging describe the momentum transfer due to the turbulent flow. Since the turbulent

Reynolds stresses are symmetric, i.e. $\overline{u_i'' u_j''} = \overline{u_j'' u_i''}$, the tensor is symmetric and considering a three dimensional flow it effectively consists of six unknown variables. This obviously creates more unknown than equations, a problem known as *closure*. To solve this problem, Boussinesq (1877) proposed the approximation that the Reynolds stresses are linearly related to the trace-less mean strain rate where the proportionality factor is called eddy (turbulent) viscosity, μ_t. Thus, in formula:

$$\overline{\rho u_i'' u_j''} = \mu_t \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial \tilde{u}_k}{\partial x_k} \right) - \frac{2}{3} \delta_{ij} \bar{\rho} k \tag{3.19}$$

where δ_{ij} is the Kronecker's delta and k is the turbulent kinetic energy. In the SA approach (Spalart-Allmaras) [82] the last term is ignored. The turbulent viscosity is consider an isotropic scalar quantity, but there are cases where this assumption is seriously violated, therefore the term *approximation* is appropriate. As a consequence, large deviations should be expected every time one of the following conditions is present: flows with sudden change of the mean strain rate, significant streamline curvature, rotation, stratification, secondary flows and boundary layer separation or reattachment.

3.5 Spalart-Allmaras Turbulence Model

To close the system resulting from the averaging a turbulence model needs to be adopted. The turbulence model selected for this work is the SA [82] one-equation model. This turbulence model, like all the others, allows the computation of the mean flow variables without calculating the full time-dependent flow field. Before proceeding to describe the turbulence model, it is important to justify why the SA turbulence model was chosen.

The choice was dictated by three factors: available time, difficulty related to the linearisation process and the resulting flow-Jacobian conditioning. The available computation power is a constraint when it come to the choice of the turbulence model. For this time-averaged solution the SA model represent a reasonable compromise. The time consuming differentiation process can be facilitated by using AD, but the memory overhead while used in reverse mode may represent an issue. Even in this case, although the differentiation of more complex turbulence models such as the Chien $k-\omega$ [83], Wilcox $k-\omega$ [84] or Menter SST (Shear Stress

Transport) [85] is facilitated, one needs to realise that this is not enough to make them good candidates for a gradient-based optimisation loop. In fact, the most important reason why such a relative simple turbulence model was chosen, i.e. SA, lays on the resulting well-conditioned flow Jacobian, something difficult to achieve for more complex turbulence models. In fact, Widhalm *et al.* [36] noted that the poor conditioning of the resulting adjoint linear system of equations (see Chap. 2.3.2) resulting from using other more complex turbulence models, can be so severe, especially for 3D cases, to require the adoption of the frozen turbulence assumption which reduces the accuracy of the resulting gradients. In conclusion the SA turbulence offered the best option available to the author in terms of accuracy, consistency and stability.

Regardless to the particulars of each turbulence model, it is possible to write all the turbulence models for compressible flows in a simple form as follows:

$$\frac{D(\bar{\rho}\varphi)}{Dt} = \frac{\partial(\bar{\rho}\varphi)}{\partial t} + \frac{\partial(\bar{\rho}u_i\varphi)}{\partial x_i} = S_P - S_D + GD \tag{3.20}$$

where D/Dt is the material derivative, φ is the transported quantity, GD is the Gradient Diffusion term, whereas S_P and S_D are the Production and Destruction Source term respectively. The compressible GD term reads as:

$$GD = \left(\frac{\partial}{\partial x_i}\left(\frac{\mu_{eff}}{\sigma}\frac{\partial\varphi}{\partial x_i}\right)\right) \tag{3.21}$$

where the sum of the laminar (or molecular) and turbulent (eddy) viscosity is generally referred to as effective viscosity, therefore $\mu_{eff} = \mu_l + \mu_t$. The SA one-equation turbulence model solves a convective-diffusive problem describing a transport equation for the eddy (i.e. turbulent) viscosity. For this turbulence model the S_P, S_D and *GD* terms are defined as:

$$GD = \left(\frac{\partial}{\partial x_i}\left(\frac{\mu_l + \tilde{\mu}}{\sigma}\frac{\partial\tilde{\nu}}{\partial x_i}\right) + \rho\frac{c_{b2}}{\sigma}\left(\frac{\partial\tilde{\nu}}{\partial x_i}\right)^2\right) \tag{3.22}$$

$$S_P = \rho c_{b1}\tilde{S}\tilde{\nu} \tag{3.23}$$

$$S_D = \rho c_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2 \tag{3.24}$$

where $\tilde{\nu}$ is the modified kinematic viscosity and d is the distance from the wall which is an important factor because when differentiated this is what causes the largest difference between frozen and non frozen assumption results as shown by Zymaris *et al.* [29]. The eddy (turbulent) viscosity is then defined as:

$$\mu_t = \rho \nu_t \tag{3.25}$$

where the turbulent kinematic viscosity, ν_t, is obtained as:

$$\nu_t = f_{v1}\tilde{\nu}, \qquad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad \chi = \frac{\tilde{\nu}}{\nu_l} \tag{3.26}$$

After Eq. (3.20) is solved for $\tilde{\nu}$, the eddy viscosity is computed as in Eq. (3.25). The definition of the S_P and S_D terms vary accordingly to the model (SA [82], SA-Edwards [86], SA-Strain-Adaptive Linear [87]). In the SA model, the scalar velocity gradient uses an augmented definition of vorticity:

$$\tilde{S} = S + \frac{\tilde{\nu}}{k^2 d^2} f_{v2} \tag{3.27}$$

where:

$$S = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.28}$$

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{3.29}$$

The destruction term is defined through the wall-blockage function defined as:

$$f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6} \tag{3.30}$$

with the following limiter function:

$$g = r + c_{w2}\left(r^6 - \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}\right) \qquad r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \tag{3.31}$$

where d is the distance from the wall and the rest of the empirical constants are defined as follows:

$$\kappa = 0.41, \quad c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{(1 + c_{b2})}{\sigma}, \quad c_{w2} = 0.3, \quad c_{w3} = 2, \quad c_{v1} = 7.1, \quad \sigma = 2/3 \tag{3.32}$$

The version implemented in the DLR TAU-Code does not use wall functions (thus the dimensionless wall distance parameter y^+ should be around 1), and is marched in time to a steady-state solution using the same implicit-step scheme used for the flow solver (see Chap. 3.6).

In a subsequent article Spalart and Rumsey [88] explained how to take care of numerical instabilities due to the negative value of the turbulent viscosity and in particular of constant f_{v2}. Alternatively, the formulation proposed by Edwards and Chandra [86] is a good workaround.

From the series of DPWs (Drag Prediction Workshops) [89] it was established the strong dependency of the aerodynamic coefficients from the choice of the turbulence model. This is because the transonic aerodynamic flow-field is characterised by stagnation points, regions of significantly strong adverse and proverse (favourable) pressure gradients, and regions of laminar or turbulent attached/detached flows. What matters is the inherent inconsistency of any turbulent model to predict the points listed above. Of course, RANS, LES, hybrid RANS-LES and DNS simulation would most definitely improve the above-mentioned points prediction, but at much higher cost and the linearisation of this method would make the implementation even more challenging.

3.6 Flow Solution Methods

Real world phenomena, described by the continuous Navier-Stokes PDEs, have two peculiar features: they are continuous, hence they do not admit discontinuity and secondly they are so complex that an analytical solution is rarely available. The only technique available is to discretise the equations and apply an iterative algorithm to solve the resulting system. An approximation is sought hoping that it will be close to the continuous problem.

From here, CFD is the art of replacing the above-mentioned integral equations with their discretised algebraic form. By doing so, the continuous domain, where the flow variables are defined everywhere in the domain, is turned into a discretised form where each finite (mesh) node in the domain is now an approximation of the continuous neighbouring field. The spatial discretisation can be performed using FDM (Finite Difference Method), FEM (Finite Element Method) or FVM (Finite Volume Method), whereas the time discretisation can be performed either by using and explicit or implicit schemes.

To avoid a mere description of the DLR TAU-Code [90] flow solver and of the plethora of methods available in the literature, the following chapters describe only the options used in this work.

3.6.1 Spatial Discretisation

It was mentioned that the continuous physical domain must be discretised. The most used technique is the FVM method for reasons that can be briefly reduced to formulation simplicity, mesh type independency and easy implementation. The spatial discretisation is done directly on the original primary grid, thus no coordinates transformation is needed. The FVM is based on the direct discretisation of the NS equations so that the conservation of mass, momentum and energy are also maintained by this spatial discretisation scheme. Furthermore, the Rankine-Hugoniot relations, at solution discontinuities (such as a shockwaves), are satisfied. This is a desirable feature especially for transonic flows.

There are two versions of the FVM, namely cell-centred (flow variables stored at the centre of the mesh) and cell-vertex (flow variables stored at the vertices of the mesh). The choice between the two is dictated by the type of mesh used, discretisation error, boundary treatment and time-dependent flow. The one implemented in the DLR TAU-Code flow solver is the cell-vertex FVM which divides the computational domain into smaller elements (referred to as control volumes) where the flow variables are stored at the dual mesh (i.e. secondary cell) vertices (see Fig. 3.2).

For each finite volume dual cell associated to any vertex, i, the discrete version of Eq. (3.1) takes the following form:

$$\Omega_i \frac{\partial W_i}{\partial t} + R_i(W_i) = 0 \tag{3.33}$$

where W_i is assumed to be constant inside the dual-cell domain (blue polygon in Fig. 3.2) and the flux terms are dominant in the same dual-cell domain. Eq. (3.33) states that, for each control volume in the computational domain, the rate of change of the state variables must be equal to the fluxes at the boundaries. The residual associated to each state variable W_i, associated to its dual volume cell, is then computed as the sum of the viscous and inviscid fluxes on each boundary:

$$R_i(W_i) = \sum_{i=0}^{N_f - 1} \left(Q_{ij}^i - Q_{ij}^v\right) S_{ij} \tag{3.34}$$

where N_f is the number of dual-cell control volume faces.

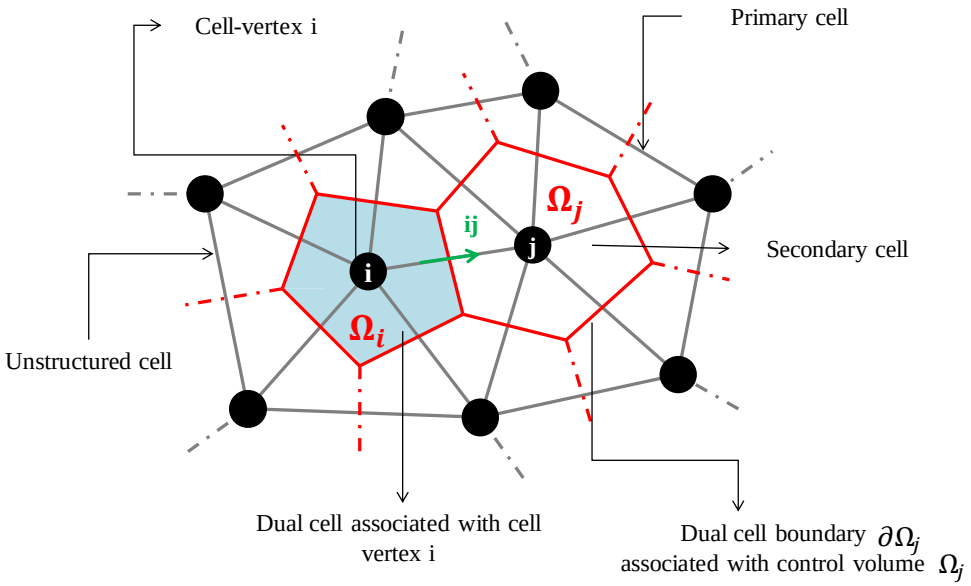


Figure 3.2 Primary and secondary grid for the cell-vertex FVM.

3.6.2 Temporal Discretisation

All the problems presented here are assumed to be at a steady state. However, while solving the FANS system of equations, the solution is marched in time. There are essentially two options to march, namely via explicit or implicit methods. The difference between the two, in a nutshell, is that implicit methods allow to use a much larger time step size which in turn accelerates the convergence to a steady state solution, but require much more memory, whereas explicit methods require a smaller step size while using less memory. For time-averaged solutions involving large cases, implicit methods are widely used, however the memory consumption may become a problem. For this reason, another class of methods which lays between explicit and implicit strategies was recently proposed. One of these is the preconditioned LU-SGS, (Lower-Upper Symmetric-Gauss-Seidel) [40, 91], implemented in the DLR TAU-Code which is the one used in this work.

Strictly speaking preconditioning was developed for low-Mach number flows, however since most of the compressible flow codes require to solve a solution domain of mix compressible and incompressible flows, these have found wide applications also in high-speed aerodynamics. The reason why preconditioning is used lays on the large disparity between convected information travelling at the fluid speed and the local speed of sound. This is also reflected in the conditioning of the matrix (i.e. ratio between the largest and the smallest eigenvalue). By doing so, the eigenvalues are effectively changed, i.e. equalised, which is reflected in an acceleration to convergence [92].

The LU-SGS first starts decomposing the matrix A representing the discretised equations into a lower, L, and upper, U, part such that $A = L + U$. The most important feature of this scheme is that there is no need to store explicitly the upper and lower matrices with an evident saving in memory. Then, the Gauss-Seidel algorithm is used to solve the resulting system.

3.6.3 Flux Discretisation

The other major obstacle is the evaluation of the convective and diffusive fluxes at the right and left side of each dual cell generated by the FV discretisation. The solution is known only at the cell vertices, therefore it is necessary to have a scheme capable to estimate their values at each

FV element interface. Fluxes can be discretised either using a central or upwind scheme. For all the computations presented in this thesis, the fluxes are discretised using a central method with scalar dissipation.

The scheme used in this thesis is the JST (Jameson-Schmidt-Turkel) [93] scheme. Let us consider a dual cell volume where it is desired to compute the convective flux between node i and j as depicted in Fig. 3.2. This can be expressed by the sum of the convective terms subtracted by the dissipative terms which are necessary for stability reason. This may be written as:

$$Q_{i,j}^{i} = \frac{1}{2}\left(Q_i^i + Q_j^i\right) \cdot n_{ij} - \frac{1}{2} D_{ij}^{i} \tag{3.35}$$

The first RHS contains the convective fluxes, whereas the second RHS contains the dissipation terms. This last is needed, as noted by Jameson [94], in order to avoid oscillation at the shock region and to damp high frequency oscillations in general, something that a second order scheme cannot achieve. This helps the residual to reach a steady-state. There are two methods to compute the dissipation term: scalar or matrix artificial dissipation, where the former is the one used in this work. On the other hand, viscous (diffusive) fluxes contrary to convective (inviscid) fluxes do not suffer from stability problem and for this reason, artificial dissipation is not added.

3.6.4 Construction of the Gradient

It is known that first order accurate solutions are too diffusive and lead to excessive growth of shear layers while solving a viscous flow field and this is the main reason why a second or higher accurate methods are used. To do so, it is necessary to compute the left and right state of the flow variable which is done via gradient reconstruction. This is by definition an interpolation of the flow variables inside each control volume. The gradient is then used to construct the viscous fluxes and for the turbulence sources as discussed in Chaps. 3.6.3 and 3.5 respectively.

There are generally two widely techniques used, namely the Green-Gauss and the least squares method [53]. The first one is preferred over the second because the least squares gradient reconstruction can be quite inaccurate for highly stretch cell such those used in viscous

computation. For each control volume Ω_i, the Green-Gauss theorem allows to express a volume integral into a surface integral:

$$\int_{\Omega_i} \nabla W_i \, d\Omega_i = \oint_{\partial \Omega_I} W_i \, dS_i \tag{3.36}$$

This is then discretised on the control volume constructed using the FVM:

$$\nabla W_i = \sum_{i=0}^{N_f - 1} \frac{1}{2\Omega_i} \left(W_i - W_j\right) n_{ij} \tag{3.37}$$

3.6.5 Thin Shear Layer Approximation

For external bodies, attached or mildly separated high Reynolds number flow, it is possible to talk of a thin boundary layer. The TSL (Thin Shear Layer) approximation was first introduced by Steger stated [95] in order to save computational time by considering only the viscous stresses normal to the wall and neglecting the others because are small in comparison.

However, the main reason why this approximation was used was not to reduce the computational time, but to maintain consistency w.r.t. the flow-adjoint linearisation which was performed considering the TSL approximation.

3.6.6 Multigrid

There are many methods to accelerate the flow solution convergence. Two of the most used are by far preconditioning and multigrid [96, 97]. These two strategies do not mutually exclude each other. In this work, a LU-SGS with a LU preconditioning is employed along with a multigrid strategy. In particular, there exist two kinds of multigrid strategy, namely geometric and algebraic.

The geometric multigrid strategy achieves low frequency error damping by working directly on the grids, whereas the algebraic multigrid does it by reducing the size of the implicit operator. Therefore, the former is governed essentially by the grid, whereas the latter is governed by the

physics. The geometric multigrid is the strategy chosen for all the simulations presented in this work because it is known to perform better for non-linear problems and it has the advantages of being independent from the equations being solved.

The geometric multigrid strategy consists essentially of three steps: restriction, prolongation and smoothing step. The first one describes the action to transfer information from finer to coarser grid, whereas the vice versa is true for the second step. The third one describes the action of damping the small residual oscillations which are the primary target of the multigrid strategy. This is needed because, the low frequency residual errors are hardly damped by the flow solver. These low frequency oscillations can be well resolved on a coarser mesh because the wavelength, as compared to the grid space, is smaller. From here, the necessity to transfer back and forth the solution between different grids.

3.7 Summary

It was shown how the governing equations and the solution methods were chosen in relation to the type of flow-field being solved, i.e. transonic flow. This consists of a mix of subsonic and supersonic regions which is challenging to solve due to the presence of shocks and the disparity between different local Mach numbers. Shocks, in particular, are considered important because of their interaction with the boundary layer where modelling viscous phenomena using the Navier-Stokes set of equations is of paramount importance.

In this chapter the flow governing equations were introduced along with their spatial and temporal discretisation techniques were briefly discussed. It was discussed how the multigrid strategy helps accelerate the convergence and how the disparity between different local Mach numbers requires to use a preconditioning technique.

The one-equation SA turbulence model was described and the choice was justified based on a compromise between computational time, differentiation difficulty and flow Jacobian conditioning.

Chapter 4 Volume Mesh Deformations

4.1 Introduction

In Chap. 2, it was highlighted how the discrete adjoint framework cannot avoid the differentiation of the mesh movement. In order to do so, it is first necessary to choose a volume mesh deformation technique which is the focus of this chapter.

To address this research question, a discussion is first provided to explain why mesh movement is used instead of mesh regeneration. This is then followed by a brief review of the existing mesh movement methods where the pros and cons of each method are briefly analysed. Subsequently, the DGM is introduced and a reasoning why it was chosen as the main investigation tool is given. The DGM [8] uses an explicit mapping technique, and as the name suggests, it is based on the Delaunay map. The original method and a modification are also discussed.

In order to highlight the advantages of the DGM, another implicit mesh movements, namely LE is presented. The comparison is performed in order to highlight both computation time and memory saving offered by the DGM.

The chapter concludes with a summary of the most important findings.

4.2 Structured versus Unstructured Mesh

There are two types of meshes available: structured and unstructured. All the other types are an hybridisation of these two. A mesh is called structured when each volume cell has the same number of neighbouring nodes and the connectivity can be retrieved using a two or three ordered indices. On the other hand, if any of the two above mentioned requirements are not satisfied the mesh is then defined unstructured.

The chief reason why researchers are putting a lot of effort to make unstructured mesh codes is that, as noted by Newman *et al.* [98] triangles (2D) and tetrahedra (3D) are the simplest geometric elements known to posses area and volume respectively, capable of resolving an irregular shape. Since triangles and tetrahedra can be generated automatically in a fairly easy

way, this explains the reason why structured grids lag behind in terms of flexibility and automation. However, this makes the N_{SM} and N_{VM} for unstructured grids much larger when compared to structured grids. This is briefly shown in Tab. 4.1 for the W1 wing [99] depicted in Fig. 4.1.

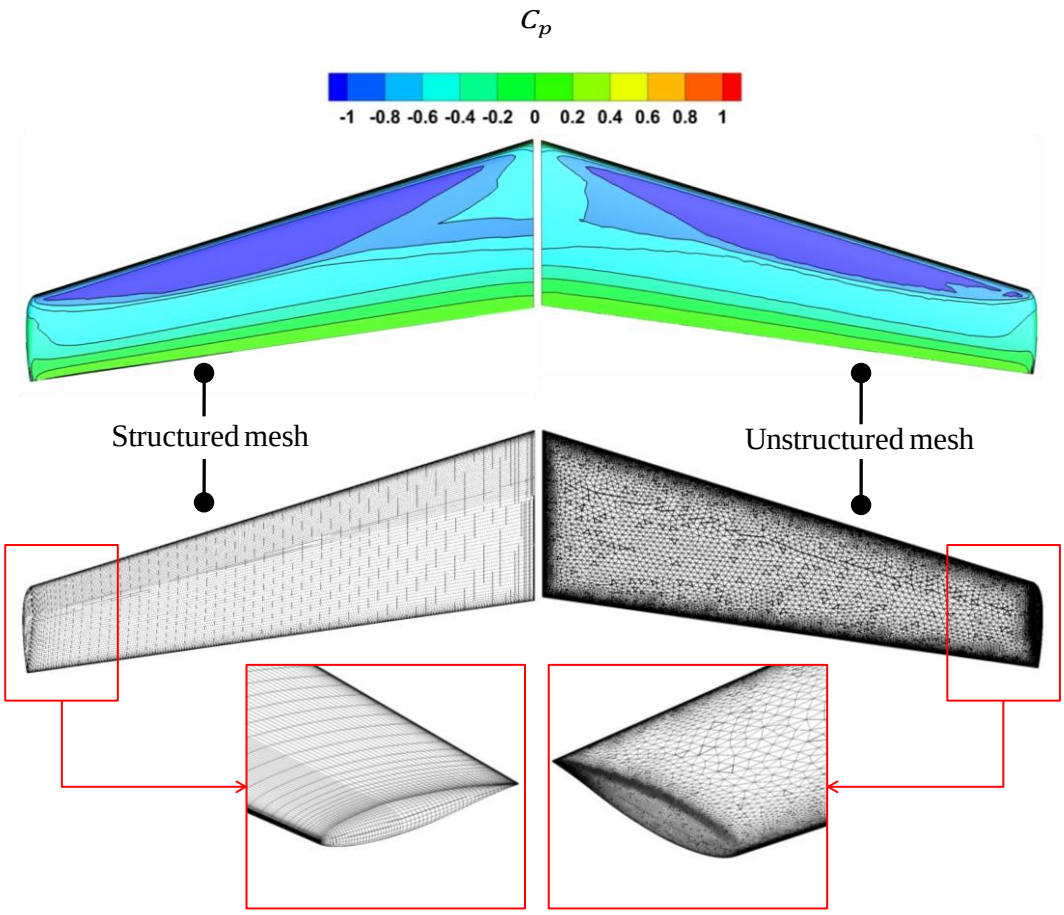


Figure 4.1 Comparison between two mesh types for the W1 wing.

Although unstructured solver are encumbered with additional expense both in memory and computing time, this has to be balanced with the capability to provide a solution for complex geometries [96, 89]. Referring to Tab. 4.1, it is clear that even for a very simple wing, structured and unstructured mesh register differences of one order of magnitude in terms of both N_{SM} and N_{VM}. These, as discussed in Chap. 2.4.1, ultimately decide the size of the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$.

Furthermore, the great disparity both in the N_{SM} and N_{VM} is also reflected in the scatter of the coefficient of drag data (see Tab. 4.1) computed at Mach number equal to 0.76 and fixed lift coefficient of 0.5.

Table 4.1 Comparison between structured and unstructured mesh.

Geometry	**Mesh type**	N_{VM} **(Mil)**	N_{SM} **(Mil)**	C_D
W1	Unstructured	3.71	0.31	**0.006**5278
	Structured	0.61	0.023	**0.006**0443

4.3 Mesh Deformation versus Mesh Regeneration

In aerodynamic optimisation an improvement of the aerodynamic surface is generally sought. At each iteration, there is first a change in the surface mesh generally governed by the parameterisation tool, followed by a change in the volume mesh which can be done either by regeneration or deformation. The discussion presented in this chapter concentrates on the second option for which there are many methods available as briefly depicted in Fig. 4.2. To justify this choice, different metrics such as order of magnitude of the deformations, consistency, differentiability, and computation time need to be analysed one by one.

Generally, if the mesh deformation strategy does not significantly impact the flow features and aerodynamic coefficients accuracy, mesh re-generation is not the favoured option. This is corroborated by the trend seen in many aerodynamic optimisation applications published in the literature, such as in Nielsen and Park [37], Mader *et al.* [25] and Mavriplis [39]. In fact, mesh deformation is still computationally much cheaper when compared to mesh re-generation. This is very important for small and repetitive changes commonly found in a gradient-based optimisation exercises. However, it is also recognised that for larger deformations, where the change in the near-wall orthogonality is significantly affected, mesh generation would become the preferred option.

Maintaining the same grid connectivity is of paramount importance to maintain consistency between each design optimisation iteration [100, 101]. On the other hand, mesh regeneration, especially for unstructured grids, changes the point discretisation which in turn inevitably leads to different truncation errors. On the other hand, for structured mesh regeneration methods, such as those based on TFI (Trans Finite Interpolation), the connectivity is not necessarily changed. In fact, if the TFI parameters stay constant, the grid topology is not changed and consistency is therefore maintained.

Resorting to unstructured grid re-generation adds considerable complexity from the linearisation point of view [102, 103]. In fact, the regeneration process contains many routines that are either non differentiable or too complex to differentiate. Although restricted to structured grids only, some examples of mesh routines differentiation have been published in the literature, as discussed in Chap. 2.5.5.

Lastly, unstructured and structured mesh generations are relatively more time consuming and require considerable manual adjustments when compared to mesh movements. Aerodynamic optimisation requires repetitive small adjustments of the lifting surface, thus employing a fast mesh movement that is able to retain the mesh connectivity is of vital importance.

4.4 Volume Mesh Deformation Methods

In order to introduce the DGM, a classification of the mesh movements is first proposed as depicted in Fig. 4.2. There are many models available and wide disparities in robustness and efficiency exist. From the surveyed literature review, it has emerged that there are many ways to classify different mesh movement techniques. The classification presented here is such that there are cases where a single approach may belong to more than one category. Volume mesh updates can be classified in three main groups:

1. Classification based on the concept
2. Classification based on the raw data
3. Classification based on the type of equations

The classification based on the concept the mesh movements emulate comprises of physical analogy and elliptic smoothing methods and are based on the connectivity. One of the most difficult aspect of these methods is that the resulting equations need to be discretised and the resulting system of equations is not easily solved.

The second type of classification describes on what the methods really work on, meaning if they work directly on a point-based structure (i.e. cloud of points) where only the Cartesian coordinates are necessary or if they are based on the edge connectivity. Examples of the latter are spring [104], LE [105] analogy, elliptic smoothing [106] and quaternion [101]. Connectivity becomes very important for unstructured mesh movement methods since the ratio between the number of edges over the volume mesh nodes can vary significantly. Examples of point-based scheme are RBF [107], DGM [8] and IDW [108]. One of the advantage of points-based methods is that they can be easily parallelised, whereas for those based on the connectivity the parallelisation is more involved due to the index order, pre-conditioning and multigrid strategy adopted as noted by Mavriplis and Yang [109].

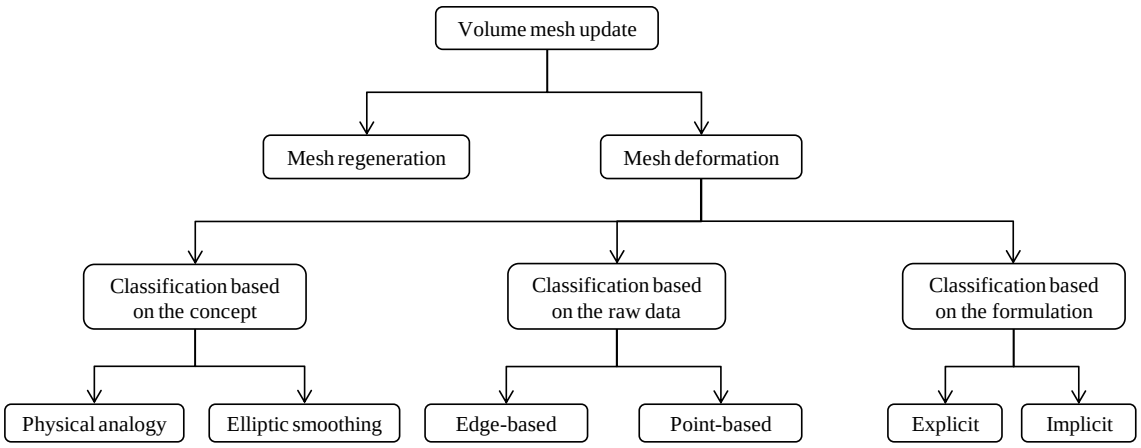


Figure 4.2 Volume mesh deformation methods classification.

Finally, the third one reflects how the equations are casted from a mathematical point of view, meaning whether the volume mesh node updates are explicitly or implicitly defined. In other words (see also Chap. 2.5.3), explicit methods are defined as those where the volume mesh node updates are provided without having to rely on an iterative procedure, otherwise are defined as implicit methods. This factor is the one that ultimately determines the cost of the method in

terms of memory or CPU time requirements. In fact, explicit methods require much less computation power because they avoid the solution of a large and (generally) stiff linear system of equations. On the down side, they are not very robust when it comes to large deformation. Another point is the ability of implicit methods, contrary to explicit methods, to locally adapt the mesh without affecting the entire volume mesh.

Mesh movement based on the elliptic equations was extensively used in the past and an example can be found in Helenbrook [106]. The most used elliptic equation is the Laplacian, which unfortunately is able to produce only small deformation although a modification was proposed by Lohner and Yang [110] to improve its robustness. Blom [111] compared this method against the spring analogy and concluded that the latter seems to be more robust.

A much more sophisticated mesh movement was proposed by Samareh [101] based on the quaternion algebra. The method is still based on mesh connectivity, but offers a superb mesh orthogonality for both small and large deformations.

TFI [112] can also be used as an algebraic explicit mesh movement and has found many applications for structured meshes. It is based on the linear interpolation of the boundaries and internal nodes along each structured mesh lines. Its biggest advantage is its speed, whereas its disadvantage is that mesh orthogonality is not guaranteed.

Batina [104] developed a method where the connectivity of the volume mesh nodes is modelled through a series of springs representing the stiffness of the mesh. This was then improved by Farhat *et al.* [113] who added a torsional stiffness to the formulation. Subsequently, the method was further improved by Murayama *et al.* [114] by adding a correction based on the relation between the springs and the angles between the facets. In terms of relative performance Nambu *et al.* [102] compared it against the LE and Liu *et al.* [8] against the DGM.

Witteveen and Hestre [108] proposed another algebraic explicit method based on the IDW (Inverse Distance Weighting) interpolation. The method was tested against the RBF where it was shown to have a large saving in computation cost and a comparable deformed mesh quality. On the down side, like for all the explicit algebraic methods, the constant parameters used need to be chosen and are subjected to fine tuning for best performance. Another downside is that the method scales linearly with the mesh dimension and as the mesh is increased the method

becomes expensive, but still less expensive than RBF. A reduction in the cost was achieved with a modified formulation proposed by Luke *et al.* [115].

Hybrid methods are defined as those that combine two different strategies together. One example of this blending was proposed by Lefrançois [116] where the LE was coupled with a fast interpolation procedure that improves the computation time. Something very similar was done by Kenway *et al.* [117] where the authors addressed the high computation cost of LE by coupling the method with an algebraic method. Another example is proposed by Wang *et al.* [118] where the authors addressed the DGM orthogonality issue by combining it with the RBF method.

4.5 Mesh Movement Based on the Delaunay Graph Method (DGM)

This chapter describes the most important features of the DGM [8]. The method is first introduced by describing the principle it is based upon followed by a step-by-step description of how it works. Furthermore, a modification that addresses some of the DGM robustness issues is described.

4.5.1 The Delaunay Map

To introduce how the Delaunay triangulation in 2D and tetrahedralisation in 3D are computed, it is first necessary to justify why the Delaunay method was chosen as the main investigation tool. Decomposing surface and volume objects into smaller polygons, a process known as object discretisation, has always been the focus of graphic computational geometry. For instance, a surface can be approximated by decomposing it into smaller and simpler planar polygons. This can be done by inserting new points or by creating the map using the given cloud of points. The latter is the option discussed here.

The simplest known map is a triangulation in 2D and a tetrahedralisation in 3D. The discussion is restricted to these two options. There are many methods available in the literature. Žalik and Lamot [119] suggested that these can be broadly classified into three groups (see Fig. 4.3, adapted from Ref. [119]): algorithms based on the diagonal insertion, on the Steiner points

insertion and on the Delaunay criterion. It is not the intention of the author to give a full account of these methods, rather to give an impression of the different available strategies. However, some important points need to be pointed out. The main differences between these three methods lay on the number of points considered and on the algorithm used. A first general comparison is depicted in Fig. 4.3.

In order to justify why the Delaunay map was chosen as the main investigation tool, these techniques are compared against the number of elements generated and the quality of the resulting map as reported in Tab. 4.2. Furthermore, a comparison of the differences between these three strategies in terms of shared and non shared segments is depicted in Fig. 4.4.

Algorithms based on the diagonal insertion are those that, as the name suggests, given a set of points, the map, i.e. triangulation, is made by the recuserviley insertion of diagonals. Since its implementation does not have any routine put in place to check the quality of the elements, it is expected to provide a poor discretisation quality [119].

Algorithms based on Steiner points are those where additional elements are added to support a good quality element index. They are execpted to perform better than those based on either diagonal insertion or Delaunay criterion in terms of generated elements quality. However, the final number of points is greater than the original cloud of points [119].

The Delaunay criterion states that the circum-circle (-sphere) for each triangle (tetrahedron) should not include other points except those constructing the element. By using the empty circle criterion, the best map possible for the given initial points distribution can be obtained [119].

Before embarking in a detailed derivation of the DGM mesh movement, it is the author's opinion that it would be beneficial to anticipate some of its requirements. Firstly, the given cloud of points must be part of the map, or in other words no extra points should be added and secondly a good quality of the triangulation/tetrahedralisation should be attained. These two requirements express the need to have the best possible elements quality withouth adding any points other than those provided. Therefore, these requirements are not far from those mentioned for a pure cloud of points discretisation. In fact, the initial points are indeed the domain boundary points of interest and that the quality of the elements is important in terms of maximum allowed mesh movement and mesh sensitivity requirements.

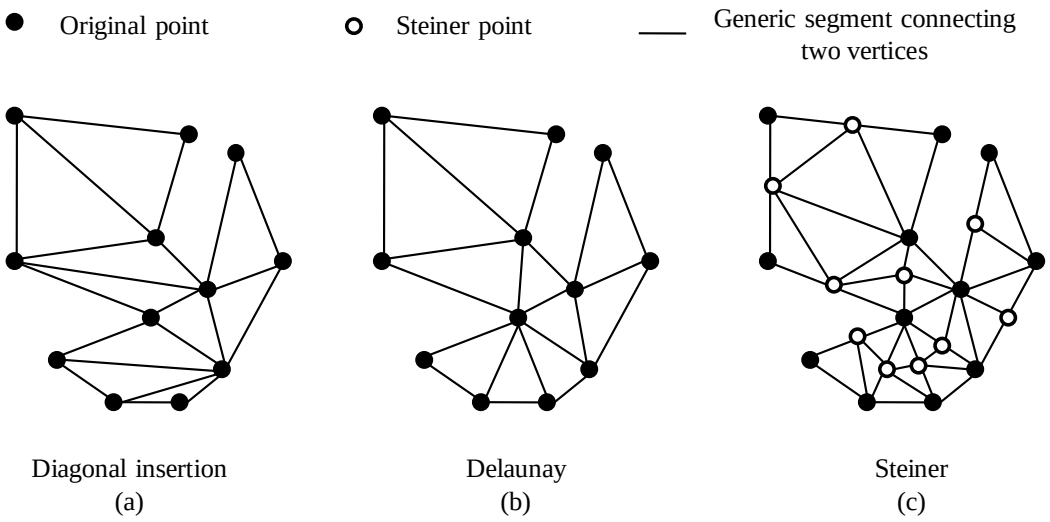


Figure 4.3 Conceptual representation of different triangulation strategies.

Table 4.2 Comparison between the geometry discretisation features and the DGM-based mesh movement requirements.

Methods	**Geometry discretisation features**		**Mesh movement requirements**	
	No. of points	**Element quality**	**No. of points**	**Element quality**
Diagonal insertion (see Figs. 4.3 (a) and 4.4 (a, c))	Equal to the original cloud of points	Generally not good	Fulfilled	Not fulfilled
Delaunay (see Figs. 4.3 (b) and 4.4 (b, c))		The best if no extra points are added	Fulfilled	Fulfilled
Steiner point (see Figs. 4.3 (c) and 4.4 (c, d))	Greater than the original cloud of points	Very good	Not fulfilled	Fulfilled

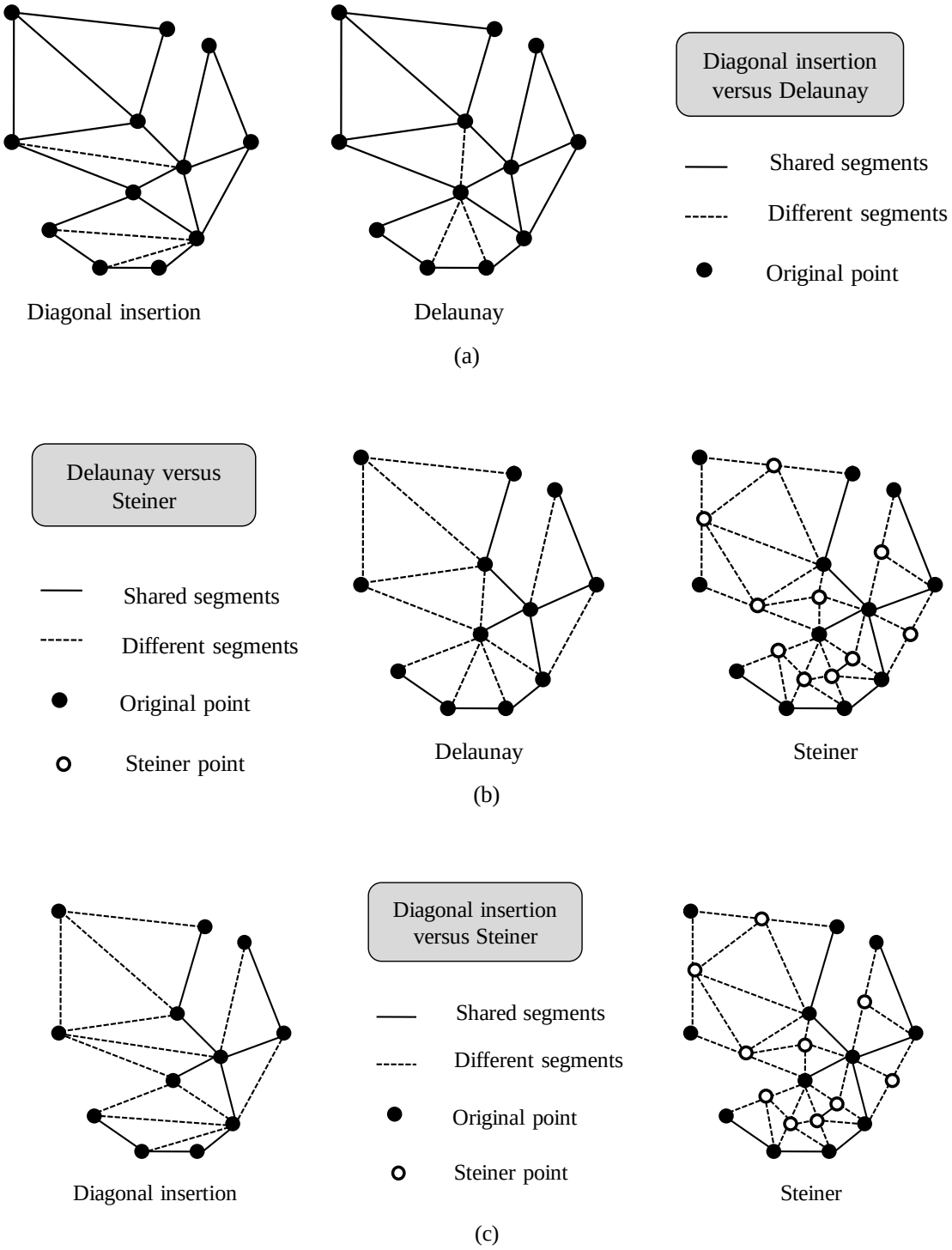


Figure 4.4 Visualisation of the differences between the Diagonal insertion, the Delaunay and the Steiner triangulation.

It is then concluded that the Delaunay method offers the best option available as briefly shown in Tab. 4.2. The implementation used in this work is based on the method published by Barber *et al.* [120] and implemented in the Qhull library.

The Delaunay's principle is also used in various mesh generation techniques, but this topic is out of scope and the reader is referred to Mavriplis [121] for an exhaustive review on this matter.

4.5.2 Mapping the Mesh via Delaunay Mapping

The computational domain is generally made up by solid, symmetry and farfield boundary nodes. The choice of which boundary types to include in the Delaunay map is defined a priori and influences the resulting deformed mesh. A few requirements must be respected in this process. Firstly, all the points at the solid boundaries are effectively Delaunay tetrahedron vertices because it is desired to have the sensitivity w.r.t. the entire solid wall, a concept that will be made clearer in Chap. 5.3. Secondly, at least one Delaunay tetrahedron vertex is generally chosen to be a farfield node with the exclusion of those laying on the symmetry plane. If the points at the symmetry plane are included, they would then become Dirichlet's boundary conditions. This is not recommended because the symmetry plane points behave effectively as the other volume mesh points and as such, depending on the surface mesh deformations, they need to be displaced accordingly. This would not be possible if they are treated as a Dirichlet's boundary conditions. Furthermore, additional points can be added in between as shown in Fig. 4.5 where the ultimate goal is to reduce the skewness of the Delaunay elements.

As it was discussed in Chap. 4.5.1, the Delaunay criterion allows to state that there exists a subdivision of a given cloud of points (solid wall, farfield and additional extra points) that maximises the smallest angle. Let us assume that the Delaunay map has been computed, since each mesh point is inside one of the Delaunay tetrahedra, a one-to-one map between boundaries and volume (internal) mesh nodes is constructed. To this regards, in Fig. 4.5 (close up views on the right) the area within the solid wall is effectively triangulated, but these Delaunay elements are not considered because no volume mesh point is present there.

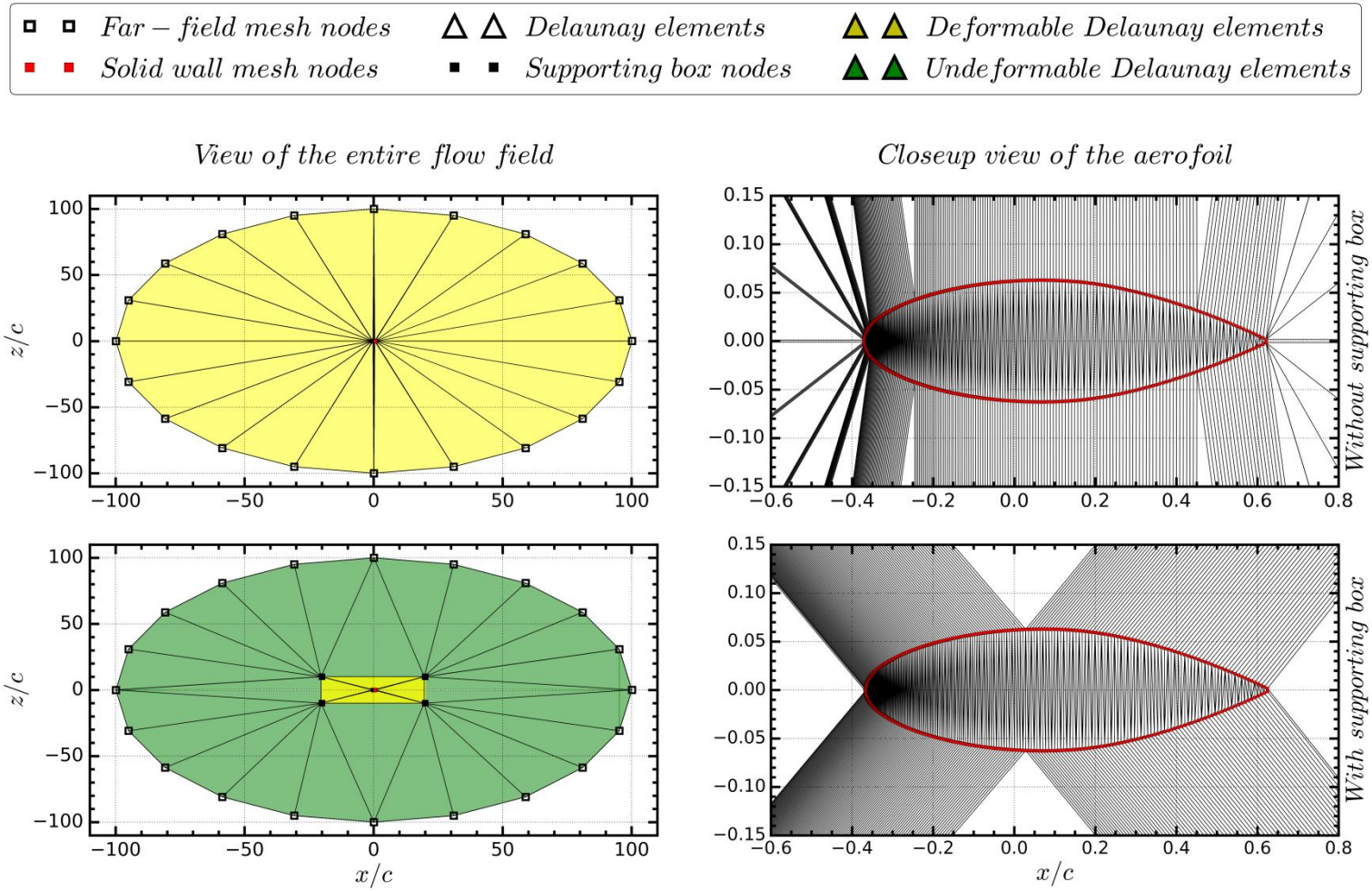


Figure 4.5 Two different strategies to build the Delaunay map.

Referring to Fig. 4.6, points $D_{1,2,3,4}$ are the Delaunay tetrahedron vertices. Point D_4 is assumed to be a farfield boundary mesh node, whereas the others are surface mesh nodes. Point P, is an arbitrary volume mesh node placed inside the Delaunay tetrahedron, whereas point P' is its new spatial position after a deformation at the wall has taken place. This is shown displacing point D_2, depicted in Fig. 4.6 (a), and maintaining fixed points $D_{1,3,4}$. Point D_2 in its new position, i.e. D_2', can be seen in Fig. 4.6 (c) and in Fig. 4.7 where the Delaunay element is overlapped on the actual mesh.

The volume ratio coefficients are computed after Liu *et al*. [8] as:

$$e_i = \frac{v_i}{V}\,, \qquad for\ i = 1,2,3,4 \tag{4.1}$$

where v_i are the sub-volumes depicted in light orange colour in Fig. 4.6 (b) and V is the Delaunay element volume depicted in light blue colour in the same figure. Therefore, by geometric construction it holds:

$$\sum_{i=1}^{4} e_i = \sum_{i=1}^{4} \frac{v_i}{V} = \frac{v_1}{V} + \frac{v_2}{V} + \frac{v_3}{V} + \frac{v_4}{V} = 1 \tag{4.2}$$

The spatial position of each mesh point is then defined as:

$$\begin{aligned} x_p &= \sum_{i=1}^{4} e_i x_{D_i} \\ y_p &= \sum_{i=1}^{4} e_i y_{D_i} \\ z_p &= \sum_{i=1}^{4} e_i z_{D_i} \end{aligned} \tag{4.3}$$

where x_{D_i}, y_{D_i}, z_{D_i} are chosen to be at the computational domain boundary and are part of the Delaunay graph. After a deformation has taken place, the new point Cartesian coordinates are computed using the constant volume ratio coefficients, e_i . Therefore, the new spatial coordinates can be found as:

$$x_p^{'} = \sum_{i=1}^{4} e_i x_{D_i}^{'}$$
$$y_p^{'} = \sum_{i=1}^{4} e_i y_{D_i}^{'} \tag{4.4}$$
$$z_p^{'} = \sum_{i=1}^{4} e_i z_{D_i}^{'}$$

In an optimisation loop, the $x_{D_i}^{'}$, $y_{D_i}^{'}$, $z_{D_i}^{'}$ updates are provided by the surface parameterisation tool chosen, and since the volume coefficient ratios stay constant it is possible to find the new volume mesh spatial coordinates $x_p^{'}$, $y_p^{'}$, $z_p^{'}$ without performing any additional calculation. Using the matrix and vector notation this can be written as:

$$\{\boldsymbol{X}\} = [\boldsymbol{E}]\{\boldsymbol{B}\} \tag{4.5}$$

where matrix $[\boldsymbol{E}]$ has dimensions equal to $[N_{VM} \times N_{SM}]$ containing all the volume ratio coefficients and $\{\boldsymbol{B}\}$ is the vector containing the Delaunay boundary vertices. This generally comprises of different elements such as $\{\boldsymbol{B}\} = \{\boldsymbol{B}_{ff}, \boldsymbol{B}_{sw}\}$ where $\{\boldsymbol{B}_{sw}\}$ represents the Delaunay vertices that coincide with the solid wall mesh nodes, therefore $\{\boldsymbol{B}_{sw}\} \equiv \{\boldsymbol{S}\}$, and $\{\boldsymbol{B}_{ff}\}$ represents the Delaunay vertices that coincide with the farfield mesh nodes.

To summarise, the following pseudo-code is provided to guide the construction of matrix $[\boldsymbol{E}]$:

1. Read-in the surface boundaries
2. Perform the Delaunay mapping as described in Chap. 4.5.1
3. Store on memory to which Delaunay element each volume mesh belongs to
4. For each internal volume node, $X_i \in \{\boldsymbol{X}\}$ with $1 \leq i \leq N_{VM}$:
   - 4.1 Retrieve the link between Delaunay element and volume mesh found at point 3
   - 4.2 Construct the four sub-volumes by connecting X_i with the Delaunay element vertices found at point 4.1
   - 4.3 Compute volume and sub-volumes associated to X_i
   - 4.4 Compute the four sub-volume ratios as per Eq. (4.1) using the values computed at point 4.3

4.5 Store for each X_i the four coefficients computed at point 4.4

4.6 Repeat the steps from 4.1 to 4.5 till $i = N_{VM}$

While executing the pseudo-code presented above there is one important point that needs to be addressed in more details. At step No. 2, the Delaunay map is performed on the provided cloud of points which means that a triangulation is effectively performed also inside the aerofoil as clearly shown in Fig. 4.5. However, this map is not directly used by the DGM method. In fact, at step No. 3, no volume mesh points are inside any of the Delaunay element triangulating the inner part of the aerofoil.

One of the disadvantages of this method is its inability to control the mesh orthogonality at the wall which is due to the initial Delaunay elements quality. By adding some auxiliary boundary points, Xiao *et al.* [122] improved the initial Delaunay map and the orthogonality at the wall.

In terms of applications, the method was applied by McDaniel and Morton [123] to the movement of a control surface and by Durrani *et al.* [124] to a flapping wing. The method was compared by Liu *et al.* [8] against the spring analogy and by Sun *et al.* [125] against the barycentric coordinates. Finally, the method was applied by Li *et al.* [126] for a complex aircraft optimisation using a non–gradient optimisation framework.

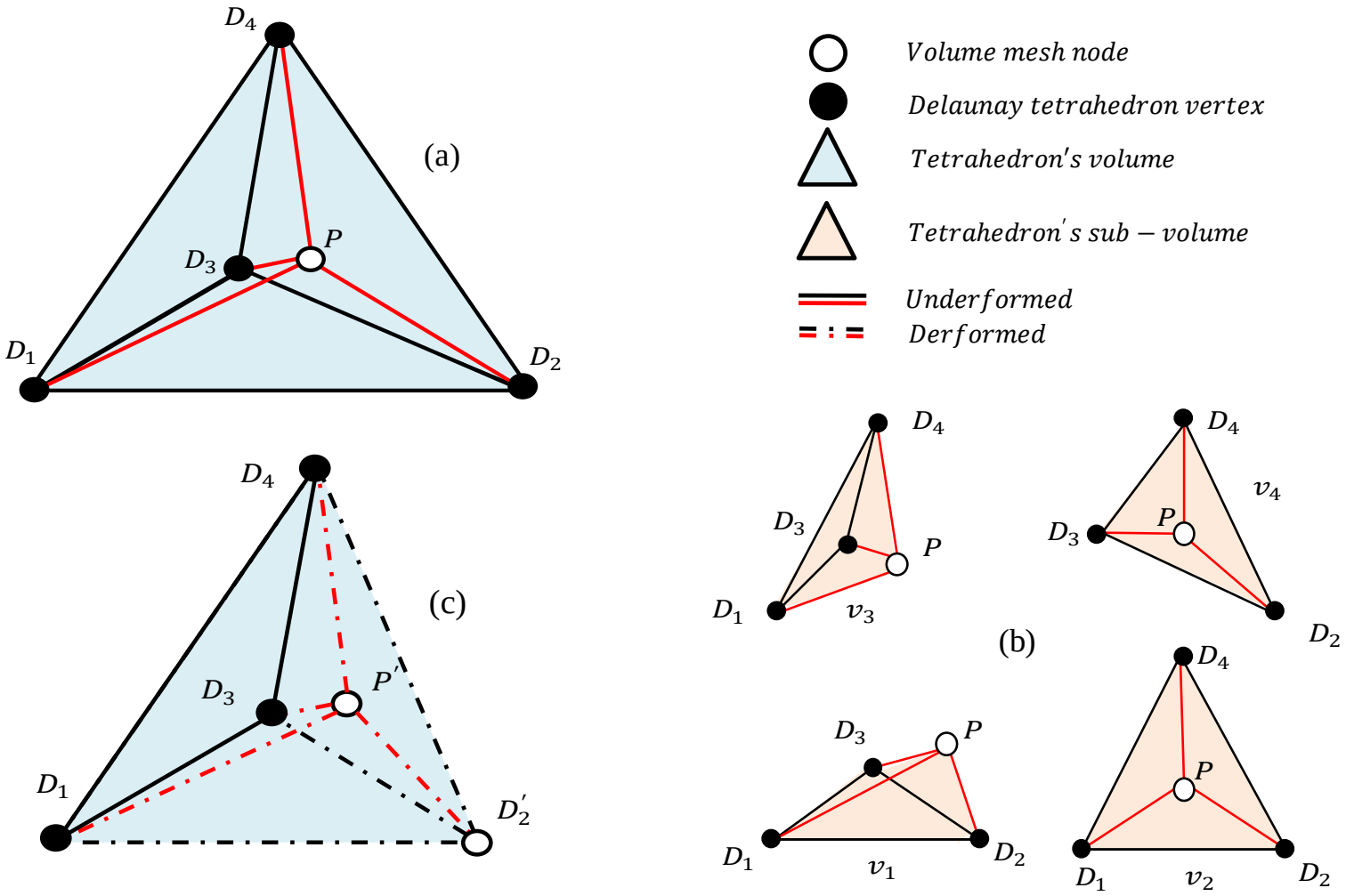


Figure 4.6 Construction of a Delaunay element (a) along with its sub-volumes (b) and a deformation of the Delaunay map (c).

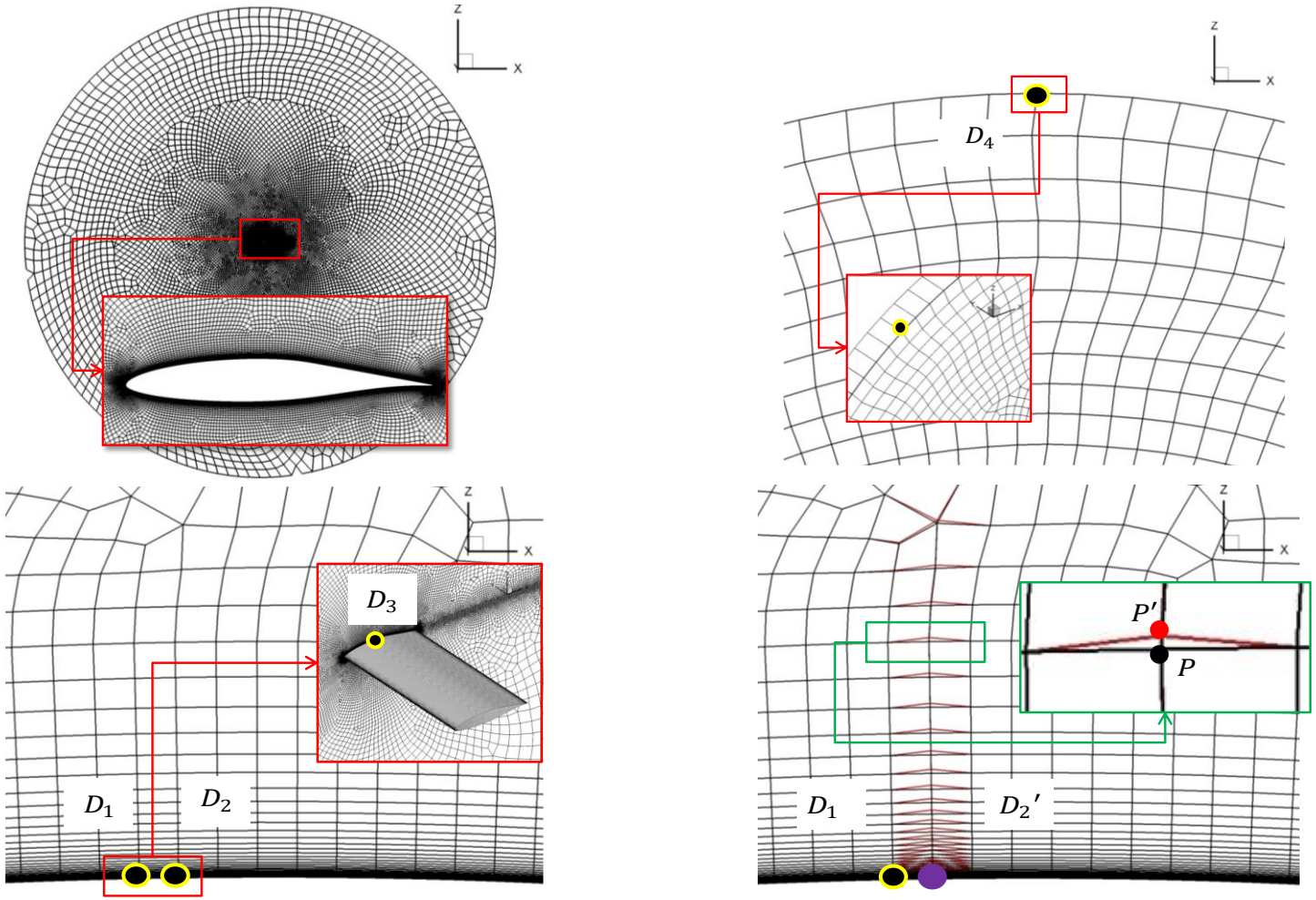


Figure 4.7 Construction of a Delaunay element overlapped to the computational mesh.

4.5.3 Construction of the Supporting Box

There are two main challenges in the implementation of the DGM-based mesh movement, namely the distribution and the quality of the Delaunay map which have an impact on the quality of the deformed mesh elements. These requirements may seem to be the same, but there are cases where even a Delaunay map with a good quality index does not provide the appropriate distribution of the Delaunay vertices from a grid sensitivity stand point of view. These issues are addressed and a solution discussed.

The Delaunay map based on the Qhull library works very well for simple geometries, such as any symmetric profile, but when it comes to complex aero shapes the triangulation/tetrahedralisation can be challenging and in some cases can reduce the amount of feasible deformations. An example of such a situation is depicted in Fig. 4.8 (left) where it is evident that the quality of the Delaunay map is not optimal close to the trailing edge. In order to improve the quality of some highly skewed Delaunay elements care should be taken to ensure that the map is performed only between the surface and the volume mesh nodes and not solely from wall-to-wall.

The novel idea proposed in this work is to construct a supporting box which is made from a series of extra points that closely follow the shape. By doing so Eq. (4.5) will not be changed in its fundamental formulation, however the Delaunay boundaries are augmented by the addition of the supporting box's points. Thus, $\{\boldsymbol{B}\} = \{\boldsymbol{B}_{ff}, \boldsymbol{B}_{sw}, \boldsymbol{B}_{sb}\}$, where $\{\boldsymbol{B}_{sb}\}$ contains the Delaunay vertices that coincide with the supporting box nodes. The supporting box points are treated like the far field points in that they are maintained fixed. This is true for all the test cases presented in this work. In order to make a distinction between the two formulations, whenever, the supporting box is used, the DGM will be referred to as mDGM where the *m* stands for modified. The effect, in terms of deformed volume mesh nodes (i.e. a reduced subset), can be seen in Figs. 4.5 and 4.17. This improves the quality of the Delaunay map as shown in Fig. 4.8 (right) and avoids the surface-to-surface map which in turn solves the distribution issue.

There are three requirements that must be considered in this process. The first requires the placement of the supporting box at a distance that allows a reasonable amount of deformations. The second requires consideration of all the volume mesh nodes where the gradient $\{dI/d\boldsymbol{X}\}$ is not null as described in more details in Chap. 5.7.4. Thirdly, if the supporting box is placed too

far away from the wall, this method no longer works because the corrected area/volume will be seen as convex hull again and therefore would be triangulated/tetrahedralised.

There are mainly four options available to construct or select the supporting box points, namely manual construction, using the normal to project each surface mesh point outward and selection of some suitable volume nodes for structured and unstructured mesh.

The first method available is to manually place the supporting box points where they are most needed. A process which is time consuming for complex shapes and can be done only a posteriori after a first triangulation/tetrahedralisation is performed, and the critical areas are indentified. An example is offered in Fig. 4.5.

The second option is to use the normal to the wall to project each surface mesh point outward. Although the procedure can be easily automated, there are two drawbacks with this technique. The first one is when sharp edges are considered, such as at the trailing edge where the normal is always ill-defined. The second issue is when complex cases are considered such as wing-to-fuselage intersections where the points, constructed using the normal to the fuselage, go over the points constructed using the normal to the wing. In this case, a complicated routine to collapse the points should be used. For these reasons, this technique is limited to simple 2D cases as depicted in Fig. 4.8.

The third and fourth options are to carefully select some suitable points from the mesh. Two techniques are presented here. The first one, depicted in Figs. 4.9 and 4.10, makes use of the buffer layer featured by hybrid meshes, whereas the second one, depicted in Fig. 4.11, applies only to structured meshes where one can "walk" away from the wall and obtain the supporting box at any distance. Regardless to the mesh types, the selected points follow the solid boundary shape offering a sort of "second skin". Some examples are depicted in Figs. 4.12, 4.13 and 4.14 for different type of meshes.

Tab. 4.3 reports how the N_{SB} (number of supporting box points) compares with the N_{SM}. The construction of the supporting box by manually placing extra points in strategically important areas is the only technique for which the $N_{SB} \neq N_{SM}$. It is clear that an optimum distribution is obtained when there is a one-to-one match, hence the $N_{SB} = N_{SM}$.

Table 4.3 Comparison of the methods available to construct the supporting box.

Method	Mesh type	Pros	Cons
Selection of the some suitable volume mesh points	Hybrid	Can handle complex 3D shapes $N_{SB} = N_{SM}$	Cannot control the distance from the wall unless regenerating the mesh
	Structured	Can control the distance from the wall without regenerating the mesh $N_{SB} = N_{SM}$	Cannot handle complex 3D shapes unless by carefully selecting the shared points between different mesh blocks
Manual construction	Hybrid and structured	Easy to implement	$N_{SB} \neq N_{SM}$ Difficult to implement for complex cases
Using the wall normal		$N_{SB} = N_{SM}$	Cannot handle complex 3D shapes due to the definition of the surface normal at sharp edges

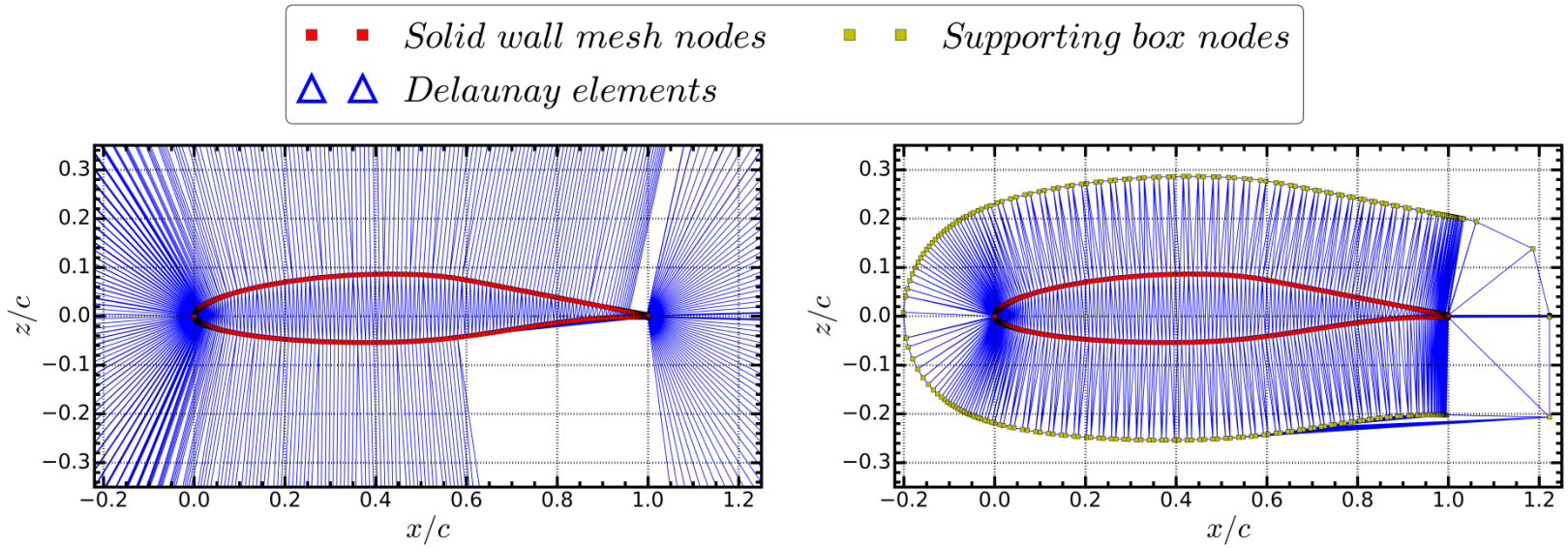


Figure 4.8 Triangulation of a the RAE 5243 profile without (left) and with the supporting box (right).

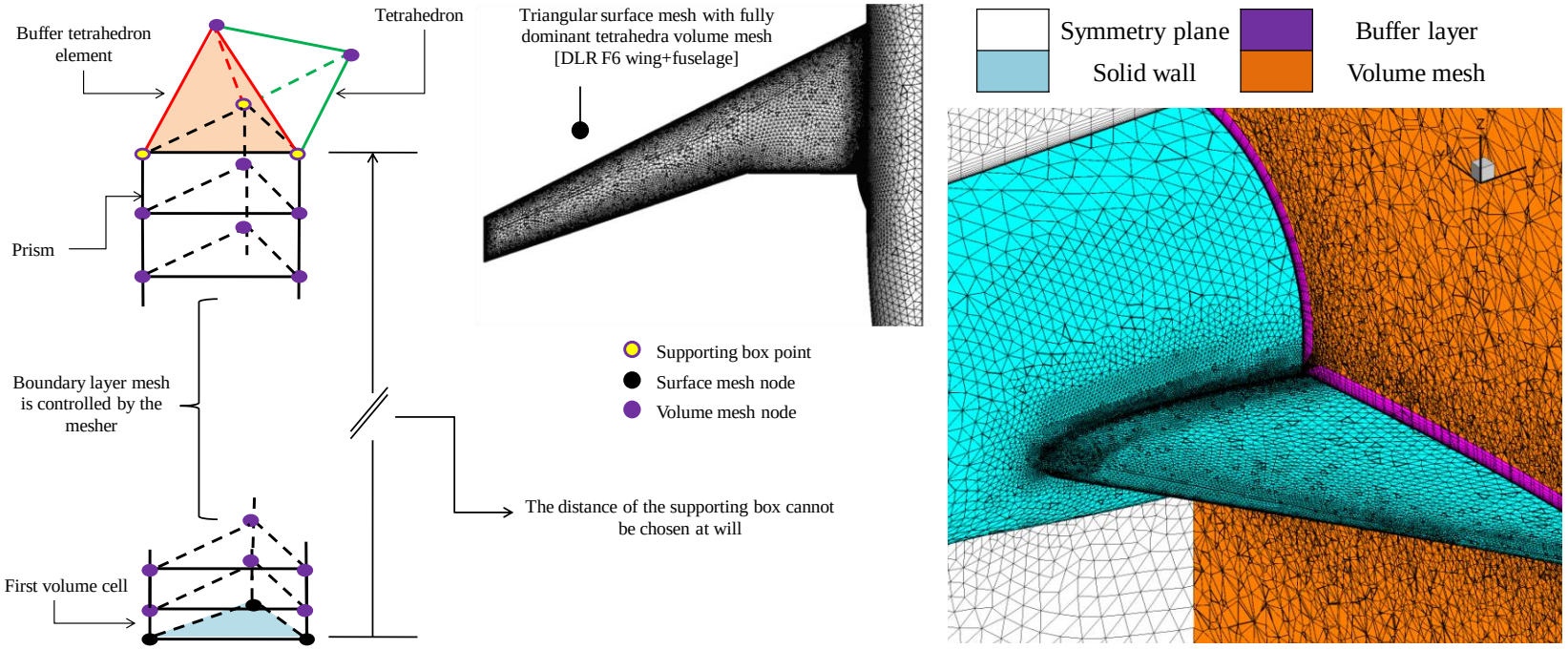


Figure 4.9 Representation of the procedure to select the supporting box nodes for a tri-dominant hybrid mesh.

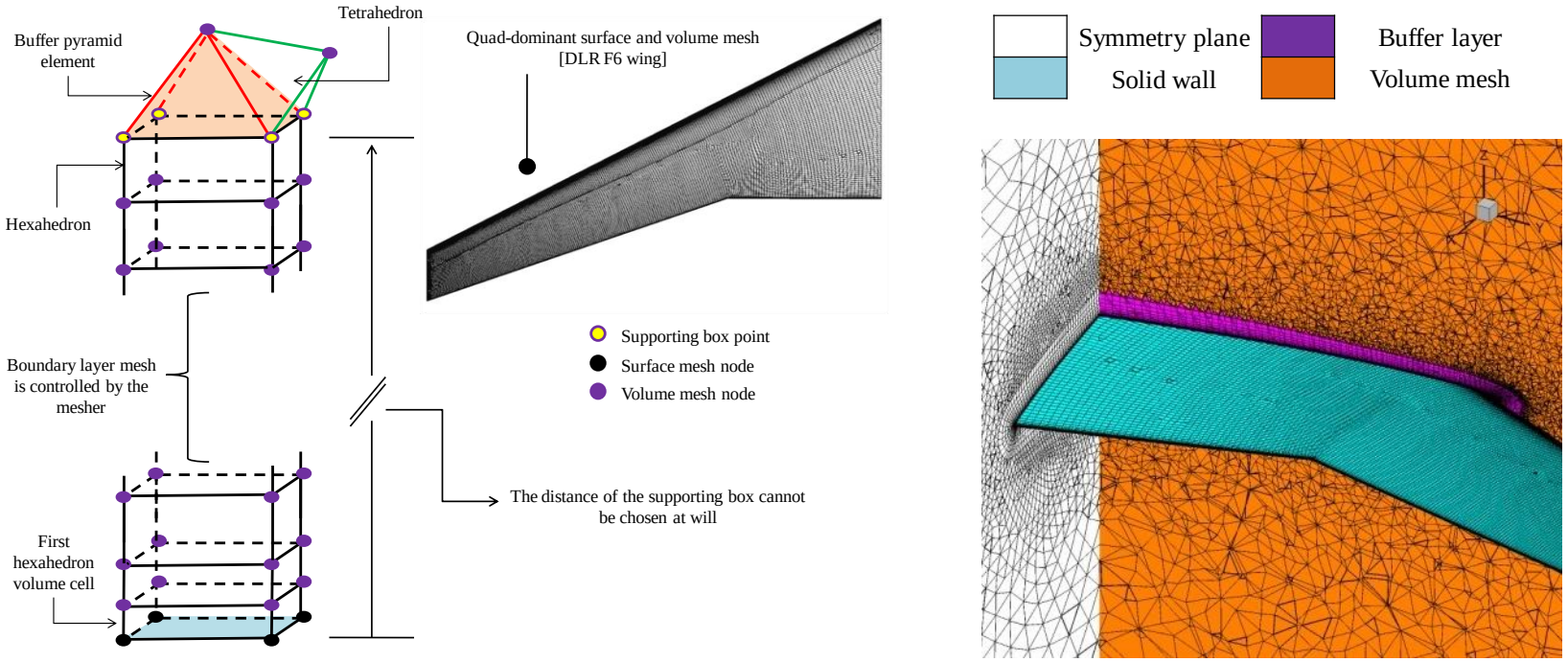


Figure 4.10 Representation of the procedure to select the supporting box nodes for a quad-dominant hybrid mesh.

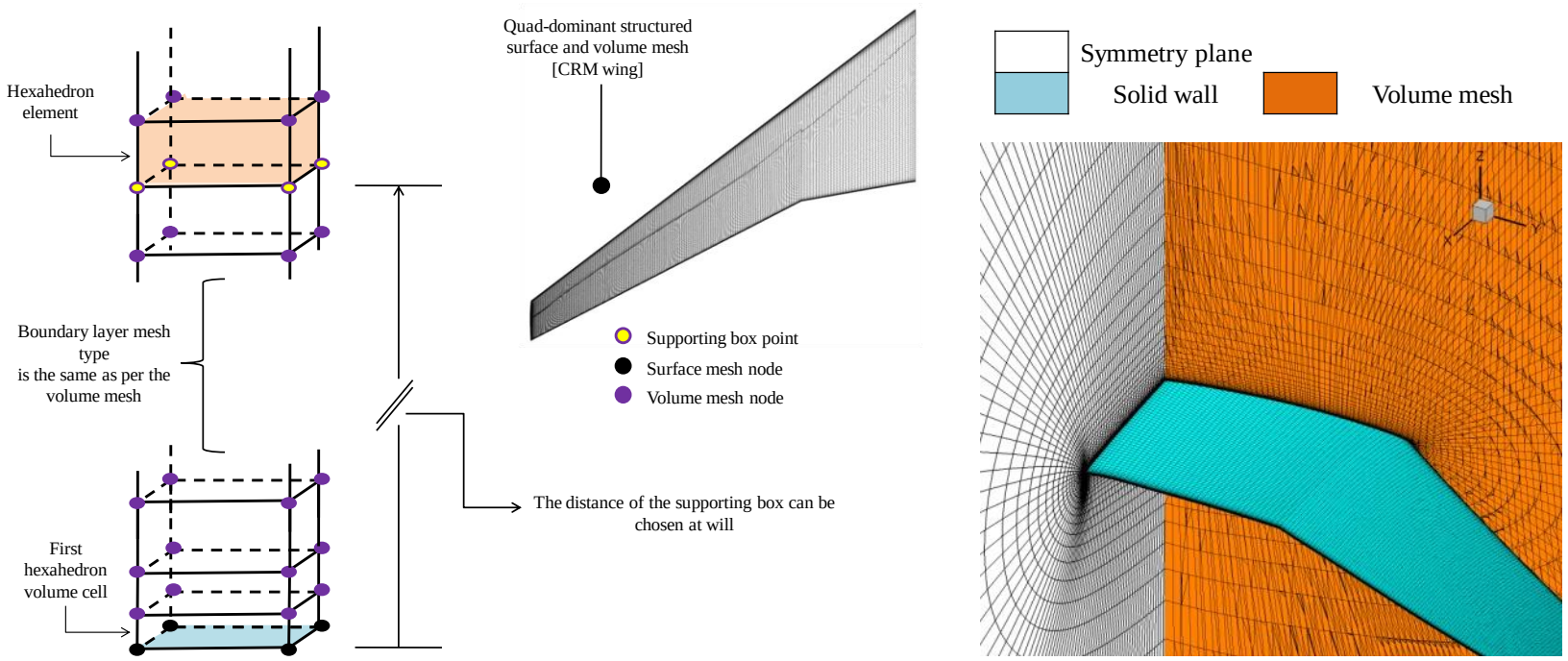


Figure 4.11 Representation of the procedure to select the supporting box nodes for a fully structured mesh.

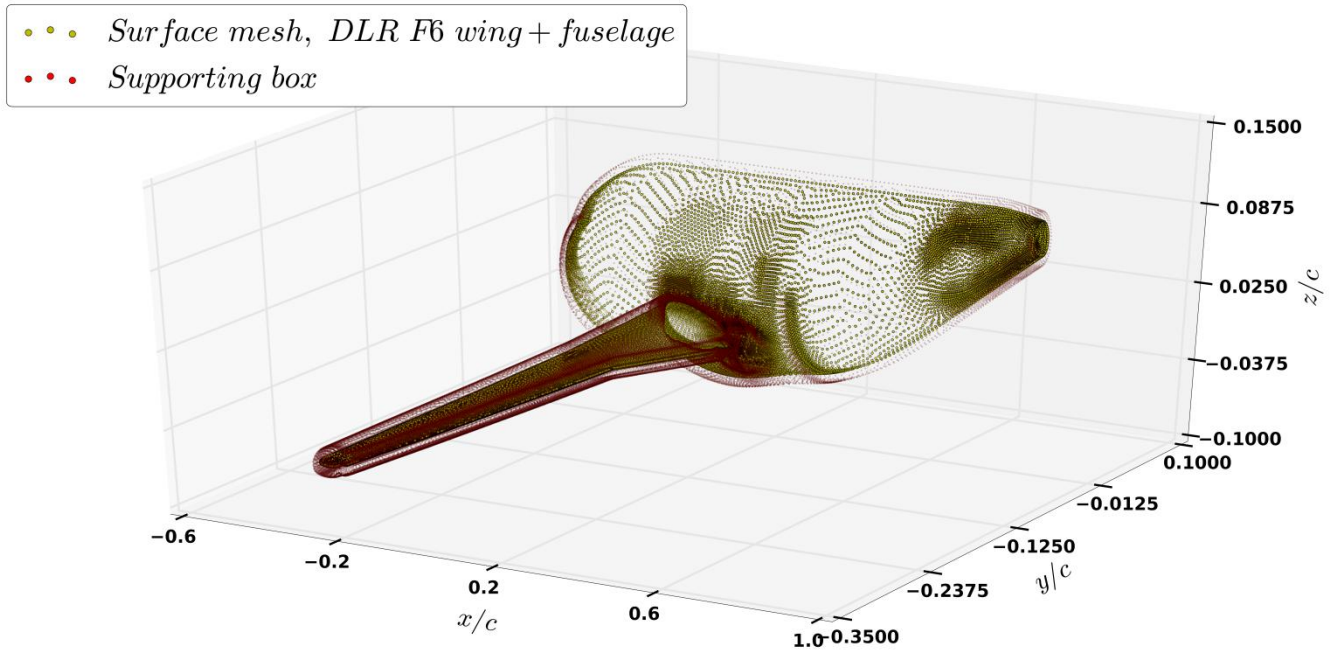


Figure 4.12 Isometric view of the supporting box for a tri-dominant hybrid mesh.

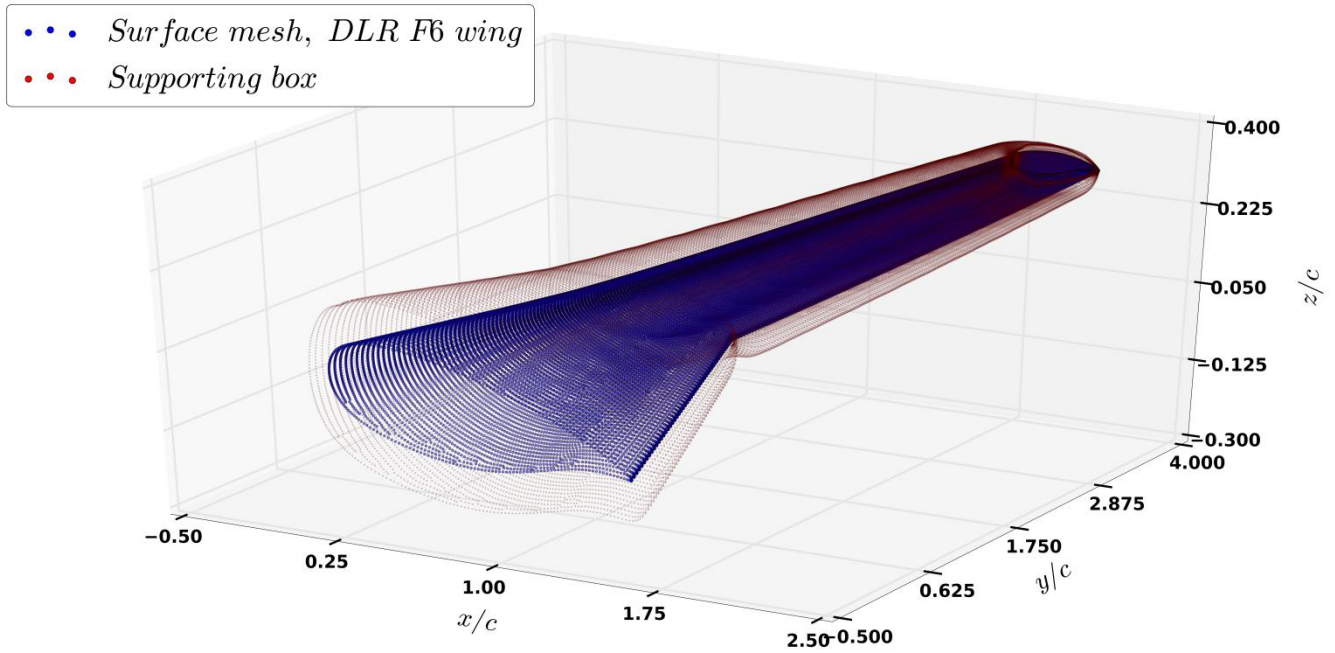


Figure 4.13 Isometric view of the supporting box for a quad-dominant hybrid mesh.

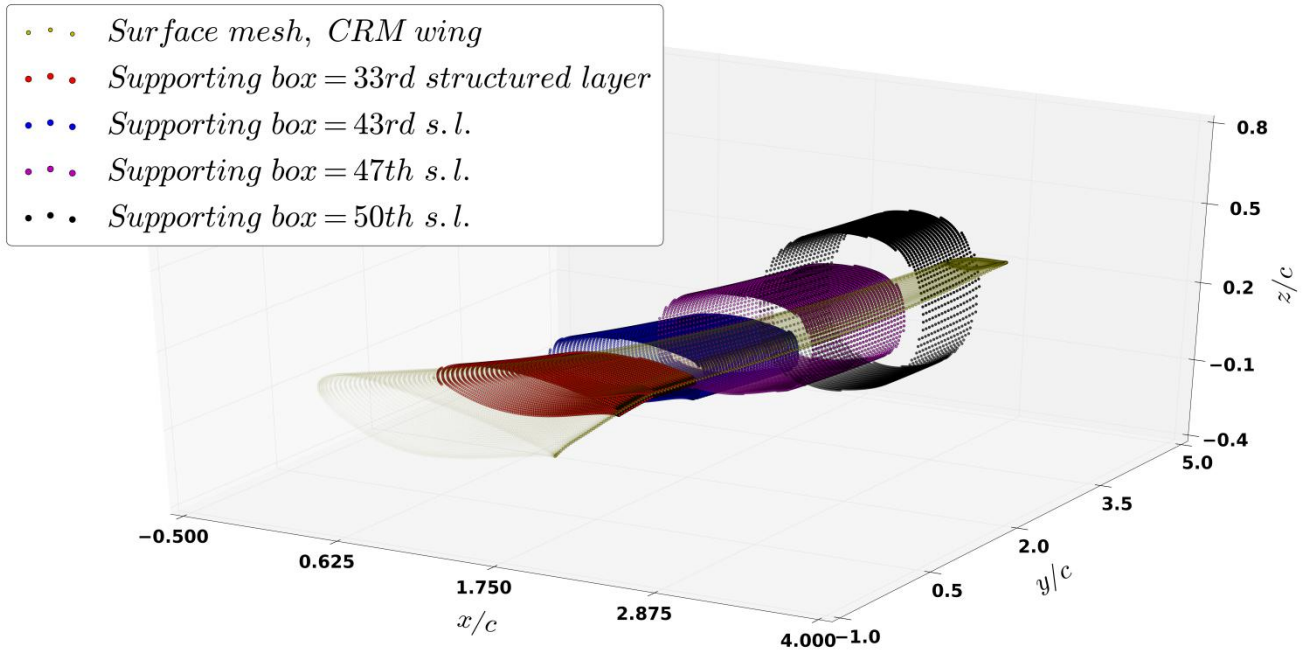


Figure 4.14 Isometric view of the supporting box for a fully structured mesh.

4.6 Mesh Movement Based on Linear Elasticity

One of the best known and widely used method to move the mesh is the LE analogy. This method emulates the elastic body using an edge-based structure where the volume mesh update is defined implicitly. In the LE mesh movement, the discretised computational domain obeys the LE equation which states that the divergence of the stress sensor, $\boldsymbol{\sigma}$, must be equal to the external applied forces, $\boldsymbol{F}$:

$$\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{F} \qquad \text{on } \Omega \tag{4.6}$$

where on the boundaries the general displacement vector is defined as $\{\boldsymbol{U}\}(=\{\boldsymbol{S}\})$ effectively imposing a Dirichlet's boundary condition. Therefore, the external force is not required. The relation between the stress tensor and the strain tensor, $\boldsymbol{\varepsilon}$, is as follows:

$$\boldsymbol{\sigma} = \lambda Tr(\boldsymbol{\varepsilon})\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon} \tag{4.7}$$

where Tr is the trace, $\boldsymbol{I}$ the identity tensor, whereas λ and μ are the Lamé constants defined respectively as:

$$\lambda = \frac{\nu E}{(1+\nu)+(1-2\nu)} \tag{4.8}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{4.9}$$

where the Young's modulus E and the Poisson's ratio ν generally take values equal to $E > 0$ and $-1 < \nu < 0.5$ respectively. In order to maintain the near wall mesh orthogonality as intact as possible, the elements close to the wall are stiffened in order to allow only a rigid body motion, whereas larger elements, which are placed away from the wall, are allowed to accommodate a greater part of the deformation. This is done by enforcing a different value of E over the computational domain. Stein *et al.* [105] proposed to use an elasticity modulus inversely proportional to each cell volume, Yang and Mavriplis [109] proposed to use a value inversely proportional to the distance from the wall, Stein *et al.* [105] corrected it with a stiffness proportional to the mesh deformation in an incremental fashion, and finally Yang and Mavriplis [127] used a distribution optimised via an adjoint technique. Another remarkable modification was proposed by Dwight [128] for large deformations, where the author used the

Lagrangian strain tensor instead of the commonly used liner kinematic law. This modification resulted in a more robust mesh movement which allows one to impose much larger mesh deformations well beyond the point where mesh regeneration become necessary. The one implemented in the DLR TAU-Code is based on an elasticity modulus inversely proportional to each cell volume.

Although the strain tensor can be related non-linearly to the gradient of the displacement vector, the version implemented in this work uses the linear version. This can be expressed as:

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T) \tag{4.10}$$

Furthermore, stress and strain tensors can be written in compact form as:

$$\{\boldsymbol{\sigma}\} = [\boldsymbol{K}]\{\boldsymbol{\varepsilon}\} \tag{4.11}$$

$$\{\boldsymbol{\varepsilon}\} = [\boldsymbol{A}]\{\boldsymbol{U}\} \tag{4.12}$$

where $[\boldsymbol{K}]$ is the elasticity matrix and $[\boldsymbol{A}]$ is the kinematic matrix. Stress, strain and displacement vectors can be explicitly written as:

$$\{\boldsymbol{\sigma}\} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{Bmatrix}; \qquad \{\boldsymbol{\varepsilon}\} = \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{31} \end{Bmatrix}; \qquad \{\boldsymbol{U}\} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{4.13}$$

where $\sigma_{11,22,33}$ and $\sigma_{12,23,31}$ are the normal and shear stress components respectively. On the other hand, $\varepsilon_{11,22,33}$ and $\varepsilon_{12,23,31}$ are the respective strains. Furthermore, the displacement at the wall is traditionally given the name $\{\boldsymbol{U}\}$, but to comply with the nomenclature used in this work, it follows that $\{\boldsymbol{U}\} = \{\boldsymbol{S}\}$. Thus:

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{Bmatrix} = \begin{bmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{31} \end{Bmatrix} \tag{4.14}$$

$$\begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{31} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{4.15}$$

Eqs. (4.11) and (4.12) are then combined together to yield:

$$[\boldsymbol{K}]\{\boldsymbol{X}\} = \{\boldsymbol{S}\} \tag{4.16}$$

It is clear that the volume mesh coordinates are defined implicitly. In fact, the elasticity matrix needs to be inverted. Eq. (4.16) is then cast in FEM form and discretised using a Galerkin method. The FEM shape function is considered to be linear on the grid elements because it is assumed that the discretisation of the computational domain is also appropriate to resolve the LE equations. Alternatively, one can use the FVM as shown by Biedron and Thomas [129]. The resulting linear system of equations is then solved using a restarted ILU preconditioned GMRES strategy.

Its main advantage is the wide range of applications and robustness that comes at a price of high computation cost. This last one is particular evident all the time the method is augmented with a stiffening procedure. Widhalm *et al.* [36] noted that the resulting stiffness matrix is sparse and stiff. It is sparse because for unstructured mesh the connectivity is not ordered and can reach high values. On the other hand, it is stiff because the viscous cells have a high-aspect ratio. This problem is generally addressed by breaking the full displacement to small subsequent steps, but this eventually increases the computation time. Barrala et *al.* [130] compared the method against the IDW and concluded that, although for several test cases the two methods were comparable, for thin viscous grids the LE appears to be better.

4.7 DGM-based Mesh Movement Performances Assessment

The assessment of the DGM-based mesh movement performance is done on four levels. The first one compares the DGM against the LE in terms of memory and CPU time, the second one

studies how the cost associated with the DGM scales with the N_{VM}, the third analyses the supporting box influence and finally the last one studies how the deformed grids differ when different mesh movement strategies are used.

In order to justify why the DGM was chosen as the main investigation tool a brief comparison against the LE is reported in Tab. 4.4. The LE was chosen because it has been widely recognised to be one of the most robust method available [109] and in addition, like the DGM, is able to provide the gradient of the objective function w.r.t. the entire surface mesh in its differentiated form as explained in Chap. 5.5.

As will be made clearer in Chap. 5.2, one of the goals of this work was to find an efficient differentiated mesh movement. To do so, the available mesh movements were surveyed and the DGM was chosen because it is based on an algebraic explicit method. This allows the method to be used without worrying either about the mesh type or the CPU time cost associated with the mesh size. These two features have implications both in the mesh movement and in the differentiated mesh movement. On the downside, it is however recognised that LE takes much better care of the near-wall orthogonality.

Table 4.4 Comparison between DGM and LE.

Method	**Pros**	**Cons**
DGM	Closed form formulation Low CPU time and memory consumption Ideal for repetitive movement Provides the gradient of the objective function w.r.t. the surface mesh when differentiated Can be easily parallelised	Non physical Pre-processing time is not negligible Mesh orthogonality is not preserved Global mesh movement (not local)
LE	Based on a physical analogy Provides good mesh quality Provides the gradient of the objective function w.r.t. the surface mesh when differentiated Local mesh movement (not global)	Large system of equations Stiff system of equations Not easy to parallelise

4.7.1 Memory and CPU Time Comparisons

Whenever two different methods are compared, it is very important to make an unbiased judgment by making the comparison as equal as possible. This is not a trivial task because the mDGM and LE deliver the deformed mesh in different steps and as such have different requirements. It is however recognised that a common ground must be found.

The methods are compared according to two metrics: RAM (Random Access Memory), shortly referred to as memory and CPU time for two different geometries depicted in Fig. 4.15. These are assessed on a single Intel Xeon 2.67GHz processor with 70.8Gb of RAM running on a Linux machine cluster.

The reason why the CPU time was considered in place of the wall-clock time, is because the latter is the sum of CPU, I/O (Input/Output) and communication delay time, which are not directly related to the methods being tested. Thus, the CPU offers a fair comparison of the real cost of the two methods. Of course, if one wants to improve the waiting time from the user point of view, a parallel implementation can be employed in order to achieve a better wall-clock time.

One step where the mDGM and LE differ is the pre-processing of the mesh, or in other words, the computation of matrix $[\boldsymbol{E}]$ and $[\boldsymbol{K}]$ respectively. In order to make a fair comparison it is necessary to break the cost between mesh pre-processing and mesh movement. Two geometries (see Fig. 4.15) were chosen and from the data reported in Tab. 4.5 three conclusions can be reached regarding the pre-processing CPU time and memory required to deform the mesh.

In terms of CPU time required to pre-process the meshes reported in Tab. 4.5, it is obvious that the mDGM requires more time than LE. This extra CPU time required by the mDGM can be explained by the fact that the method needs the construction of matrix $[\boldsymbol{E}]$ which (see Chap. 4.5.2) consists of three different steps: computation of the Delaunay graph, location of the volume mesh nodes and computation of the relative volume coefficients. As reported in Tab. 4.5, for the ONERA M6 wing, this amounts to 598s where 4s of which are needed to compute the Delaunay map and the rest accounts for the other two necessary steps. On the other hand, the LE consists of just building matrix $[\boldsymbol{K}]$ which roughly takes only 121s, and represents a notable saving in CPU time as compared to the mDGM. A similar trend is also registered for the DLR F6 wing for the two mesh movements tested.

On the other hand, the situation is reversed when the mesh movement CPU time is a analysed. It is obvious that LE requires much more CPU time than the mDGM. For the DLR F6 wing, the mDGM requires only 85s as compared to 2,376s required by the LE which equates to an improvement of two orders of magnitude. The same conclusion can be made while assessing the mesh movement memory requirement. In fact, there is again a saving as large as two orders of magnitude when using the mDGM over the LE.

The data reported in Tab. 4.5 are in line with what has been reported in the literature [123]. In fact, explicit algebraic methods require much more pre-processing than implicit methods. This is the price one has to pay for the reduction in CPU time offered.

The results presented in terms of pre-processing performance, could lead some readers to wrongly conclude that the mDGM is not that faster than the LE. To understand the reason, the following argument is given. In Chap. 7.4.7, it will be discussed how important is in an optimisation loop to use mesh deformation instead of mesh regeneration. If it is so, matrix $[\boldsymbol{K}]$ for the LE and matrix $[\boldsymbol{E}]$ for the mDGM do not need to be recomputed at each iteration. If these two matrices stay constant, apart from the initial pre-processing advantages of the LE over the mDGM, what really makes the difference is the CPU time required to effectively deform the mesh at each iteration. If there are more than one iteration, it is obvious that, having a fast mesh deformation outweighs the disadvantage registered in the initial pre-processing CPU time.

In the argument discussed above it was mentioned the necessity to look at the broader picture, i.e. the cost in an optimisation loop. However, as a matter of fact, the CPU time needed to solve the differentiated mesh movement plays a big role and for this reason needs to be considered as well. This matter is discussed again in Chaps. 5.8.2 and 7.4.7.

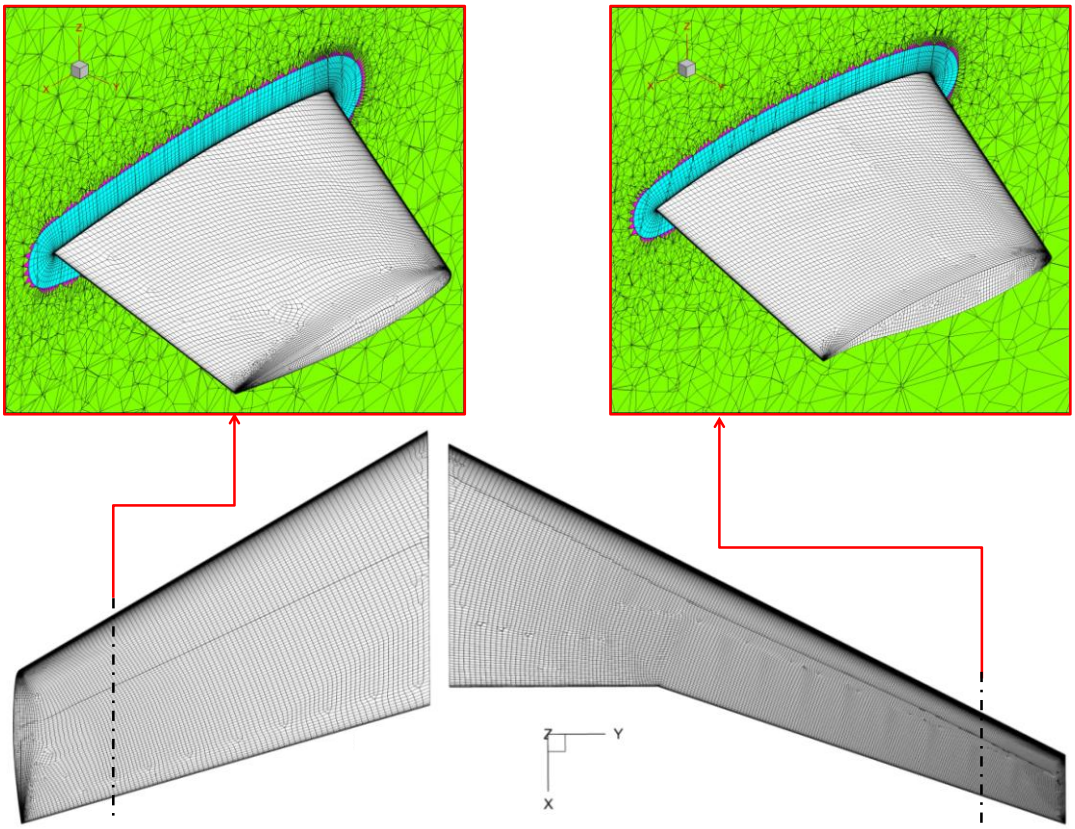


Figure 4.15 Overview of the ONERA M6 (left) and DLR F6 wing (right) mesh.

Table 4.5 CPU time and memory comparison between mDGM and LE.

Method	**Geometry**	N_{VM} **(Mil)**	**Memory (Gb) for the mesh movement**	**CPU time (s)**	
				Mesh pre-processing	**Mesh movement**
mDGM	M6 wing	1.6	0.23	598	50
	F6 wing	2.7	0.38	974	85
LE	M6 wing	1.6	14	121	1,807
	F6 wing	2.7	22	213	2,376

4.7.2 Performance Scaling with Mesh Size

This section studies how the mDGM computation expense scales with the N_{VM}. To this end, the DLR F6 wing-body with the FX2B bump fairing [131] available in three different mesh sizes as depicted in Fig. 4.16, is chosen for the analysis. From the results reported in Tab. 4.6, the conclusion is that, every time the mesh size is increased by a given delta value, the variation in memory, mesh pre-processing and deformation CPU time do not show the same increase.

To be more precise, the CPU time needed to pre-process the mesh shows an increase greater than those reported for the mesh size, but both CPU time needed to deform the mesh and memory show an increase lower than the increase in the N_{VM}. These data confirm the large advantage in terms of CPU time to deform the mesh over the less advantageous mesh pre-processing CPU time.

Table 4.6 mDGM CPU time and memory requirements for different mesh sizes.

N_{VM} **(Mil)**	Δ%	**Memory (Gb) for the mesh movement**	Δ%	**CPU time (s)**			
				Mesh pre-processing	Δ%	**Mesh movement**	Δ%
2.9	Datum	0.347	Datum	1,035	Datum	96	Datum
6.1	+110.34	0.515	+48.41	2,249	+117.29	188	+95.83
10.0	+244.82	0.804	+131.70	4,083	+294.49	318	+231.25

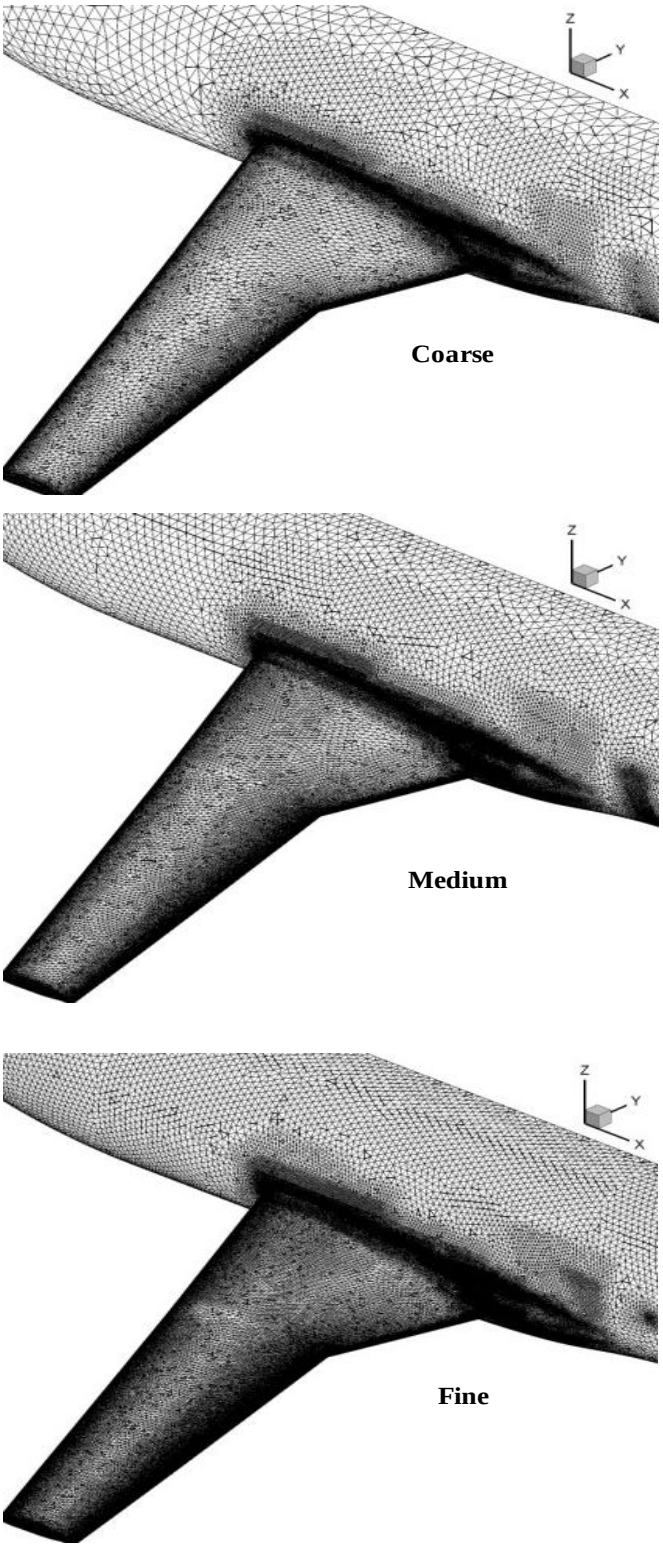


Figure 4.16 Close-up view of the DLR F6 wing-body for different mesh sizes.

4.7.3 Influence of the Supporting Box

In order to assess the impact of the supporting box in terms of CPU time while pre-processing the mesh, Tabs. 4.7 and 4.8 compare both oDGM (as in original) and mDGM performances on the DLR F6 wing body with the FX2B bump fairing configuration [131]. To do so, the time needed to pre-process the mesh is split into Delaunay mapping and volume ratio coefficients computation.

Four comments can be made on the data reported in Tabs. 4.7 and 4.8. The first one is on the increase in boundary points to be mapped, the second is on the CPU time to find the supporting box points, the third and fourth are on the CPU time required to compute the actual Delaunay map and the volume ratio coefficients respectively.

The idea of the supporting box is to create an offset copy of the surface mesh nodes at a given distance from the wall. Since the farfield points are generally one order of magnitude lower in number compared to the N_{SM}, the increase of points fed to the Delaunay triangulation/tetrahedralisation algorithm strongly depends on the N_{SM}. This is also corroborated by the fact that, if the buffer layer technique is used as described in Chap. 4.5.3, the $N_{SM} = N_{SB}$. As a result, the increase in the boundary points is generally between 80% and 90% w.r.t. the original boundaries.

However, the increase in CPU time to compute both the Delaunay map and the volume ratio coefficients is not comparable with the increase in N_{VM}. In fact, for the Delaunay map, the maximum registered increase was for the fine mesh at a value of 20.75% compared to an increase in tetrahedralised points of 88.98%. The same conclusion can also be applied for the CPU time required to compute the volume ratio coefficients, where the increase is far less than the increase in boundary points.

Table 4.7 Increase in the Delaunay boundary points for the mDGM.

Mesh movement	N_{VM} **(Mil)**	**Boundaries**		
		Solid wall + farfield points	**Supporting box points**	Δ%
oDGM	2.9	59,865+15,484	None	Datum
	6.1	111,852+18,084		
	10	166,299+20,585		
mDGM	2.9	59,865+15,484	59,865	+79.45
	6.1	111,852+18,084	111,852	+86.08
	10	166,299+ 20,585	166,299	+88.98

Table 4.8 DGM CPU time scaling with the supporting box.

Mesh movement	N_{VM} **(Mil)**	**CPU time (s)**				
		Supporting box points	**Delaunay map**	Δ%	**Computation of the volume ratio coefficients**	Δ%
oDGM	2.9	None	26	Datum	1,035	Datum
	6.1		44		2,249	
	10		53		4,083	
mDGM	2.9	57	29	+11.53	1,048	+1.25
	6.1	111	48	+9.09	2,348	+4.40
	10	178	64	+20.75	5,094	+24.76

4.7.4 Mesh Deformations Comparison

The last point to cover is the influence of the supporting box on the deformed mesh. This is done by comparing the two versions of the DGM against the LE. In order to give an idea of how different these methods are, Fig. 4.17 depicts various deformed meshes for an arbitrary surface mesh update.

While using the oDGM, it can be clearly seen that the deformed volume mesh closely follows the Delaunay map embedded in the computational domain (constructed using all the solid wall and the farfield points). On the other hand, whenever the mDGM is used, the Delaunay map is performed from the wall to the supporting box and then from this last one to the farfield. However, when the surface mesh is updated, the volume ratios that govern the volume mesh deformation have an influence that spans only within the area from the solid wall up to the supporting box as clearly depicted conceptually in Fig. 4.5. The advantage is that, for small deformations, the quality of the deformed mesh is improved because the skewness of the Delaunay elements (see Fig. 4.8) are reduced to the minimum possible thanks to the mirrored one-to-one map between the surface mesh and the supporting box points. However, if the supporting box nodes are not allowed to move, for some large deformations the boundary layer mesh quality could deteriorate because the deformation is not spread on the entire volume mesh. Although, this was not investigated in this thesis, a possible solution to this would be to allow the supporting box nodes to move. In this case, one would have to decide the relationship governing the displacement between the surface mesh and the supporting box nodes.

For what regards the LE, it is interesting to see how the area affected is much bigger compared to the DGM-deformed mesh. This can be explained by fact that the DGM is by definition based on an anisotropic operator, whereas the LE is based on an isotropic operator.

Furthermore, it is important to make another important point about this comparison. The way the supporting box modifies the deformed mesh is, to some extent, equal to the process of changing the mesh element stiffness in the LE. In fact, all the time the original stiffness is modified, the direct result is a modification of the process of distributing the deformation over the computational domain. By doing so, although with different laws, the LE is effectively imposing the same computational domain subdivision between deformed and non-deformed mesh depicted in Fig. 4.5.

In order to investigate more the quality of the deformed grids shown in Fig. 4.17, two widely used unstructured grid metrics are used, namely the mean mesh quality and the dihedral angle. For instance, for a 3D tetrahedron mesh element, the mean mesh quality is defined as:

$$q_i = 12 \cdot sign\,(V_i) \cdot \frac{\sqrt[3]{(3V_i)^2}}{\sum_{j=1}^{6} l_{ij}^2} \tag{4.17}$$

where V_i is the grid element cell and l_{ij} is the length of the edge connecting two tetrahedron vertices. The general guideline is to have a value of q_i as close as possible to the unity value. On the other hand, the more intuitive dihedral angle is defined as:

$$\theta_i = min_j\,(\Phi_{ij}\,, 180° - \Phi_{ij}\,) \tag{4.18}$$

where Φ_{ij} is the angle formed by two tetrahedron edges. The general guideline for this metric is to avoid values lower than five. Based on these two definitions, Tab. 4.9 lists the values obtained from the three deformations shown in Fig. 4.17.

Table 4.9 Mesh quality metrics.

Mesh movement	**Mean mesh quality**			**Dihedral angle**	
	q_i	q_i^{min}	q_i^{max}	θ_i^{worst}	No. $\theta_i < 5°$
Baseline	0.797843	0.0158565	0.999173	2.07613	25
oDGM	0.797843	0.0158565	0.999173	0.52355	41
mDGM	0.797843	0.0158565	0.999173	1.83613	28
LE	0.797843	0.0158565	0.999173	0.52355	41

The original mesh has a fairly good mean mesh quality of ~0.79 with only 25 out of 2.5 Mil. elements (note that for an unstructured mesh the No. of elements is different from the N_{VM}) with a dihedral angle lower than 5 degrees.

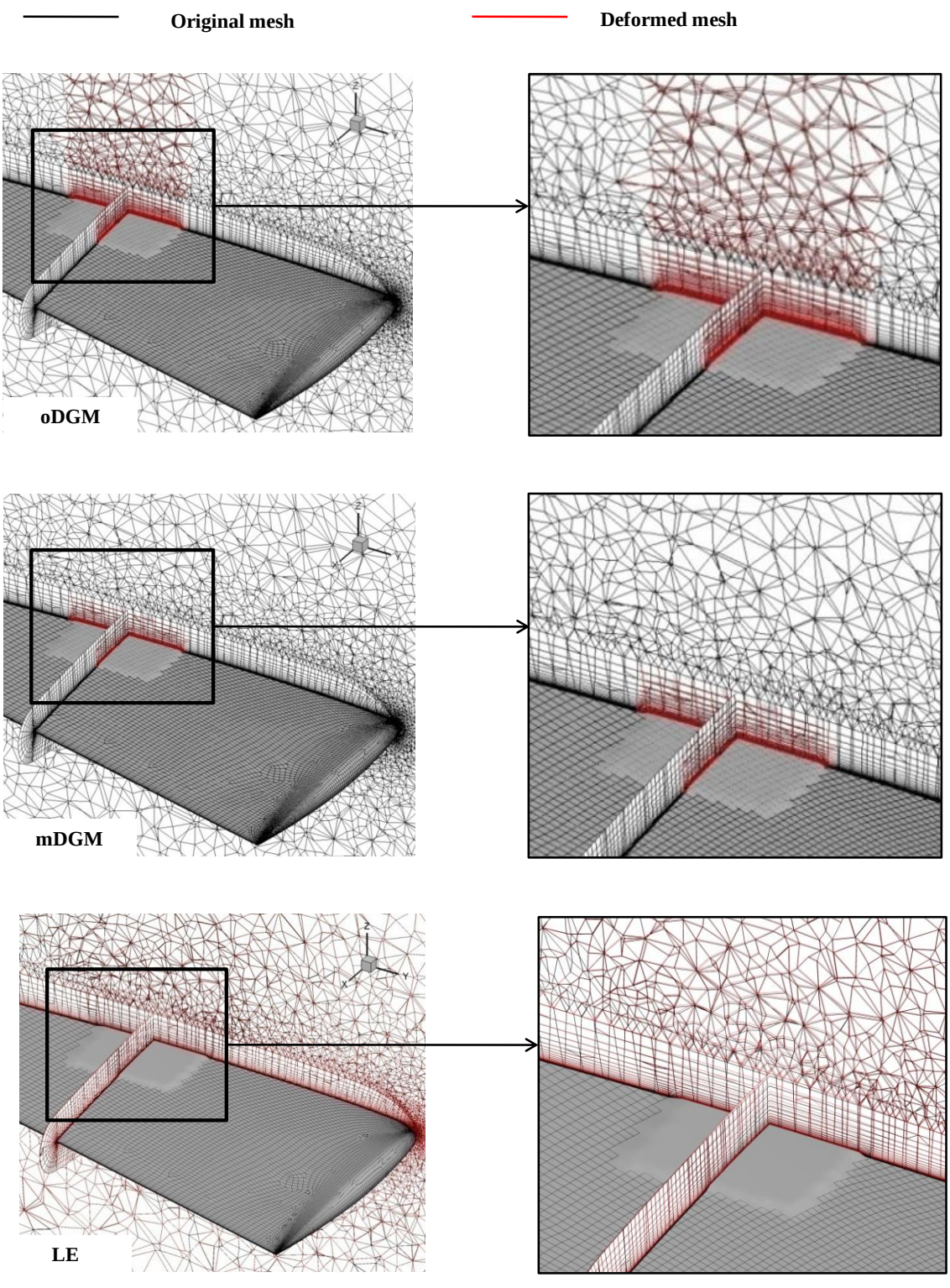


Figure 4.17 Comparison of different mesh movement methods.

From the data reported in Tab. 4.9, the mean mesh quality index does not seem to capture any relevant deviation for the deformations shown in Fig. 4.17. This is probably due to the more general, i.e. mean, feature this parameter describes. On the other hand, the dihedral angle seems to be more sensitive to the small changes investigated here.

It seems that by using the mDGM, which makes use of the supporting box, the quality of the deformed mesh is greatly improved as compared to the case where the oDGM is used. This results in a worst dihedral angle for the mDGM of just 1.83613 compared to the 0.52355 registered for the oDGM. The same trend is also registered for the number of elements featuring a dihedral angle lower than 5 degrees. In fact, while using the oDGM, 41 elements have a dihedral angle lower than 5 degrees which is quite high if compared to the 28 elements featured by the mDGM deformed mesh.

Interestingly, when the LE is compared against the oDGM, both metrics registered the same values. This can be explained by the fact that, the small deformations considered in these test cases are not enough to trigger a change. On the other hand, when the LE is compared against the mDGM, the latter is able to better preserve the quality of the mesh. However, it is expected that for larger deformations a non negligible deviation would be present.

4.8 Summary

In this chapter, the reasons why mesh deformation was chosen in place of mesh regeneration were given. These were indentified to be the need to maintain consistency and the lower computation time offered by mesh movements when compared to mesh regeneration strategies. On this matter, a brief review of the existing mesh movement methods was discussed along with a discussion of the relative advantages and disadvantages of each method.

One method was duly described, namely the DGM which was chosen as the main investigation tool because it provides the updated volume mesh explicitly without the need to solve a large system of equations as per the case where the LE is used. The DGM formulation was described in all its most important aspects along with one important modification. This addresses the quality of the initial Delaunay elements and distribution by constructing a supporting box. This has an impact on both the volume mesh update and the grid sensitivity.

In order to have a method to compare the DGM against, another mesh movement, namely LE was chosen. This can be considered representative of one of the best current implicit iterative strategies available. The comparison was performed both in terms of CPU time and memory requirements on different mesh refinement levels, where it was highlighted the advantages offered by the DGM. The reason for this was indentified in the different mathematical formulation of the methods being compared. In fact, where the DGM maps explicitly boundaries and volume mesh nodes, the LE provides the same map implicitly, meaning that an iterative procedure needs to be used to find the equilibrium satisfying the mesh movement governing equations.

Chapter 5 Mesh Sensitivity Based on the Delaunay Graph Method

5.1 Introduction

The flow adjoint successfully decouples the cost of obtaining the sensitivity of the flow variables w.r.t. the DVs from the N_{DV} (assuming that the DVs have an effect on the flow field). In contrast to the continuous approach (surface integral), the discrete approach requires the evaluation of the sensitivity of the volume mesh w.r.t. those DVs that control the aerodynamic shape. This part of the sensitivity chain, i.e. grid sensitivity, becomes more computationally expensive as the mesh grows in size and if an iterative mesh movement is considered. As Nielsen and Park [37] noted, this has hindered previous optimisations involving large meshes. This is especially true for unstructured grids.

As was mentioned in the literature review presented in Chap. 2, the current state-of-the-art method known to address the grid sensitivity computation is the mesh adjoint technique. Nielsen and Park [37] proposed solving an extra adjoint equation, i.e. mesh adjoint, to obtain the grid sensitivities. By doing so, instead of performing a series of repeated mesh deformations equal to the N_{DV}, a single mesh adjoint solution yields the grid sensitivity at a cost independent from the N_{DV}. However, the cost in both CPU time and memory involved in this approach needs to be further investigated as explained in Chap. 2.4.1.

In this chapter, the focus is on the differentiated mesh movement which is an important step closely related to the mesh movement strategy. Explicit algebraic methods offer the advantage of being less CPU and memory demanding because they avoid the need to solve a large and stiff linear systems. It is also explained how their differentiation offers the same advantages.

Having said that, this chapter describes the differentiation of an algebraic mesh movement method, i.e. DGM, along with the verification of the resulting gradient. As a second step, the DGM-based gradient is then compared against the LE-based gradient and a reasoning for the discrepancies is offered, Furthermore, these two methods are then compared in terms of CPU time and memory requirements where the advantages of the DGM over the LE-based gradient are highlighted.

The chapter concludes with a summary of the most important findings.

5.2 Importance of Analytical Approaches for Unstructured Meshes

In Chap. 2 the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ was introduced along with a discussion of its importance within the discrete adjoint framework, especially for unstructured grids. The computation of this term can be very memory and CPU time consuming, and this has encouraged researchers to find an alternative formulation either in the form of mesh movement hybridisation or algebraic mesh movements. Therefore, the computational feasibility of the grid sensitivity calculation in the discrete adjoint approach hinges on the development of a method featuring both low memory and low CPU time.

If the strategy employed to compute the grid sensitivity establishes the computation of the grid sensitivity of all the DVs at the same time, the memory is then proportional to the N_{DV} and N_{VM}. Since $N_{VM} \gg N_{DV}$, the grid size is of much more concern than the N_{DV}. On the other hand, if the strategy is such that for each DV, the grid sensitivity is computed and saved, the memory requirement is reduced and the cost becomes effectively no longer proportional to the N_{DV}, but the CPU time is increased. This last strategy, does not address either the CPU time or the memory issue. In fact, even if the grid sensitivity is computed for each DV, the cost of computing Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ still scales with the N_{VM}. Sending to the disk, requires bringing it to the memory sooner or later. A schematic view of these two strategies is presented in Fig. 5.1.

The justification to further investigate this matter lays on the fact that the mesh adjoint approach does not address the cost of inverting a very large and sparse matrix. Although this is a direct consequence of the mesh movement, it is also true that, all the time the mesh adjoint approach is used, it is in relation to an implicit iterative mesh movement. Furthermore, no comparison w.r.t. an explicit mesh movement method in terms or resulting gradients, CPU time and memory requirements has been studied so far. This chapter aims to provide an insight to these research questions.

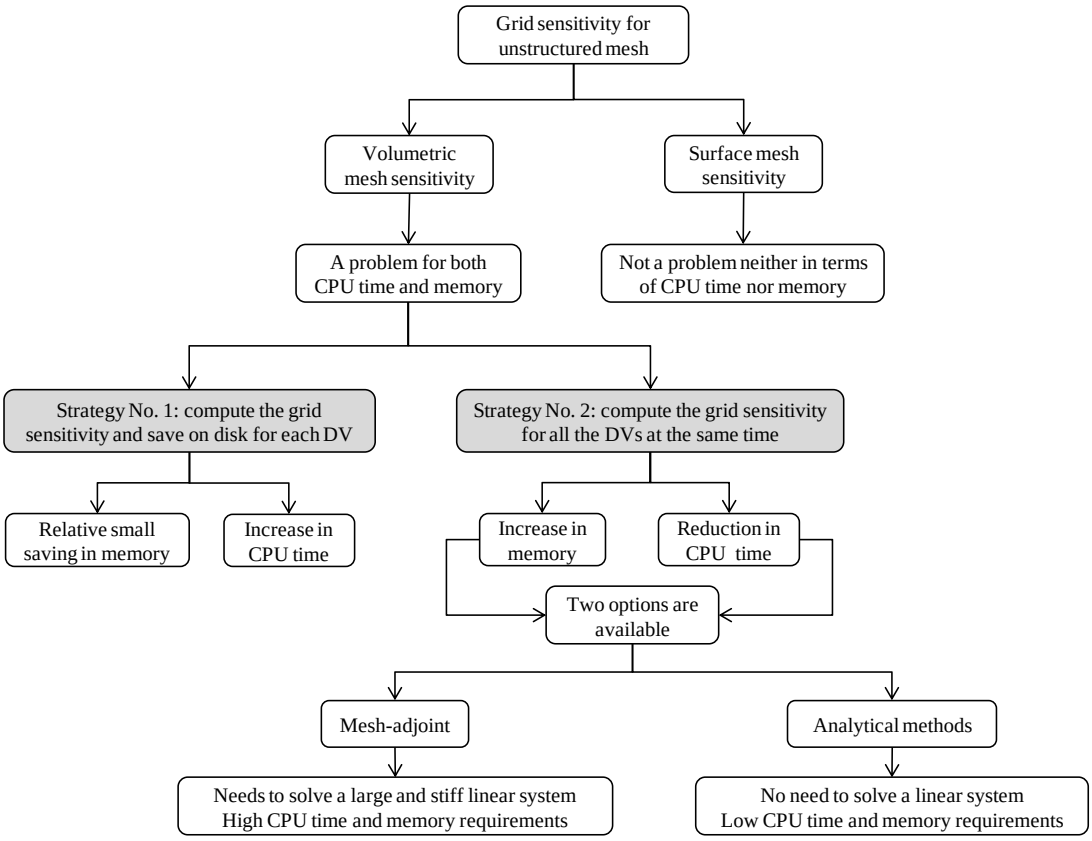


Figure 5.1 Two different strategies to tackle unstructured grid sensitivity.

5.3 DGM-based Gradient Chain

A method based on the DGM mesh movement is proposed to efficiently compute the mesh sensitivity in the discrete adjoint approach. The method is based on a one-to-one explicit algebraic map built between the volume and solid wall mesh nodes. The chain allows a straightforward computation of the mesh sensitivity without the need to invert a large and sparse matrix generally associated with any implicit mesh movement, such as the spring or LE analogy. This substantially improves the efficiency in terms of CPU time and memory requirements.

If an explicit mesh movement is used, there are two methods to derive the chain. The first one is based on the construction of the Lagrangian for the mesh movement constraint, whereas the second is based directly on the linearisation of the DGM explicit mesh movement. Both methods are described because each one of them highlights a different feature of the method. In

fact, the former allows one to make an easy and elegant comparison against the LE, whereas the latter shows a way to obtain the volumetric grid sensitivity directly from the available analytical map.

5.3.1 First Method to Derive the Chain

The mesh adjoint method was duly introduced in the literature review (see Chap. 2.5.6) in its fundamental aspects, but the differentiation of the mesh movement was not there discussed. Here, the equations are briefly re-introduced in order to allow an easy-to-follow derivation along with the full differentiation of the mesh movement constraint. Two Lagrangian multipliers are constructed, one for the flow and the other for the mesh constraint:

$$\mathcal{L}\left(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X},\boldsymbol{\Lambda}_F,\boldsymbol{\Lambda}_{G,DGM}\right) = I(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X}) + \{\boldsymbol{\Lambda}_F\}^T\{\boldsymbol{R}(\boldsymbol{D},\boldsymbol{W},\boldsymbol{X})\} + \left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T\{\boldsymbol{T}(\boldsymbol{X},\boldsymbol{D})\} \tag{5.1}$$

Differentiating the augmented cost function w.r.t. the DVs vector yields:

$$\begin{aligned}\left\{\frac{dI}{d\boldsymbol{D}}\right\} = &\left\{\frac{\partial I}{\partial \boldsymbol{W}}\frac{d\boldsymbol{W}}{d\boldsymbol{D}} + \frac{\partial I}{\partial \boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial I}{\partial \boldsymbol{D}}\right\} + \{\boldsymbol{\Lambda}_F\}^T\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\frac{d\boldsymbol{W}}{d\boldsymbol{D}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}}\right] \\ &+ \left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\frac{d\boldsymbol{X}}{d\boldsymbol{D}} + \frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right]\end{aligned} \tag{5.2}$$

Under the assumption of pure geometric changes, Eq. (5.2) becomes:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{W}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right\}\left[\frac{d\boldsymbol{W}}{d\boldsymbol{D}}\right] + \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_{G,DGM}^T\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] + \left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right] \tag{5.3}$$

There are two important terms in Eq. (5.3), namely sensitivity of the state variables w.r.t. the DVs, and the sensitivity of volume mesh points w.r.t. the DVs. The first expensive term can be eliminated constructing the canonical flow-adjoint system (see Eq. (2.4). After solving the flow-adjoint system, the flow-adjoint vector is used to form the final total sensitivity:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_{G,DGM}^T\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] + \left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right] \tag{5.4}$$

The matrix $[d\boldsymbol{X}/d\boldsymbol{D}]$ of dimensions $[N_{VM} \times N_{DV}]$ comes from the differentiation of the mesh movement and represents the sensitivity of the volume mesh w.r.t. the DVs. This total derivative consists of two different components, namely volumetric and surface grid sensitivity.

The most expensive in terms of memory and CPU time is the former, i.e. Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$. It is obvious that for large unstructured grids this can become an issue. To address this issue it is necessary to analyse the DGM mesh movement and differentiate it. The mesh movement residual associated with it read as (from Eq. (4.5)):

$$\{\boldsymbol{T}(\boldsymbol{X},\boldsymbol{D})\}_{DGM} = \{\boldsymbol{X}\} - [\boldsymbol{E}]\{\boldsymbol{B}\} = \{\boldsymbol{0}\} \tag{5.5}$$

Differentiating w.r.t. the $\{\boldsymbol{D}\}$ yields:

$$\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right] = [\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.6}$$

where $[\partial \boldsymbol{B}_{ff,sb}/\partial \boldsymbol{D}] = [\boldsymbol{0}]$ and $[\partial \boldsymbol{B}_{sw}/\partial \boldsymbol{D}] = [\partial \boldsymbol{S}/\partial \boldsymbol{D}]$ because $\{\boldsymbol{D}\}$ is the DV vector that controls the parameterised shape which coincides with the solid wall surface mesh nodes only. In fact, both far field and supporting box mesh nodes are not allowed to move. Considering the differentiated general mesh movement in Eq. (2.12), the following relations hold:

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\right]_{DGM} = [\boldsymbol{I}] \tag{5.7}$$

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right]_{DGM} = -\left[\boldsymbol{E}\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.8}$$

where matrix $[\boldsymbol{I}]$ is the identity matrix. Substituting both Eqs. (5.7, 5.8) back in Eq. (5.4), the final total sensitivity reads:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_{G,DGM}^T\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] - \left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T\left[\boldsymbol{E}\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.9}$$

It is straightforward to see that the cost associated with the evaluation of the sensitivity of the volume mesh nodes w.r.t. the DVs can be eliminated by setting:

$$\left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T = -\left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\} \tag{5.10}$$

Substituting Eq. (5.10) back in Eq. (5.9) yields:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{DGM} = -\left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T[\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.11}$$

where matrix $[\boldsymbol{E}]$ is exactly the same matrix used for the mesh movement presented in Chap. 4.5.2. Therefore, it is possible to use an important part of the chain employed in the mesh movement also for the evaluation of the mesh sensitivity with no additional computation. For the particular case where each surface mesh point is considered as a DV, i.e. $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$, Eq. (5.11) reduces to:

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_{DGM} = -\left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T [\boldsymbol{E}] \tag{5.12}$$

Under this assumption, the sensitivity is equal to a simple inexpensive product. Precisely, Eq. (5.12) prescribes for each surface mesh point the magnitude and direction to decrease or increase the objective function being analysed. This allows one to map a design space that scales with the N_{SM} knowing that the cost does not scale with the $N_{SM}\,(= N_{DV})$. Examples of how this map can be useful to visualise the most sensitive areas can be found in Palacios *et al.* [45] for supersonic flow regimes using the continuous adjoint and Hinchliffe and Qin [46] for transonic flow regimes using the discrete approach.

5.3.2 Second Method to Derive the Chain

There exists a second way to derive the final gradient. This method is based directly on the explicit mesh movement differentiation, and contrary to the method described in Chap. 5.3.1, is not based on the mesh adjoint vector. This is possible because the differentiated Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is available explicitly. Consider the gradient expressed as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{dI}{d\boldsymbol{X}}\right\}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.13}$$

If $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$:

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\} = \left\{\frac{dI}{d\boldsymbol{X}}\right\}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right] \tag{5.14}$$

where it can be proven differentiating Eq. (2.1) w.r.t. the volume mesh that:

$$\left\{\frac{dI}{d\boldsymbol{X}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\} \tag{5.15}$$

Direct differentiation of the volume mesh dependency $\{\boldsymbol{X}\} = \{\boldsymbol{X}(\boldsymbol{S}(\boldsymbol{D}))\}$ w.r.t. the geometric DVs yields:

$$\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right] = \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.16}$$

Differentiating the DGM mesh movement as expressed in Eq. (4.5) w.r.t. the DVs yields:

$$\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right] = [\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.17}$$

Comparing Eq. (5.17) with Eq. (5.16) leads to:

$$[\boldsymbol{E}] \equiv \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right] \tag{5.18}$$

and comparing Eq. (5.14) with Eq. (5.12) leads to Eq. (5.10). Substituting Eq. (5.18) back in Eq. (5.13), it is clear that the same final gradient (i.e. Eq. (5.11)) expression is recovered. Therefore, it was proven how, if an explicit algebraic mesh movement capable of providing explicitly the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is used, the second Lagrangian associated to the mesh constraint does not need to be constructed.

The reason of these two different derivations was to facilitate the mathematical comparison against the LE for which the construction of the mesh adjoint is not a choice.

5.4 Verification

The DGM-based gradient was verified for the 2D RAE 5243 [132] aerofoil and for the 3D ONERA M6 [133] wing for two cases: verification w.r.t. each surface point against FDs, and verification based on the analytical manual linearisation of the drag, lift, and longitudinal moment coefficient w.r.t. the AoA (Angle of Attack). Complex step differences could have been used, but this was not an option available for the version of the DLR TAU code used by the author.

At this point, it is important to highlight the difference between verification and validation. They are both used in the literature to check if the results are correct. Verification is the

procedure to check if the *equation is solved correctly*, whereas validation is the procedure used to check if the *correct equation is being solved*. For the former, a simple comparison against a small variation obtained via FD is used, whereas for the latter a comparison against wind tunnel data is required. For this work, verification was the only option available.

A couple of fundamental important observations can be made on the use of FD gradients in relation to the grid size. Anderson and Venkatakrishnan [35] proved that for the continuous adjoint approach, if insufficient grid resolution is used, gradients may not agree with those obtained using FDs. On the other hand, if the discrete adjoint is used, the gradients agree with the FDs regardless to the mesh resolution. However, Hou *et al.* [134] noted that, when verifying the chain using a thick mesh close to the wall, FD gradients are overly sensitive to both step size and flow solver convergence. For this reason, Anderson and Bonhaus [135] suggested a coarse wall grid be used in order to avoid the need to deform the mesh and therefore reduce the sensitivity from the step size and flow solver convergence.

In conclusion the adjoint gradients, by definition should agree with those obtained with FDs, as long as the chain is discretely adjoined (i.e. no frozen turbulent viscosity) and the step size is carefully chosen.

5.4.1 Test Cases

The canonical conditions for the NLF (Natural Laminar Flow) RAE 5243 aerofoil [132] prescribe a freestream Mach number of 0.68 at constant lift coefficient of 0.82. Fig. 5.2 depicts the close-wall mesh (top), pressure contours (centre) and the y^+ (bottom). The mesh consists of 95,972 volume mesh nodes and the Reynolds number is 19 Mil. based on the Reynolds chord length equal to 1. The canonical conditions for the ONERA M6 wing [133] prescribe a freestream Mach number of 0.83 at constant lift coefficient of 0.2788. Fig. 5.3 depicts the close-wall mesh (top), pressure contours (centre) and the y^+ (bottom). The mesh consists of roughly 1.6 Mil. volume mesh nodes and the Reynolds number is 11.72 Mil. based on the Reynolds chord length equal to 0.64607. All the hybrid meshes were generated using the unstructured quad-dominant software SOLAR [136].

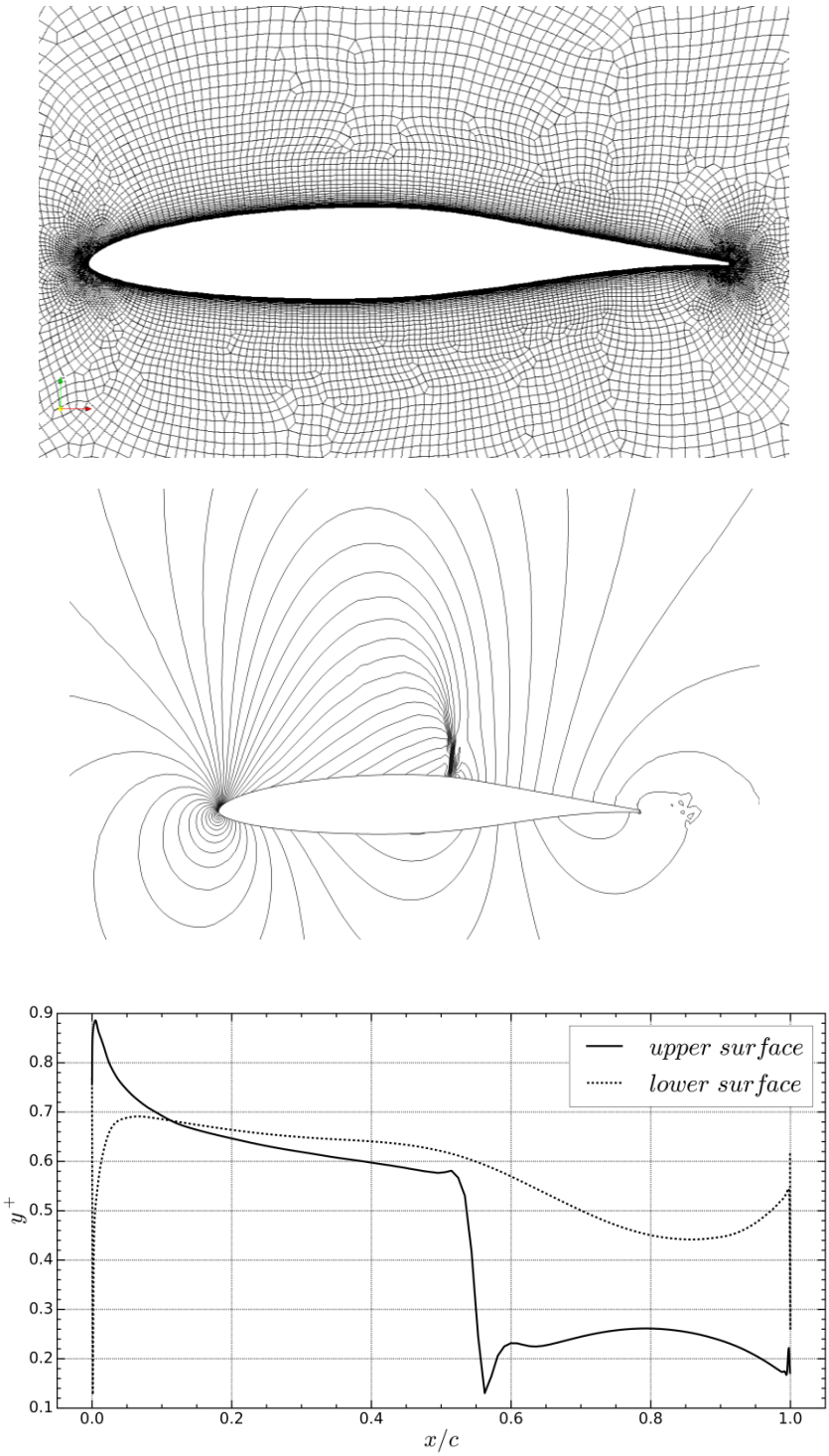


Figure 5.2 Close-up view of the mesh at the solid wall (top), pressure contours (centre) and y-plus (bottom) for the RAE 5243.

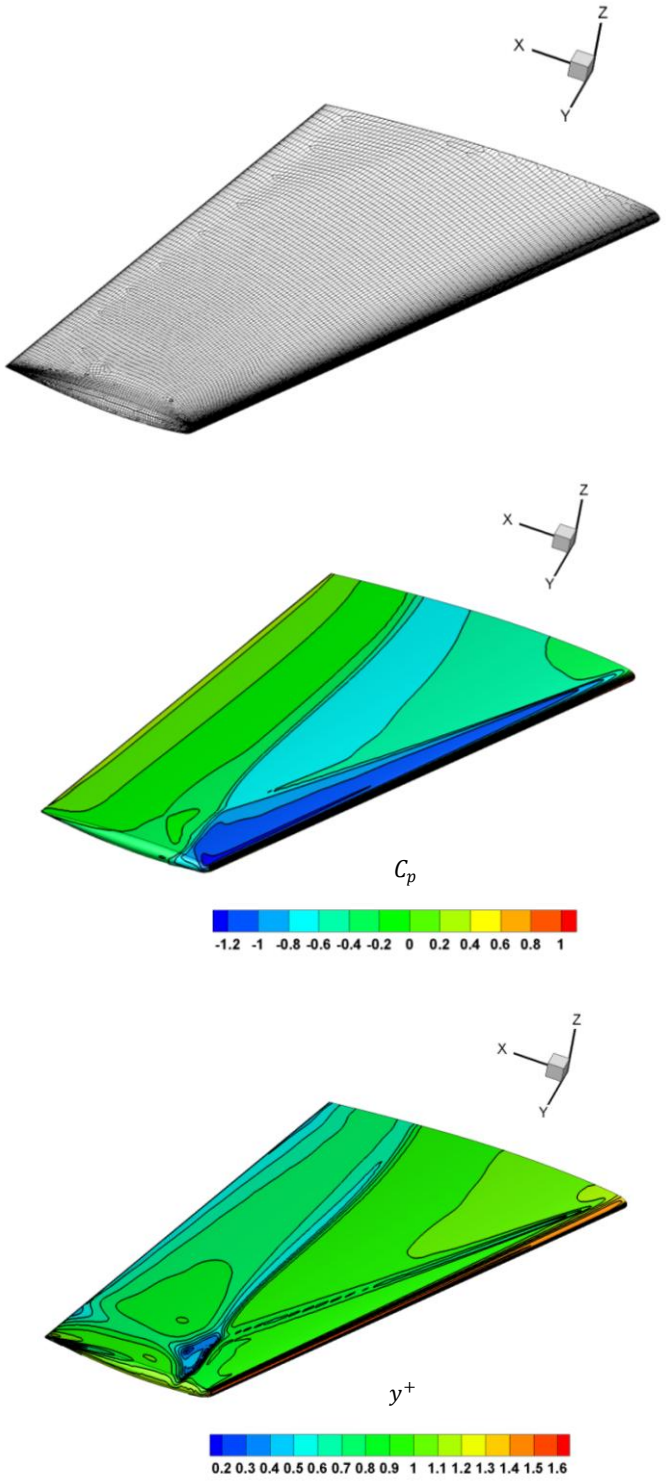


Figure 5.3 Unstructured surface mesh (top), pressure contours (centre) and y-plus (bottom) for the ONERA M6 wing.

5.4.2 Verification Against each Surface Mesh Point

The DLR TAU built-in tool VOLGRAD short for "*VOLume approach for GRADient calculation*" was used as the main FD verification tool. Consider the total sensitivity written as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{D}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}}\right\} \tag{5.19}$$

VOLGRAD evaluates the discrete version of Eq. (5.19) which reduces to a simple sum of scalar differences. For sake of simplicity, let us consider the forward FD formulation:

$$\frac{dI}{dD} \approx \frac{\Delta I}{\varepsilon} = \frac{(I_M - I_0) + \boldsymbol{\Lambda}_F^T (R_M - R_0)}{\varepsilon} \tag{5.20}$$

where subscript '*0*' and '*M*' denote the initial and modified (following a small deformation) value. ΔI is the variation in the cost function due to the chosen perturbation ε, whereas I_M and R_M are computed on the perturbed grid generated by the DGM mesh movement, where R_M is obtained interpolating the residual on the modified mesh. I_0 and R_0 come from the flow solution, computed on the original (not modified) mesh. Eq. (5.20) involves two evaluations of the residual vector for each DV and not two flow solutions. The verification process tries to estimate the error expressed as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{DGM} - \left\{\frac{dI}{d\boldsymbol{D}}\right\}_{FDs} \equiv Error \tag{5.21}$$

If the step size is too small, FDs are susceptible to round-off error, whereas on the other hand a step size too big results in too large truncation errors. For this reason a parametric study is needed to find a range of steps where the resulting sensitivities do not change. The step size used for the verification was 10^{-6} and the sensitivities do not change in the range between 10^{-7} and 10^{-5}.

For the RAE 5243 2D case, all the points belonging to the upper surface (see Fig. 5.4) were finite differenced. However, the ONERA M6 wing has roughly 30,000 upper surface mesh nodes, thus only one spanwise station was selected consisting of roughly 100 surface mesh nodes as shown in Fig. 5.5 (top right). This gives a good overview of the sensitivity with minimal computation cost.

Referring to Figs. 5.4 and 5.5 (top right), the results obtained using FDs are first shown using red coloured vectors overlapped onto the black ones to provide an easy-to-follow spatial view. This is then followed by a point-to-point comparison (see Figs. 5.4 and 5.5 (left)) where the error is plotted along the z-coordinate. The gradient at each surface mesh compares very well away from the leading and trailing edge where the error increases, but the error at the leading edge seems to be less severe than that registered at the trailing edge. There are four reasons for the registered discrepancies:

- Inadequate FD step size
- Inaccuracy in the surface normal computation
- Interpolation error in the flow residual
- Flow-adjoint solver not fully discretely adjoined

In theory, each surface point requires its own parametric study which is known to be very sensitive when the mesh elements are very close to each other (close to the wall) as noted by Anderson and Bonhaus [135]. Secondly, the normal is ill-defined at sharp edges. Thirdly, the interpolation error on the residual computed by VOLGRAD, is expected to be greater where the solution is not optimal and this generally happens close to strong gradients. Lastly, the solver may not be fully adjoined. This possibility, which is not so rare, can ultimately lead to some discrepancies as noted by Anderson and Venkatakrishnan [35].

To conclude, the four points mentioned above, when considered applied at the trailing edge, make the FD method difficult to use and this explains the large error in that area.

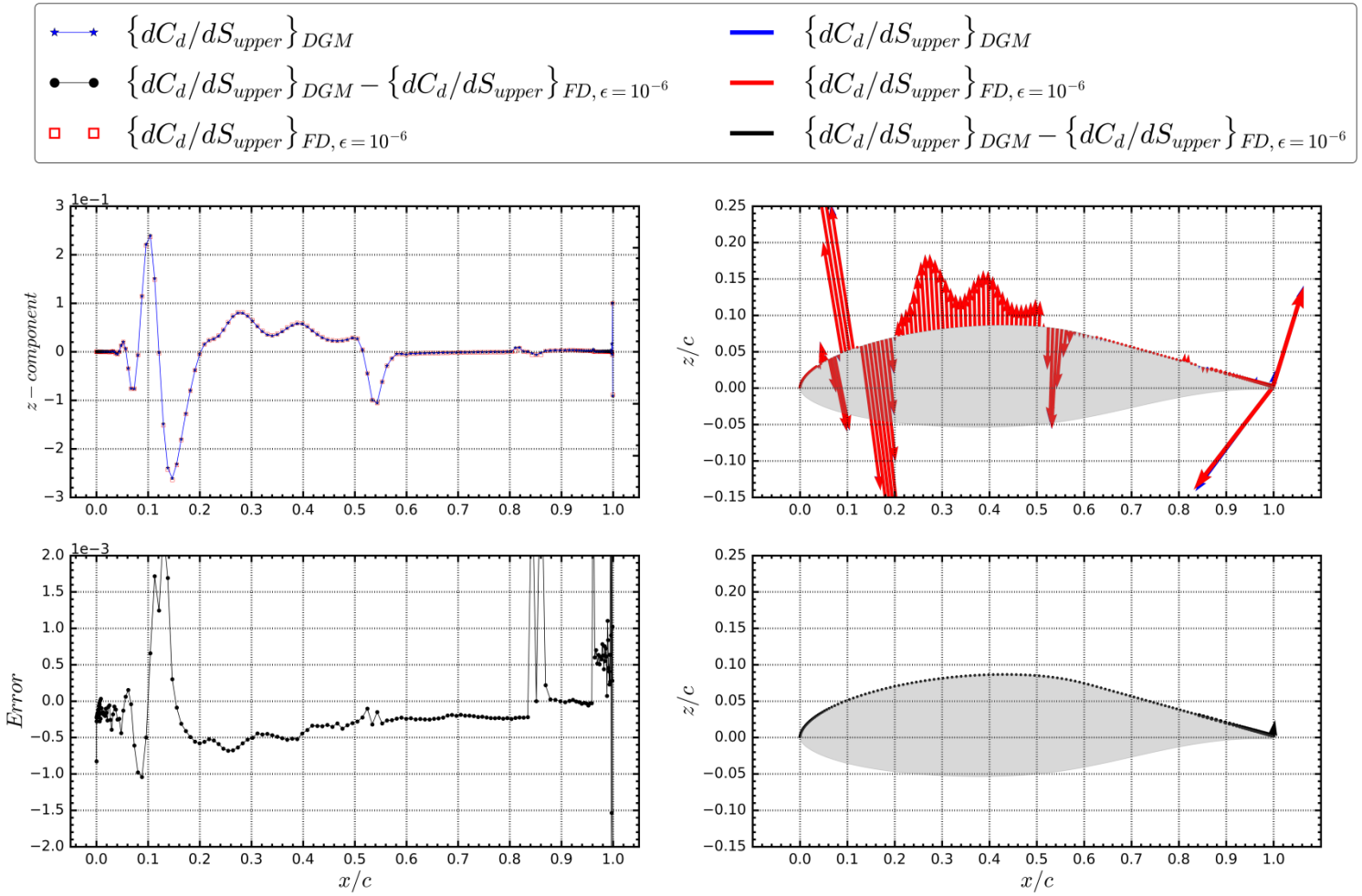


Figure 5.4 Comparison between the DGM and FD gradients for the RAE 5243.

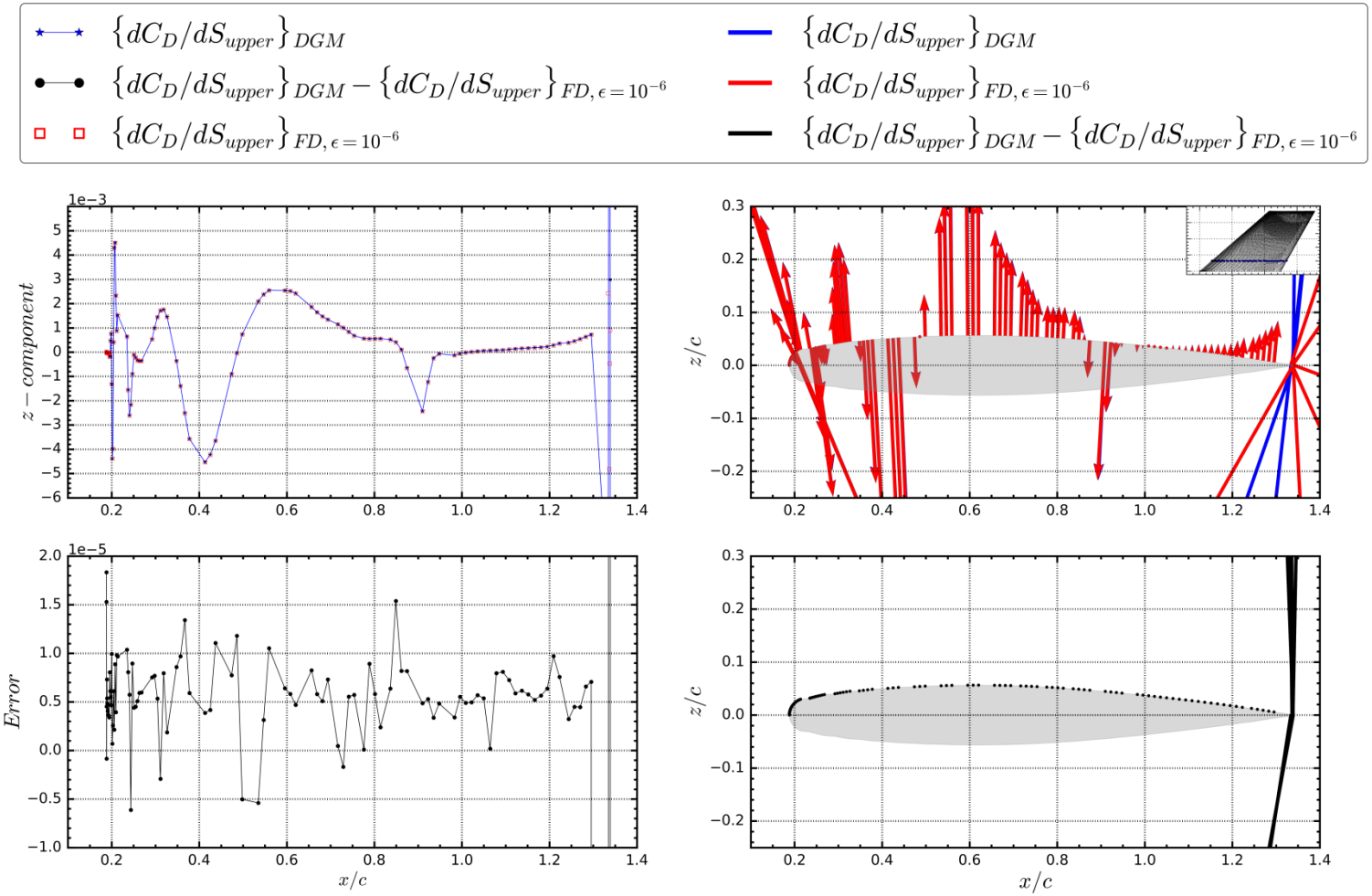


Figure 5.5 Comparison between the DGM and FD gradients for the ONERA M6 wing.

5.4.3 Verification Against a Small Change in the Angle of Attack

Another type of verification is represented by the comparison against a small change in the AoA. In the DLR TAU-Code the sensitivity of the drag and lift coefficients w.r.t. the AoA is obtained as:

$$\left(\frac{dI}{d\alpha}\right)_{TAU} = \left(\frac{\partial I}{\partial \alpha}\right) + \{\boldsymbol{\Lambda}_F\}^T \left\{\frac{\partial \boldsymbol{R}}{\partial \alpha}\right\} \tag{5.22}$$

where first and second RHS terms are differentiated manually and no simplification are made. Therefore no FDs are used in the computation of Eq. (5.22). From Eq. (5.11), adapting the equation to the problem at hand, the DGM-based chain yields:

$$\left(\frac{dI}{d\alpha}\right)_{DGM} = -\left\{\boldsymbol{\Lambda}_{G,DGM}\right\}^T [\boldsymbol{E}] \left\{\frac{\partial \boldsymbol{S}}{\partial \alpha}\right\} \tag{5.23}$$

This can be decomposed in its Cartesian components (where for the 2D verification case the y-contribution is null):

$$\left(\frac{dI}{d\alpha}\right)_{DGM} = -\left\{\boldsymbol{\Lambda}_{G,DGM}^T \boldsymbol{E}\right\}_x \left\{\frac{\partial \boldsymbol{S}}{\partial \alpha}\right\}_x - \left\{\boldsymbol{\Lambda}_{G,DGM}^T \boldsymbol{E}\right\}_y \left\{\frac{\partial \boldsymbol{S}}{\partial \alpha}\right\}_y - \left\{\boldsymbol{\Lambda}_{G,DGM}^T \boldsymbol{E}\right\}_z \left\{\frac{\partial \boldsymbol{S}}{\partial \alpha}\right\}_z \tag{5.24}$$

The partial derivatives $\{\partial \boldsymbol{S}/\partial \alpha\}_{x,y,z}$ are known after a rotation of the solid wall about the pitching moment reference point. It is clear that, the verification against a small change in the AoA represents a more difficult test as all the solid surface mesh points, including those noisy values at the trailing edge as depicted in Figs. 5.4 and 5.5, are now taken into account simultaneously.

The goal of this verification is to quantify the following error:

$$\left(\frac{dI}{d\alpha}\right)_{DGM} - \left(\frac{dI}{d\alpha}\right)_{TAU} \equiv Error \tag{5.25}$$

The results are reported in Tab. 5.1, illustrating a high degree of accuracy.

Table 5.1 Results for the verification against a small change in the AoA.

Case	$\left(\frac{dC_{d,D}}{d\alpha}\right)_{DGM}$	$\left(\frac{dC_{d,D}}{d\alpha}\right)_{TAU}$	**Error [%]**
RAE 5243	**0.00556**3	**0.00556**7	-0.0719
ONERA M6	**0.007378**2	**0.007378**4	-0.00271
Case	$\left(\frac{dC_{l,L}}{d\alpha}\right)_{DGM}$	$\left(\frac{dC_{l,L}}{d\alpha}\right)_{TAU}$	**Error [%]**
RAE 5243	**0.133**995	**0.133**864	+0.0977
ONERA M6	**0.091**4981	**0.091**3225	+0.192
Case	$\left(\frac{dC_{my,MY}}{d\alpha}\right)_{DGM}$	$\left(\frac{dC_{my,MY}}{d\alpha}\right)_{TAU}$	**Error [%]**
RAE 5243	**0.0012**47	**0.0012**54	-0.558
ONERA M6	**0.0014**16	**0.0014**24	-0.562

5.5 Gradient Chain Based on the Linear Elasticity

Considering Eq. (4.16), the mesh movement residual associated with the LE is defined as:

$$\{\boldsymbol{T}(\boldsymbol{X},\boldsymbol{D})\}_{LE} = [\boldsymbol{K}]\{\boldsymbol{X}\} - \{\boldsymbol{S}\} = \{\boldsymbol{0}\} \tag{5.26}$$

Differentiating w.r.t. the DVs vector yields:

$$[\boldsymbol{K}]\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right] = \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.27}$$

Considering the differentiated general mesh movement in Eq. (2.12) the following relations hold:

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{X}}\right]_{LE} = [\boldsymbol{K}] \tag{5.28}$$

$$\left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right]_{LE} = \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.29}$$

As noted by Nielsen and Park [37], the cost associated to the solution of Eq. (5.27) is comparable to the cost of a single mesh movement. Substituting Eqs. (5.28) and (5.29) back in Eq. (5.4), the final total sensitivity reads as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{LE} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_{G,LE}^T \boldsymbol{K}\right\}\left[\frac{d\boldsymbol{X}}{d\boldsymbol{D}}\right] + \{\boldsymbol{\Lambda}_{G,LE}\}^T \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.30}$$

The expensive term, i.e. $[d\boldsymbol{X}/d\boldsymbol{D}]$, can be eliminated constructing a second mesh adjoint linear system of equations as suggested by Nielsen and Park [37]:

$$[\boldsymbol{K}]^T\{\boldsymbol{\Lambda}_{G,LE}\} = -\left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\}^T \tag{5.31}$$

This system is most of the time large and very stiff to invert. This is because the viscous meshes present a very high aspect ratio making the system poorly conditioned (see also Chap. 4.6). The system is then solved using a restarted preconditioned ILU GMRES algorithm. The convergence of Eq. (5.31) is the same as the mesh deformation in Eq. (4.16) because the eigenvalues of matrix $[\boldsymbol{K}]$ are not changed by the transpose operator. Once the mesh adjoint vector is obtained, the final gradient reads:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{LE} = -\{\boldsymbol{\Lambda}_{G,LE}\}^T \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right] \tag{5.32}$$

For the particular case where $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$, Eq. (5.32) reduces to:

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_{LE} = -\{\boldsymbol{\Lambda}_{G,LE}\}^T \tag{5.33}$$

Tab. 5.2 helps highlight the differences between the LE and DGM-based gradient. The most striking difference between the two methods is that, where the LE needs to solve a linear system of equations the DGM needs to perform only a dot product. This conclusion can then be extended to any implicit or explicit mesh movement.

Table 5.2 Comparison between the LE and DGM-based gradient formulation.

Assumptions	Mesh movements	Required iterative computations	Gradient
Only geometric changes but $\{\boldsymbol{D}\} \neq \{\boldsymbol{S}\}$	DGM: $\{\boldsymbol{X}\} = [\boldsymbol{E}]\{\boldsymbol{B}\}$	Primal and dual solution only	$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{DGM} = -\{\boldsymbol{\Lambda}_{G,DGM}\}^T [\boldsymbol{E}] \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right]$
	LE: $[\boldsymbol{K}]\{\boldsymbol{X}\} = \{\boldsymbol{S}\}$	Primal, dual and mesh adjoint solution	$\left\{\frac{dI}{d\boldsymbol{D}}\right\}_{LE} = -\{\boldsymbol{\Lambda}_{G,LE}\}^T \left[\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{D}}\right]$
Only geometric changes and $\{\boldsymbol{D}\} = \{\boldsymbol{S}\}$	DGM: $\{\boldsymbol{X}\} = [\boldsymbol{E}]\{\boldsymbol{B}\}$	Primal and dual solution only	$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_{DGM} = -\{\boldsymbol{\Lambda}_{G,DGM}\}^T [\boldsymbol{E}]$
	LE: $[\boldsymbol{K}]\{\boldsymbol{X}\} = \{\boldsymbol{S}\}$	Primal, dual and mesh adjoint solution	$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_{LE} = -\{\boldsymbol{\Lambda}_{G,LE}\}^T$

5.6 Effect of the Supporting Box

As discussed in Chap. 4.5.3, the necessity to have a relatively low skewed Delaunay elements is not only related to the mesh movement, but also to another reason related to the way the gradient is computed. In fact, when it comes to complex aero shape geometries, the triangulation/tetrahedralisation can be challenging, meaning the resulting map is not optimal. If it is so, there two main consequences. Firstly, a reduced quality of the deformed mesh is obtained, and secondly, the map established by DGM-based Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}](= [\boldsymbol{E}])$ is no longer respected.

To understand the importance of the second observation, a distinction is made between quality and distribution of the Delaunay elements. The quality of the Delaunay elements can be judged by using metrics such as mean quality and the dihedral angle as already discussed in Chap. 4.7.4. On the other hand, more importantly than the Delaunay quality elements is to have a correct Delaunay distribution. In other words, the Delaunay elements must map only the surface and volume mesh points which means that at least one vertex, both for a triangle or a tetrahedron must not be a solid wall surface mesh node. The distinction is important because there are cases where, a Delaunay element quality is considered good, but its distribution is not

correct. To be more precise, the issue comes when the map is made from surface-to-surface. In fact, matrix $[\boldsymbol{E}]$ represents the sensitivity of the volume mesh w.r.t. the surface, i.e. Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$. Therefore, having all the Delaunay element vertices on the solid wall is not correct.

This surface-to-surface map happens because the QHull library [120] works based on the Delaunay criterion and there is nothing wrong from a mathematical point of view if such as map is constructed. As an example, Fig. 5.6 (left) depicts one of this case for the DLR F6 wing. A quick solution to this issue is, as mentioned in Chap. 4.5.3, to introduce a supporting box, depicted in Fig. 4.9 (centre). This, as depicted in Fig. 5.6 (right), avoids the surface-to-surface map issue.

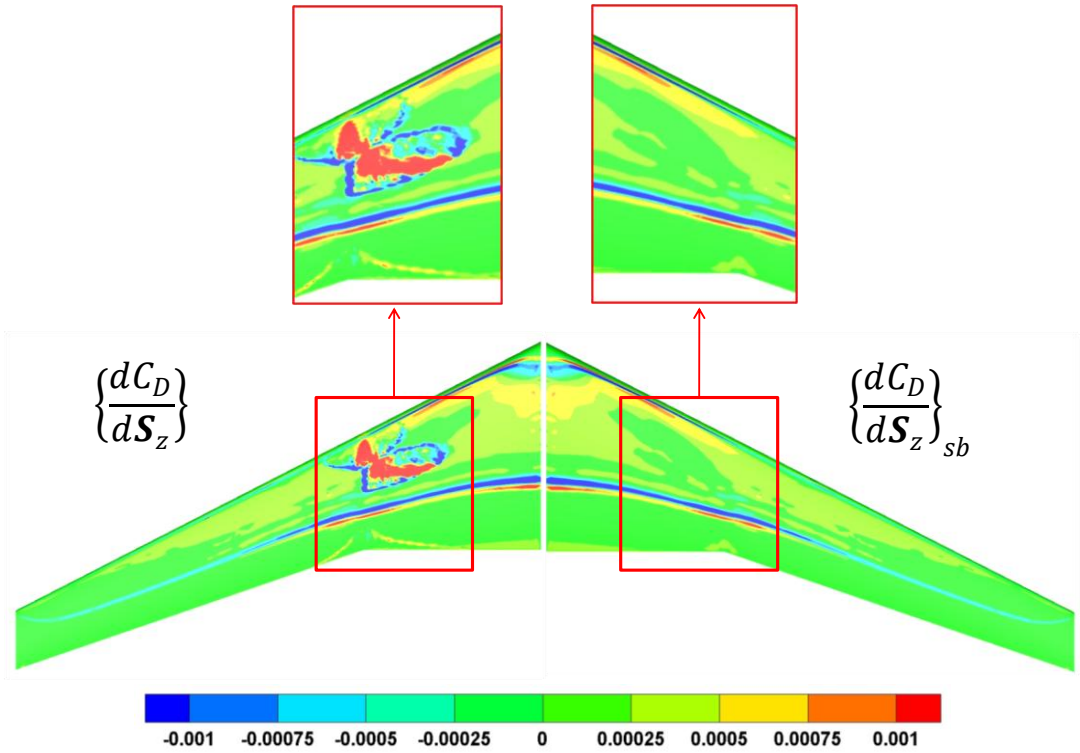


Figure 5.6 Effect of the supporting box on the gradient for the DLR F6 wing.

5.7 Analysis of the Chain

Before starting this discussion, a distinction has to be made between accuracy and consistency or in other words, how well the flow field is resolved and how rigorous the sensitivity analysis is respectively. The first point is generally concerned on how well the solution agrees with the physics and the validation (as opposed to verification) process is the only option available to check (i.e. validate) it. However, this is very difficult to perform as it requires expensive wind tunnel test campaigns. The second point refers to the mathematical rigour of the chain derivation. In other words, is the chain fully differentiated? For instance, one can even start with a turbulence model which is not best suited for the flow field, but consistently derive the chain (e.g. no frozen turbulence model assumption). In this case, the chain is consistent although the results will not be accurate.

Fig. 5.7 helps understand these differences and where the academic effort was concentrated in the last 30 years (dashed square in Fig. 5.7). As can be seen, the past and current effort has so far focused on the differentiation of *low fidelity* turbulence models which, although not extremely accurate, are currently the only viable options. Examples of differentiated algebraic method can be found in Hou *et al.* [134], whereas examples where a field equation for the eddy viscosity is used can be found in Anderson and Bonhaus [135] for the one-equation, in Kim *et al.* [137] for the two-equations and in Khayatzadeh and Nadarajah [138] for the four-equations. Differentiating a turbulence model is a tedious task and for this reason many studies were published using the frozen turbulence assumption.

In this work, the fully differentiated SA one-equation turbulence model [82, 88] is used. This places the following analysis in the consistent but not accurate framework as depicted in Fig. 5.7. The choice was dictated by three factors, namely available computation power where one is forced to rely on a time-averaged solution, difficulty related to the differentiation of more complex turbulence modelling and more importantly the conditioning of the flow-adjoint Jacobian.

In this section the underlying physics behind the three main steps necessary to compute the gradient are described. The analysis is structured as follows:

1) Flow-adjoint solution focused on the meaning of the flow-adjoint vector

2) Sensitivity of the objective function w.r.t. the volume mesh, i.e. $\{dI/d\boldsymbol{X}\}$, focused on its distribution over the entire computational domain
3) Gradient $\{dI/d\boldsymbol{S}\}$ and how is influenced by both $\{dI/d\boldsymbol{X}\}$ and $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$

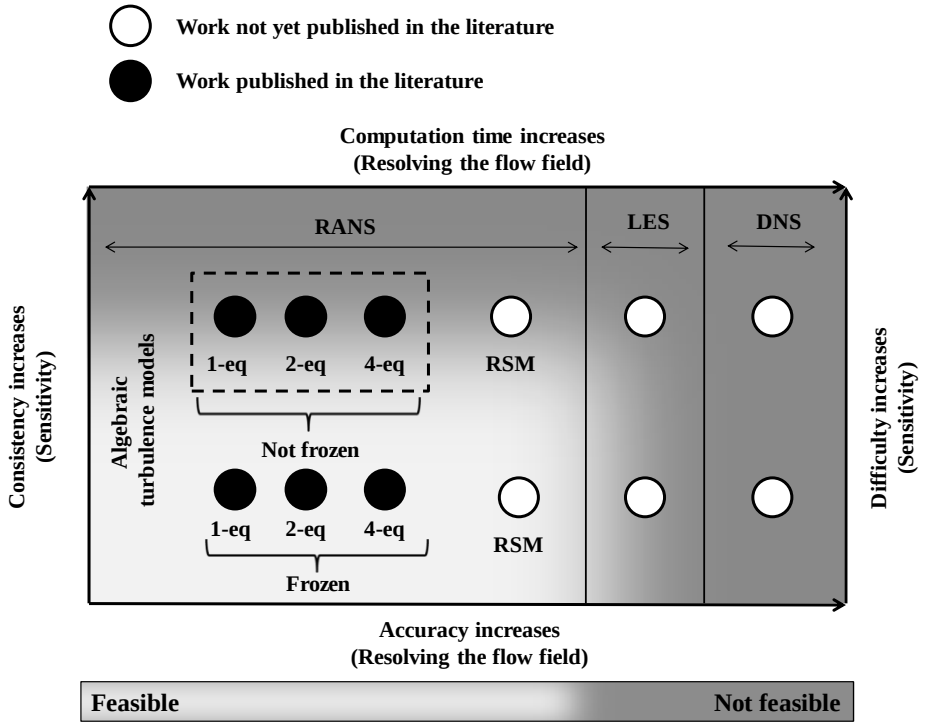


Figure 5.7 Conceptual comparison between computation time, accuracy, difficulty, consistency and feasibility of the adjoint chain.

5.7.1 Flow-adjoint Solution

The flow-adjoint solution delivers the flow-adjoint vector which has the same dimensions of the flow variables vector and can be re-written as (using a one-equation turbulence model):

$$\{\boldsymbol{\Lambda}\} = \{\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5, \Lambda_6\}^T \tag{5.34}$$

Note that, if the frozen turbulence assumption is used, the adjoint value associated to the turbulence model, i.e. Λ_6, expresses the sensitivity w.r.t. the laminar viscosity, only because the turbulent viscosity is considered constant.

The adjoint vector has found various applications in the literature. Apart from being widely used in sensitivity analysis, there are applications in error estimation and mesh quality refinement. These last two are closely related as the error estimated via the adjoint vector is then used to drive the mesh refinement (re-meshing) or adaptation of the discretised domain. Valid examples of these applications can be found in Giles and Suli [139] for inviscid cases and Venditti and Darmofal [140] for viscous cases.

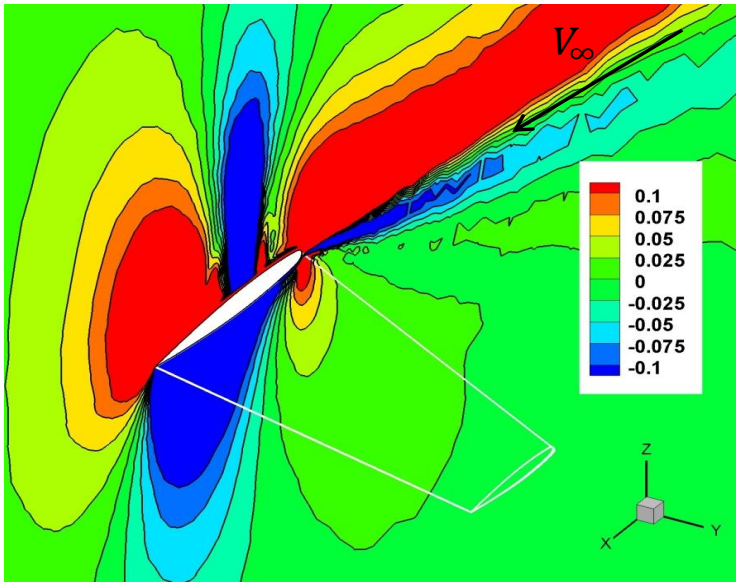
The adjoint vector gives the sensitivity of the objective function w.r.t. the residual, i.e. level of convergence in the primal solution [32]:

$$\{\boldsymbol{\Lambda}_F\} = \left\{\frac{\partial I}{\partial \boldsymbol{R}}\right\} \tag{5.35}$$

This can be obtained multiplying the RHS and LHS of Eq. (2.4) by matrix $[\partial \boldsymbol{W}/\partial \boldsymbol{R}]$. Therefore, the adjoint vector can be helpful to establish the solution error in evaluating the objective function. This is, most of the time, due to poor mesh resolution, thus it has found many applications in mesh-refinement solution techniques [32]. Based on Eq. (5.35), Fig. 5.8 plots exactly this vector which maps where and to which flow variable the error in evaluating the drag (the same thing holds for any other objective function) should be appointed to. Fig. 5.8 shows poor mesh resolution on colours tending to red or blue, whereas colours tending to green show sufficiently resolved mesh. It is no surprise that, regions that need to be refined are those that experience a high pressure gradient such as close to the wall and the shock. Furthermore, what it is interesting is that this map highlights areas that are not intuitive, such as that located forward the leading edge.

For instance, let us consider a positive change in the *x*-momentum residual, the effect of this change on the drag coefficient (objective function) is represented by the respective adjoint variable and its contour is depicted in Fig. 5.8 (top). This means that the drag increases where the adjoint variable is also positive (colours tending to red), whereas drag decreases where the adjoint variable is negative (colours tending to blue). Therefore, these plots highlight the areas that need to be a-posteriori mesh-refined in order to get a solution no longer dependent from the flow residual or in other words, a mesh-converged solution [141, 34]. This shows one type of a-posteriori mesh refinement. However, there are many other methods such as a priori knowledge and a posteriori pressure-gradient mesh refinement as discussed in Harris and Qin [142].

$$\{\boldsymbol{\Lambda}_{F,2}\} = \left\{\frac{\partial C_D}{\partial \boldsymbol{R}_2}\right\} \rightarrow x\ momentum$$



$$\{\boldsymbol{\Lambda}_{F,4}\} = \left\{\frac{\partial C_D}{\partial \boldsymbol{R}_4}\right\} \rightarrow z\ momentum$$

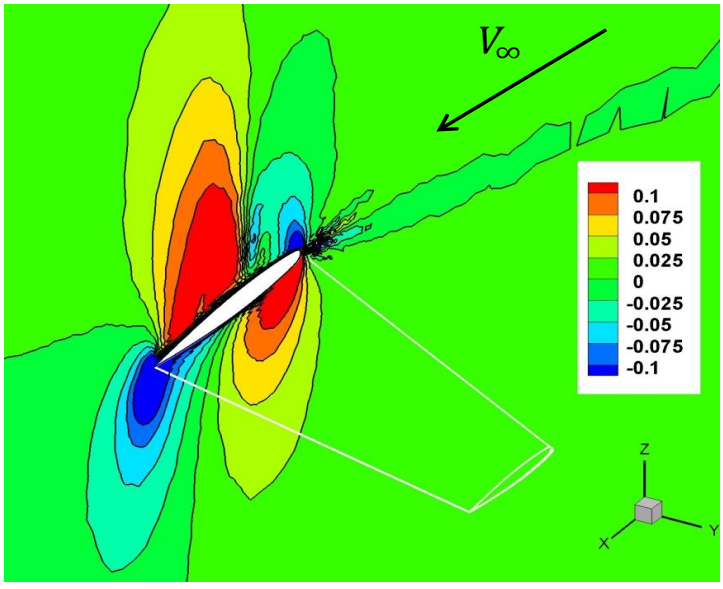


Figure 5.8 Cut-views in the ONERA M6 computational domain of two flow-adjoint variables.

5.7.2 Sensitivity of the Objective Function w.r.t. the Volume Mesh

The sensitivity of the objective function w.r.t. the volume mesh includes both a pressure (subscript p), and a viscous component (subscript v):

$$\left\{\frac{dI}{d\boldsymbol{X}}\right\} = \left\{\frac{\partial I}{\partial \boldsymbol{X}} + \boldsymbol{\Lambda}_F^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right\} = \left\{\frac{\partial}{\partial \boldsymbol{X}}(I_p + I_v) + \boldsymbol{\Lambda}_F^T \frac{\partial}{\partial \boldsymbol{X}}(\boldsymbol{R}_p + \boldsymbol{R}_v)\right\} \tag{5.36}$$

To better understand the non trivial physical meaning of these terms, each partial derivative is analysed separately. The first RHS term expresses the variation of the target function due to the variation of the surface (line in 2D) of integration at constant pressure and skin friction coefficient [32]. The second RHS term expresses the variation of the target function due to the variation of the pressure and skin friction coefficient, both induced by a variation of the geometry [32].

The next step is to analyse its distribution over the computational domain. This is important because it allows one to represent graphically where and if there is any cancelling effect while performing the product expressed in Eq. (5.14). Widhalm *et al.* [36] noted that the gradient $\{dI/d\boldsymbol{S}\}$ is influenced mainly by values of $\{dI/d\boldsymbol{X}\}$ at the surface for the inviscid (Euler solution) cases and by values of $\{dI/d\boldsymbol{X}\}$ at the very first (close to the wall) mesh layers for viscous (Navier-Stokes solution) cases. Therefore, it is interesting to see to what extent vector $\{dI/d\boldsymbol{X}\}$, if only its surface values are considered, i.e. $\{dI/d\boldsymbol{X}\}_{surface}$, depicted in Fig. 5.9 (left), compares to vector $\{dI/d\boldsymbol{S}\}$ depicted in Fig. 5.9 (right). As it can be seen, some similarities such as the lambda shock boundaries can be distinguished, but the final gradient is fairly different. This is particularly true in the area enclosed within the lambda shock boundaries.

Furthermore, Widhalm *et al.* [36] investigated also the effect of both viscous and pressure terms on the gradient $\{dI/d\boldsymbol{X}\}$. They conclude that for the subsonic flow regime the gradient $\{dI/d\boldsymbol{S}\}$ is mainly influenced by the viscous part of Eq. (5.36), whereas for the transonic flow regime the gradient $\{dI/d\boldsymbol{S}\}$ is mainly influenced by the pressure part of Eq. (5.36).

To further investigate this, the spatial position of all volume mesh nodes that carry a value greater than two pre-determined small thresholds are depicted in Fig. 5.10 (to help visualise

them, the quantities are depicted within different y/c bounds). As can be seen, points that carry non-negligible values are not in large number and are mostly located very close to the wall.

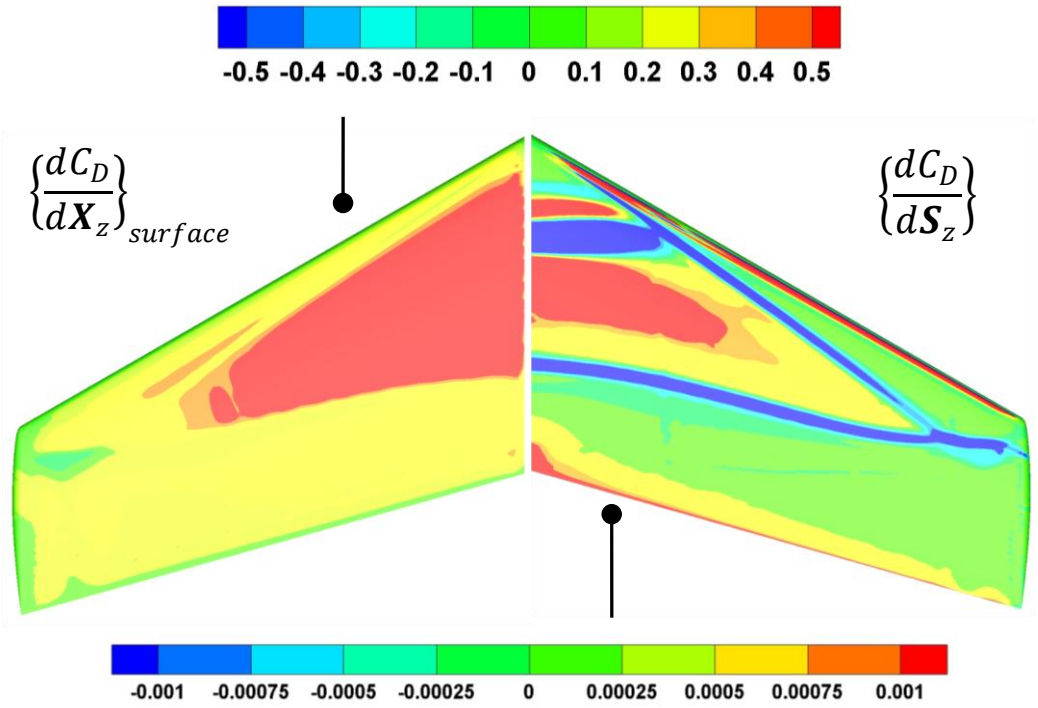


Figure 5.9 Comparison between the values at the wall for two different types of gradient.

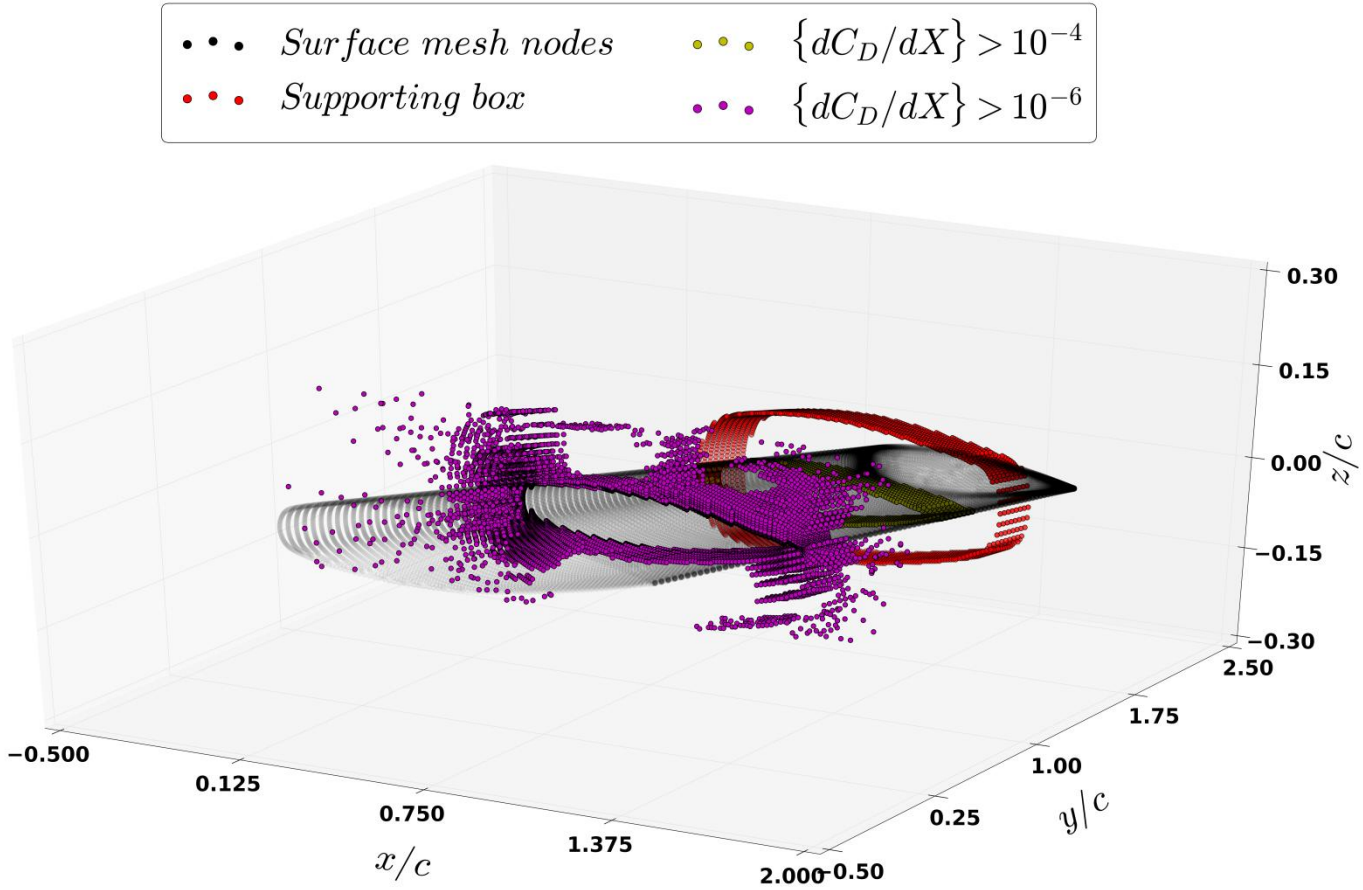


Figure 5.10 Extent of the information stored in the gradient for values greater than two thresholds.

5.7.3 Sensitivity of the Objective Function w.r.t. the Surface Mesh

The objective function is defined through a surface (line in 2D) integral and is computed accordingly to NFA (Near-Field Approach) where both the pressure, $C_{D,p}$, and the viscous component, $C_{D,v}$, are taken into account:

$$C_{D,NFA} = C_{D,p} + C_{D,v} \tag{5.37}$$

As a consequence, any change in the objective function comes from a mix of pressure and shear stress contributions. It is then reasonable to ask if the resulting gradient map, i.e. $\{dI/d\boldsymbol{S}\}$, reflects the pressure and the skin friction coefficient trend, although, as described in Chap. 5.7.2, at transonic speed the influence of the pressure component is larger than the viscous one. The link between the two is not trivial as it may seem, and as it is shown later, it may lead to target non-intuitive areas as successfully done by Hinchliffe and Qin [46].

Having said that, it is important to highlight why a common trend is being investigated. This stems from the common practice in aerodynamic shape optimisation to justify the high sensitivity at the shock, leading and trailing edge by comparing them to the pressure and skin friction coefficient plots. In fact, referring to Figs. 5.11 and 5.12 (point values are read directly from the unstructured surface mesh from the sections sampled as depicted in Fig. 5.21 (left), thus this explains the gap between the sampled points), it can be clearly seen that there are some similarities between these two coefficients and the gradient at the leading, trailing edge and shock area. However, a rapid change in the sensitivity map is not always reflected in a rapid change in the pressure or in the skin friction plot. This is evident in the forward wing area between the two (lambda footprint) shocks.

This brings the question whether this may be the result of some numerical by-product or not. Hinchliffe and Qin [46], showed through a numerical optimisation, how areas not highlighted by the pressure or the skin friction plot can be exploited by the optimiser proving that the target area is indeed not a numerical artefact. This is the proof that in certain case, the surface map can highlight some non intuitive areas where an improvement can be made.

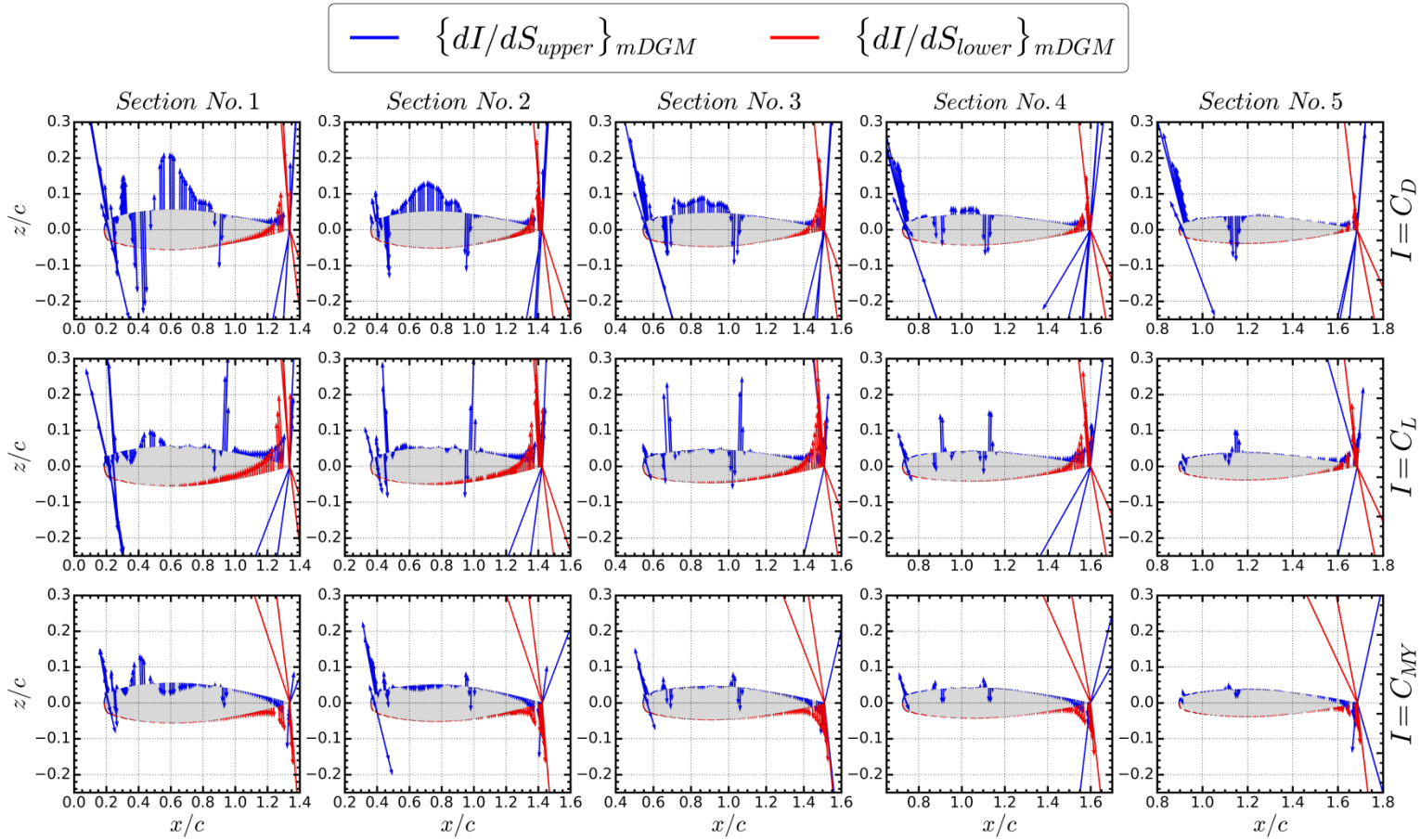


Figure 5.11 2D gradients comparison at different stations.

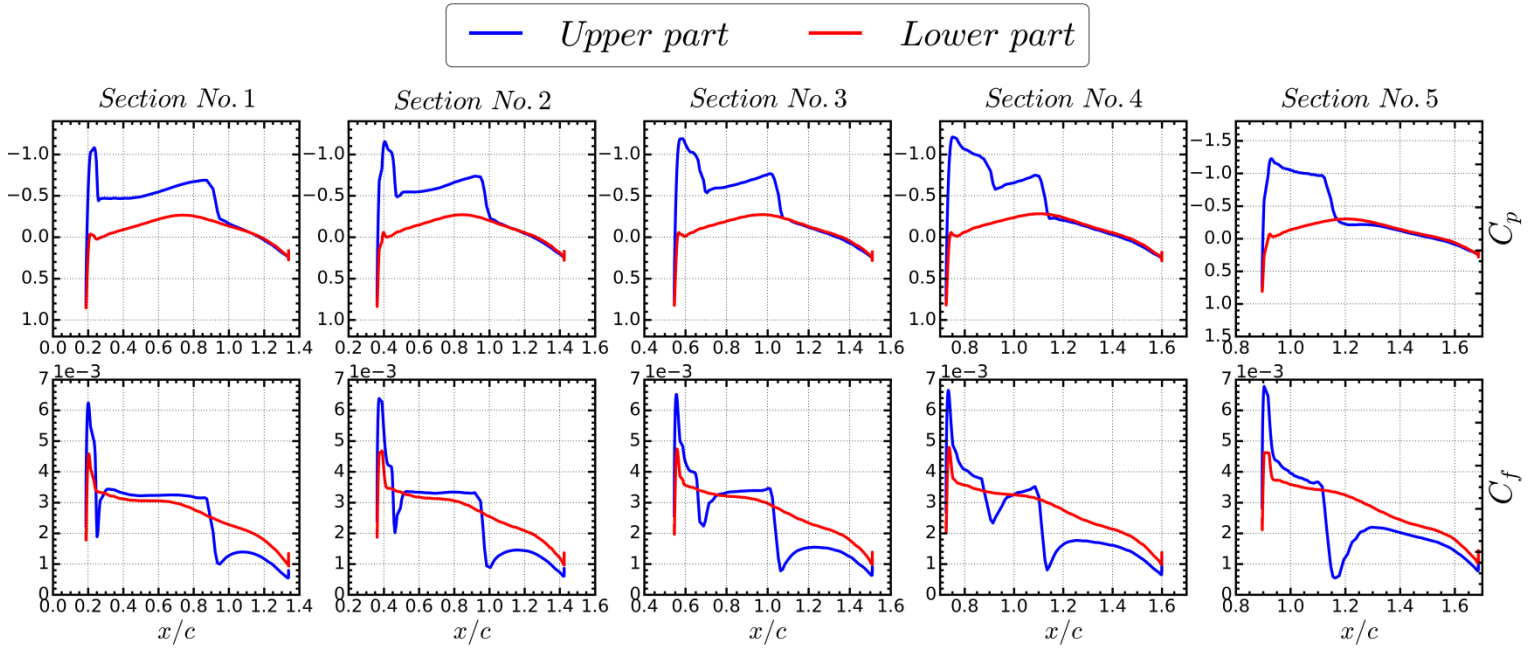


Figure 5.12 Pressure and skin friction coefficients at different stations.

5.7.4 Observation on the Product

There are two quantities involved in the product reported in Eq. (5.14). The first is the gradient of the objective function w.r.t. the volume mesh nodes, whereas the second is the Jacobian of the volume mesh w.r.t. the surface mesh nodes. The former, contrary to the latter, is the only one which carries physical information represented by the objective function. The goal of this sub-chapter is to check that the volumetric mesh sensitivity, i.e. $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is not taking null values when the value of $\{dI/d\boldsymbol{X}\}$ are not null.

To study the distribution over the entire computational domain of these two factors, it is necessary to find a common metric for the comparison. One is a vector, whereas the other is a matrix, but they can be both visualised on the volume mesh. However, working on the entire mesh is difficult because plotting its characteristic everywhere in the domain is not practical. One possible solution is to select a small area at the surface and then consider all the points in the volume mesh that lays on a small neighbourhood along its normal. Since the mesh is unstructured, it is not possible to select a single mesh line running from the solid wall to the far-field, thus a small neighbourhood at the wall (as shown in Fig. 5.13) needs to be selected in order to make sure that the points within a small neighbourhood (around the normal) cover the entire domain.

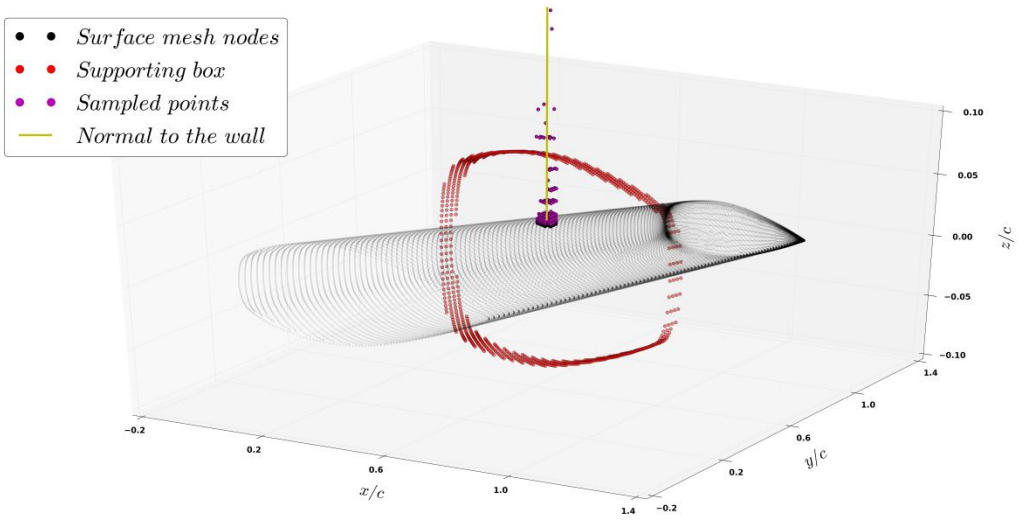
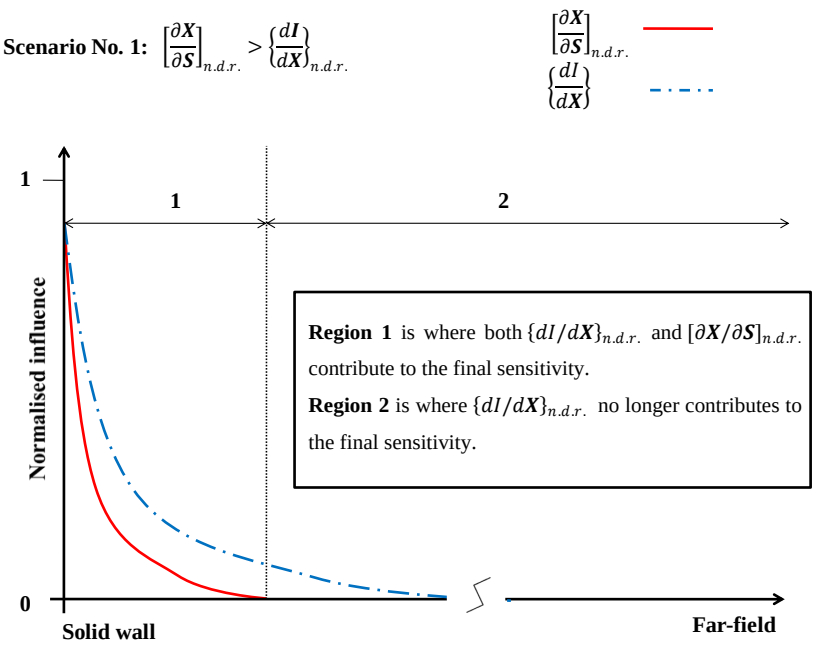


Figure 5.13 Isometric view of the selected point along an imaginary line normal to the wall.

This means it is possible to describe their distributions along an imaginary line drawn normal from the wall to the farfield. From here, the definition of n.d.r. (normal decay rate). There are many possible distributions, however, for the sake of simplicity, a preliminary isotropic and exponential decay rate along the normal to the wall was assumed. By doing so, three possible scenarios can be distinguished:

- Scenario No. 1 is depicted in Fig. 5.14 (top). It shows the case where the decay rate of the matrix $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is greater than that of the gradient $\{dI/d\boldsymbol{X}\}$. In this case, when the former reaches values close to zero much earlier than the latter, the influence of the gradient $\{dI/d\boldsymbol{X}\}$ is nullified earlier than its decay rate would impose.
- Scenario No. 2 is depicted in Fig. 5.14 (bottom). It shows the case where the decay rate of the matrix $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is smaller than that of the gradient $\{dI/d\boldsymbol{X}\}$. In this case, when the latter reaches values close to zero much faster than the former, the influence of the matrix $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is nullified earlier than its decay rate would impose.
- Scenario No. 3 is when $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ and $\{dI/d\boldsymbol{X}\}$ have the same decay rate and therefore no cancelling effect is present.

The first two cases describe the possibility of one of the factors offsetting the information carried by the other. It is obvious that, the first is the worst case scenario as the non-physical quantity, i.e. matrix $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$, is cutting off the physical information carried by $\{dI/d\boldsymbol{X}\}$. In order to numerically investigate this, the ONERA M6 wing described in Chap. 5.4 is considered. Referring to Fig. 5.15, a semi-log scale was used in order to highlight the relative importance of quantities that have their values distributed over different orders of magnitude. Fig. 5.15 clearly shows that, regardless to the mesh movement used, the n.d.r. of the Jacobians decays at a rate which is much smaller than the n.d.r. registered for the gradient $\{dI/d\boldsymbol{X}\}$.



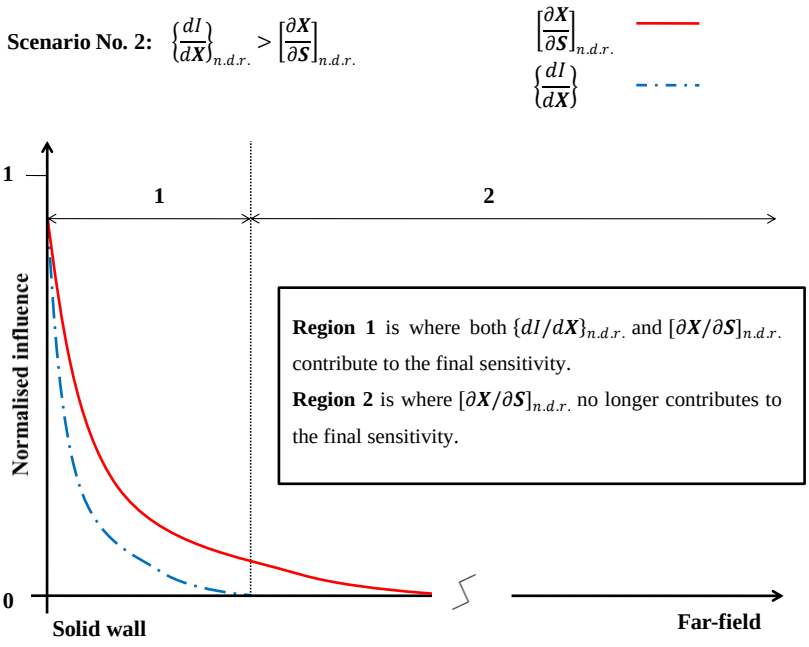


Figure 5.14 Two likely possible scenarios while analysing the n.d.r. of the information carried by the metric terms.

This places the ONERA M6 in scenario No. 2 where the physical information is not actually nullified by the non-physical grid sensitivity. It is the author's opinion that this is always the case because it is preferable to distribute the deformations as evenly as possible over the entire mesh in order to reduce the risk of mesh element cross over. In fact, having a slow decay rate of the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ is an indication of the capability to spread the deformations away from the wall.

This study served two purposes. Firstly, it was established that a mesh movement that is too stiff may become a problem in its differentiated form because it could cut-off physical information, and secondly that the differentiated mesh movement does not need to include the entire computational domain because extending the differentiation to the mesh nodes away from the wall does not contribute to the gradient since the physical information (that has a greater n.d.r.) is already taking null values.

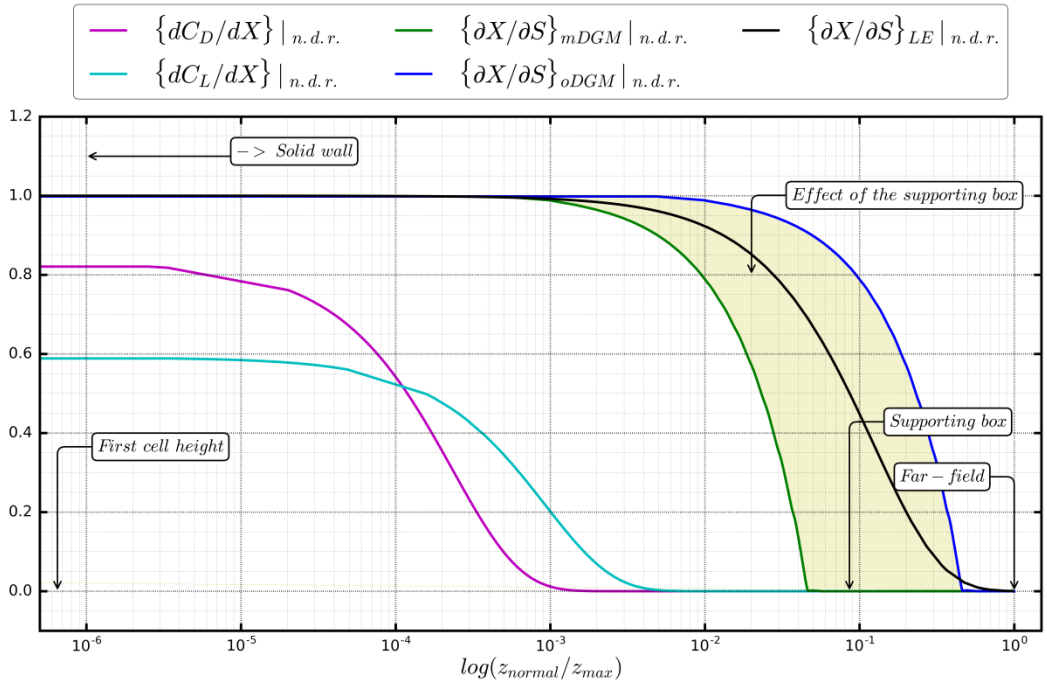


Figure 5.15 Comparison between different distributions on the computational domain.

5.8 Gradient Chain Performance Assessment

The assessment of the performance is done on three levels. The first one compares the gradient obtained using the mDGM and the LE-based chain, the second compares the CPU time and the memory requirements to deliver the gradient against the LE, and the third studies how the mDGM-based gradient chain requirements scale with the mesh size.

5.8.1 Gradients Comparison

This section compares two gradients which describe the variation of the same objective function w.r.t. a movement of the surface mesh nodes. However, these gradients are effectively computed using two different differentiated mesh movements. It is desired to investigate whether they are equal or different. Should they be different, it is desired to establish how much and where the deviations occur.

Two geometries are chosen: the first geometry is the ONERA M6 [133] (see Fig. 5.3), whereas the second one is the DLR F6 wing [143] at flow conditions of Mach number of 0.8 and constant lift coefficient of 0.6 solved on a quad-dominant mesh consisting of roughly 2.7 Mil. volume mesh nodes. Fig. 5.16 shows the surface mesh (top), the pressure coefficient (centre) and the y^+ (bottom). For these geometries, the gradients of three objective functions, drag, lift and longitudinal moment coefficient w.r.t. the surface are shown in Figs. 5.17-5.20 using a series of contour plots. Although, these are very effective in showing the general trend, they are not ideal to show the small differences in the gradients because of the interpolation procedure they are based upon. For this reason, in order to visualise even the smallest differences, a series of spanwise cuts are sampled as shown in Fig. 5.21 and the resulting 2D vectors are shown in Figs. 5.22 and 5.23.

For both geometries the series of contour plots shown in Figs. 5.17-5.20 associated to each target function do not show any particular deviation that can be captured by the naked eye. However, the 2D vectors depicted in Figs. 5.22 and 5.23 (the scale is different for each sub-graph in order to make all the trends visible and the scale on the z-axis is true to the profile coordinates only), confirm that the trend is indeed equally described by the two methods, but large deviations are registered at the trailing edge area and where the gradients change sign.

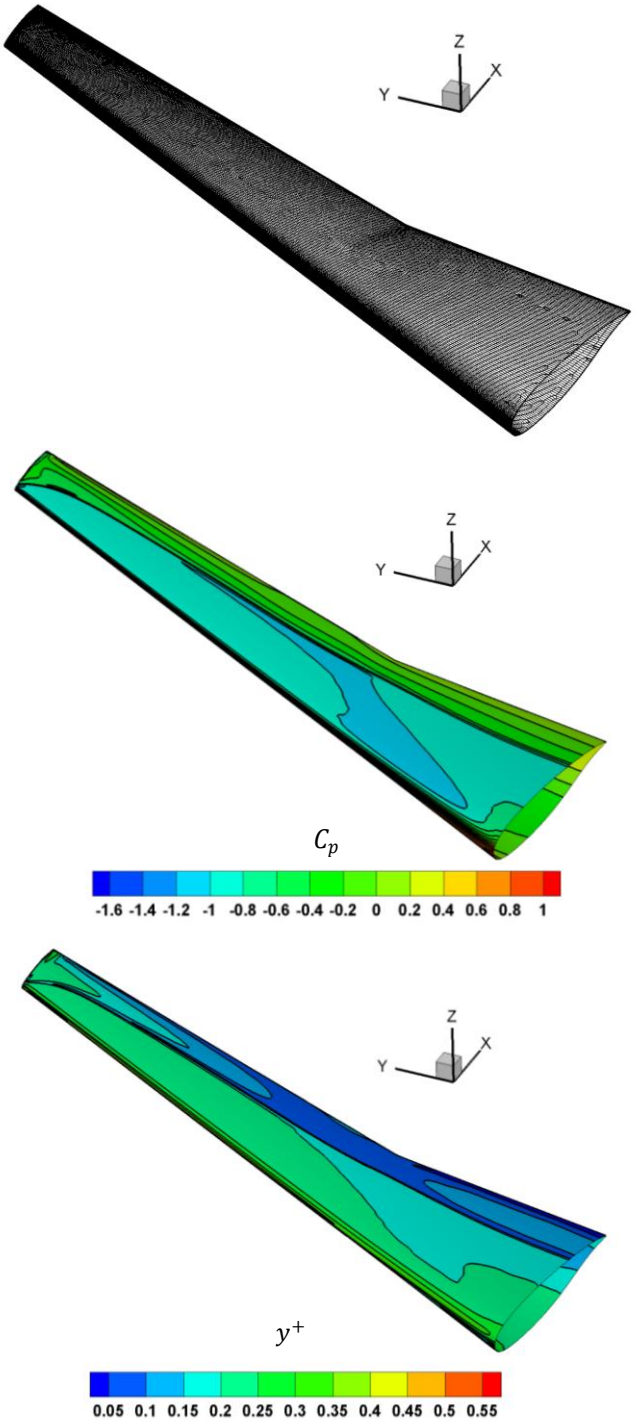


Figure 5.16 Unstructured surface mesh (top), pressure contours (centre) and y-plus contours for the DLR F6 wing.

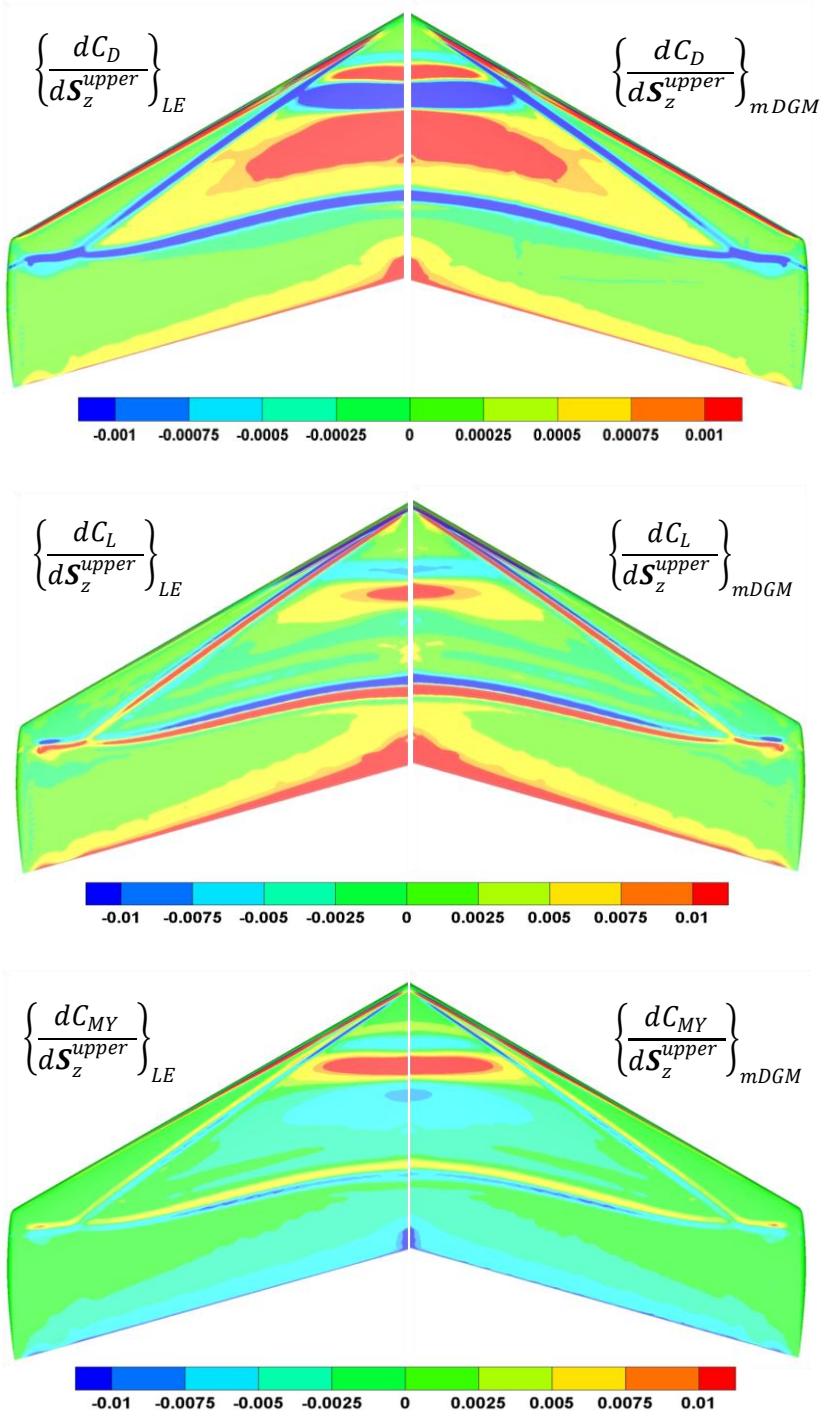


Figure 5.17 Upper gradient comparison for the ONERA M6 wing.

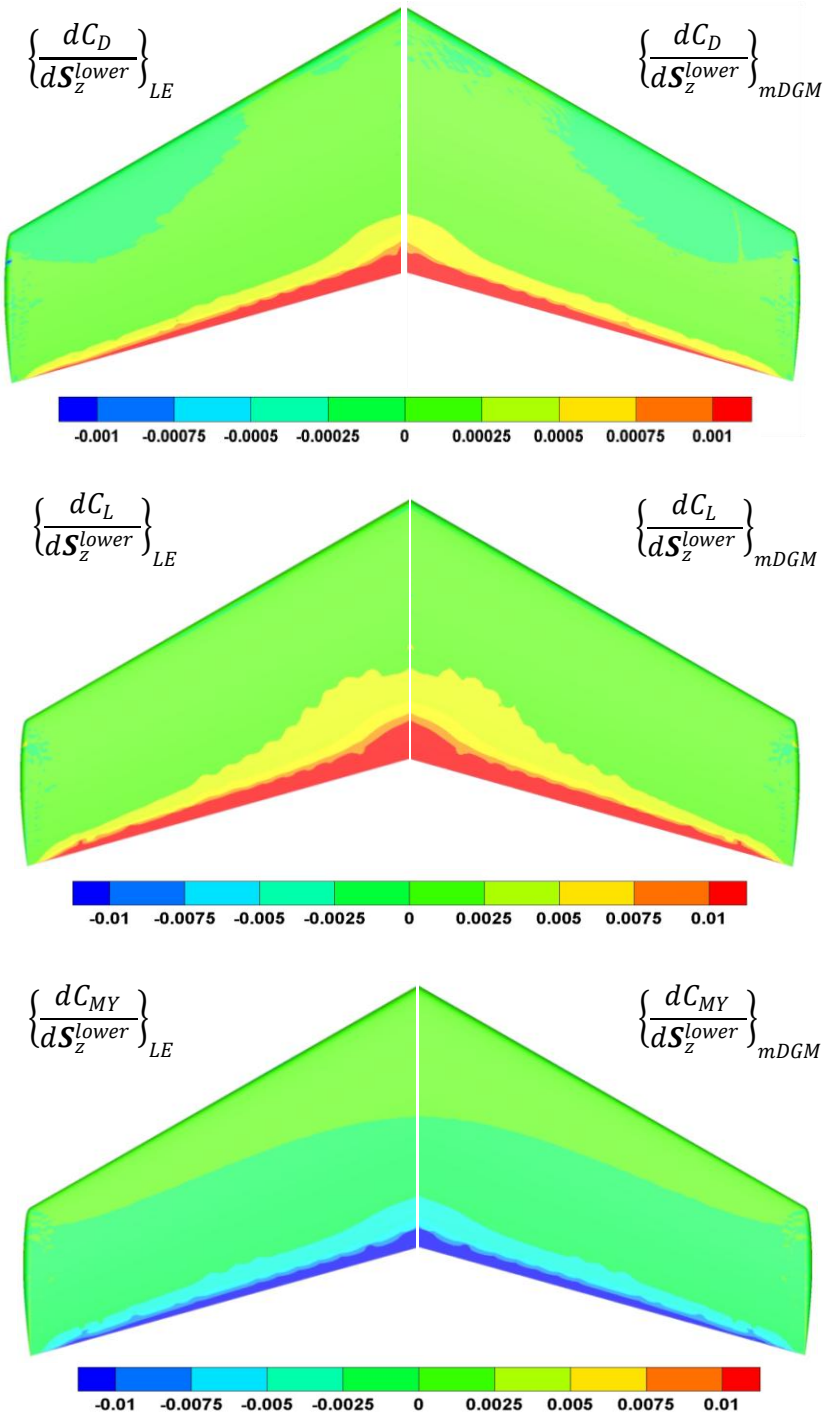
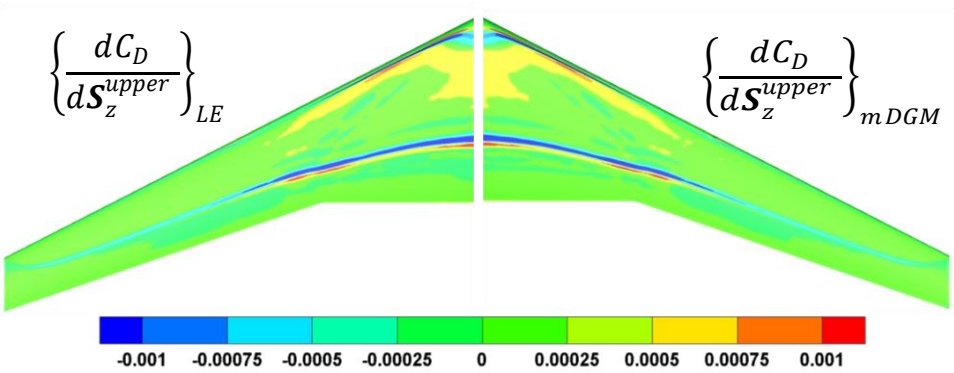
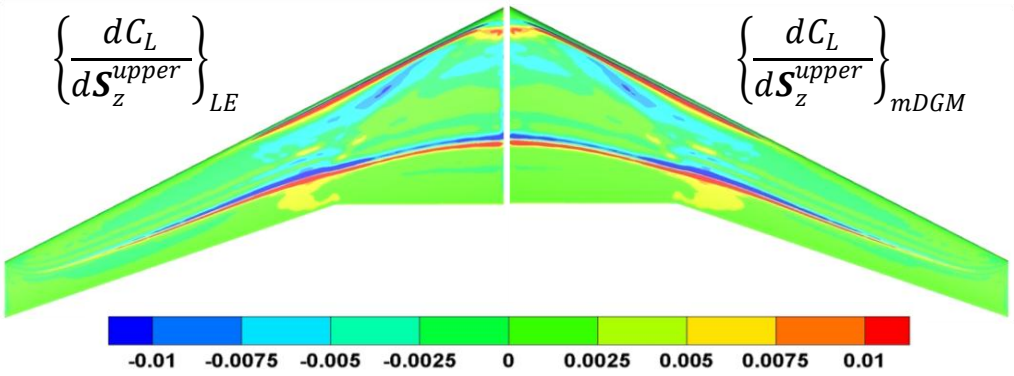


Figure 5.18 Lower gradient comparison for the ONERA M6 wing.





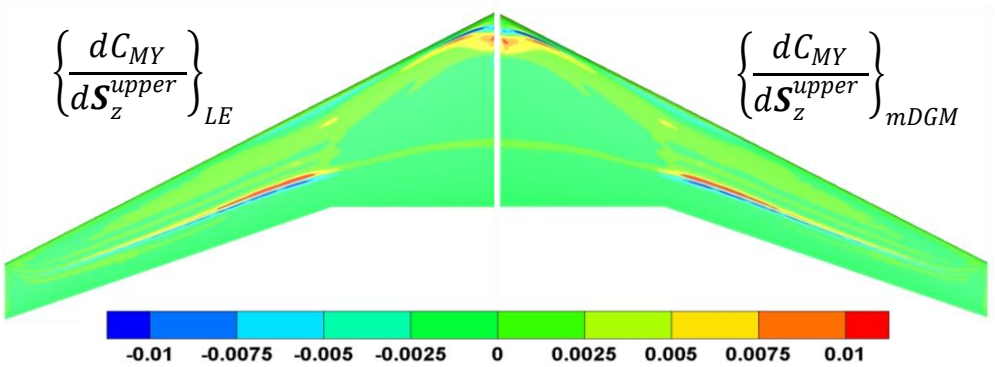


Figure 5.19 Upper gradient comparison for the DLR F6 wing.

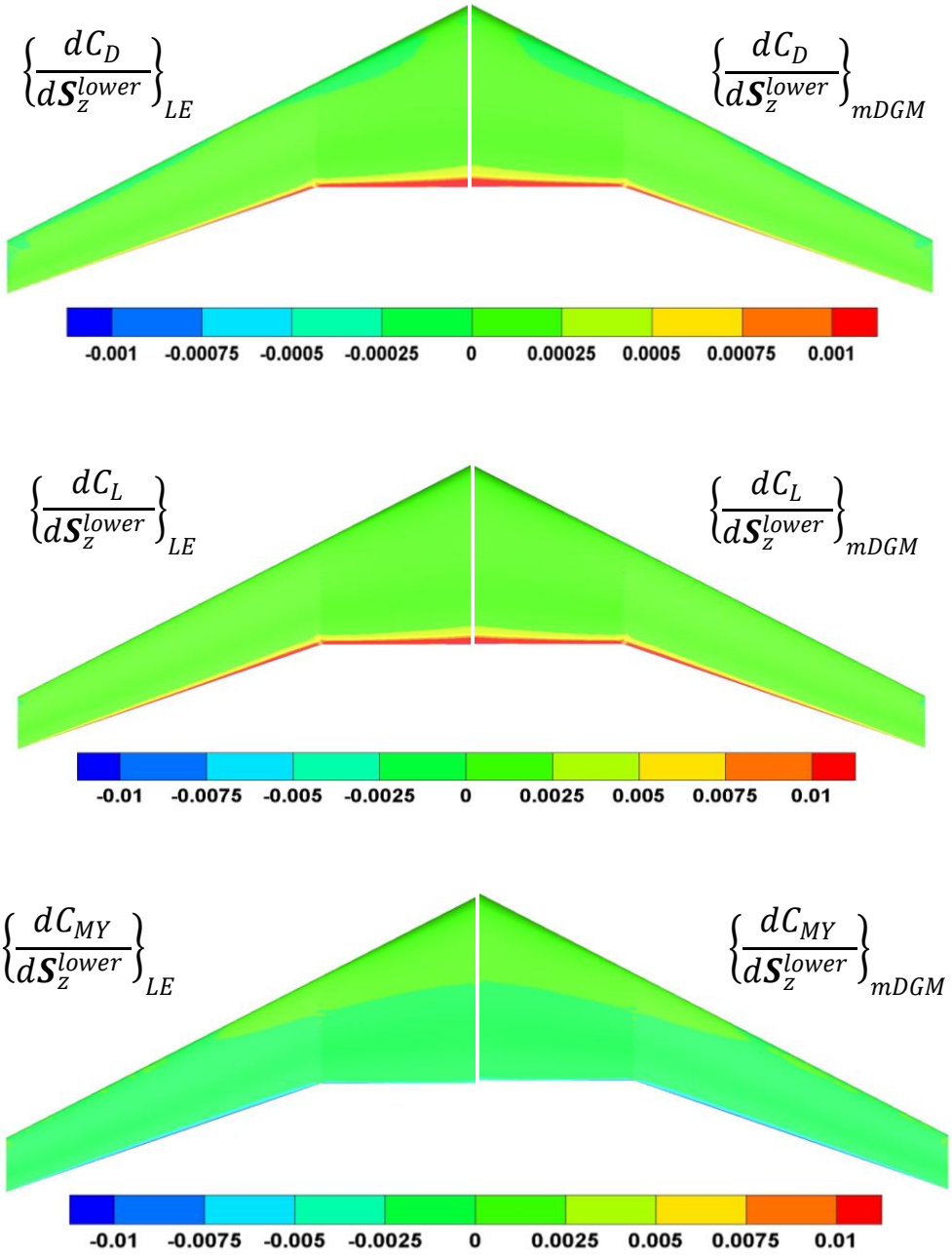


Figure 5.20 Lower gradient comparison for the DLR F6 wing.

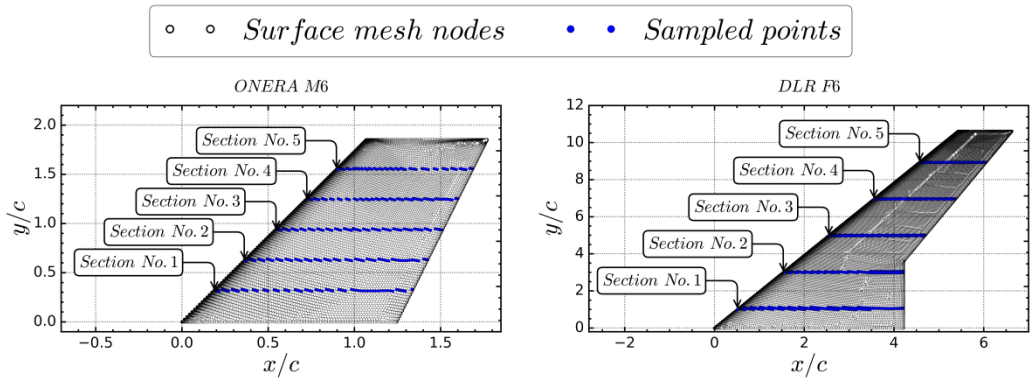


Figure 5.21 Selected points for the 2D gradient vectors comparison.

This following discussion tries to explain why, although two different mesh movements are used, the general gradients trend is surprisingly not too different. The proposed argument follows the flowchart depicted in Fig. 5.24. Firstly, the Jacobian $[\partial \boldsymbol{S}/\partial \boldsymbol{D}]$ is not considered because it does not depend on the mesh movement. Secondly, gradient $\{dI/d\boldsymbol{X}\}$ is the only part of the chain that carries physical information which depends on the flow adjoint and the flow solution. Having said that, this gradient is the same regardless to the chosen mesh movement. As a consequence, the first thing that could be considered for the deviations in the gradient is the fact that LE elastic matrix needs to be inverted, whereas the DGM provides the Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ directly. Therefore, the DGM-based Jacobian could be taken as an example to match, but this is conceptually wrong because the two methods are mathematically different and thus they are not expected to deliver the same Jacobian. The next step is to actually study both Jacobian matrices $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE,mDGM}$. It is desired to study if they are different, to what degree and where in particular. As can be seen from Fig. 5.15, the distribution of these two Jacobians is very similar close to the wall where gradient $\{dI/d\boldsymbol{X}\}$ is different from zero. However, just before this vector reaches values tending to zero, the two Jacobian distributions start to diverge. Since the area where the two Jacobians are equal covers most of the area where gradient $\{dI/d\boldsymbol{X}\}$ is different from zero, but differs in a small relative region, this explains why they are largely equal but still different in some details.

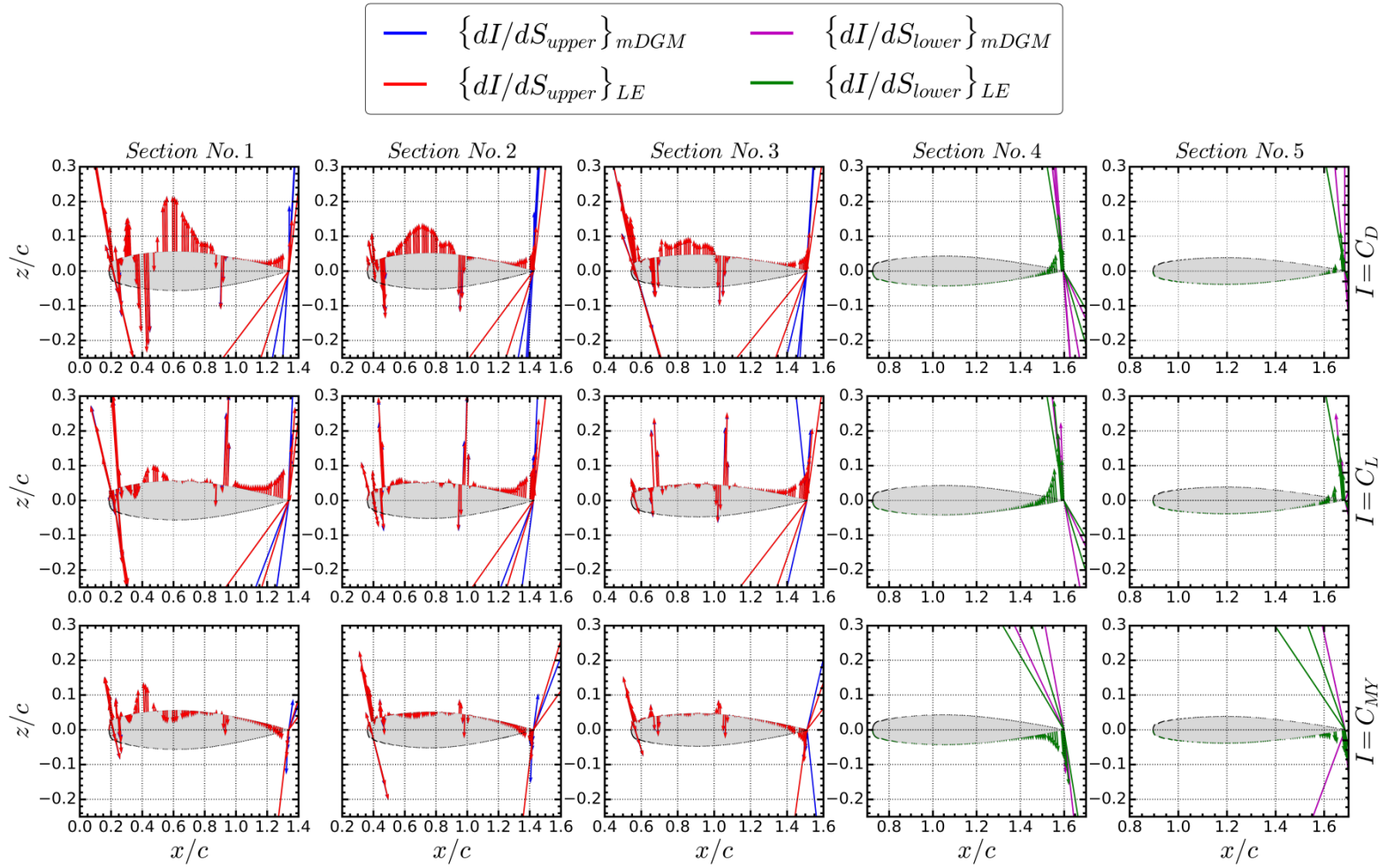


Figure 5.22 Point-to-point gradient comparison for the ONERA M6 wing.

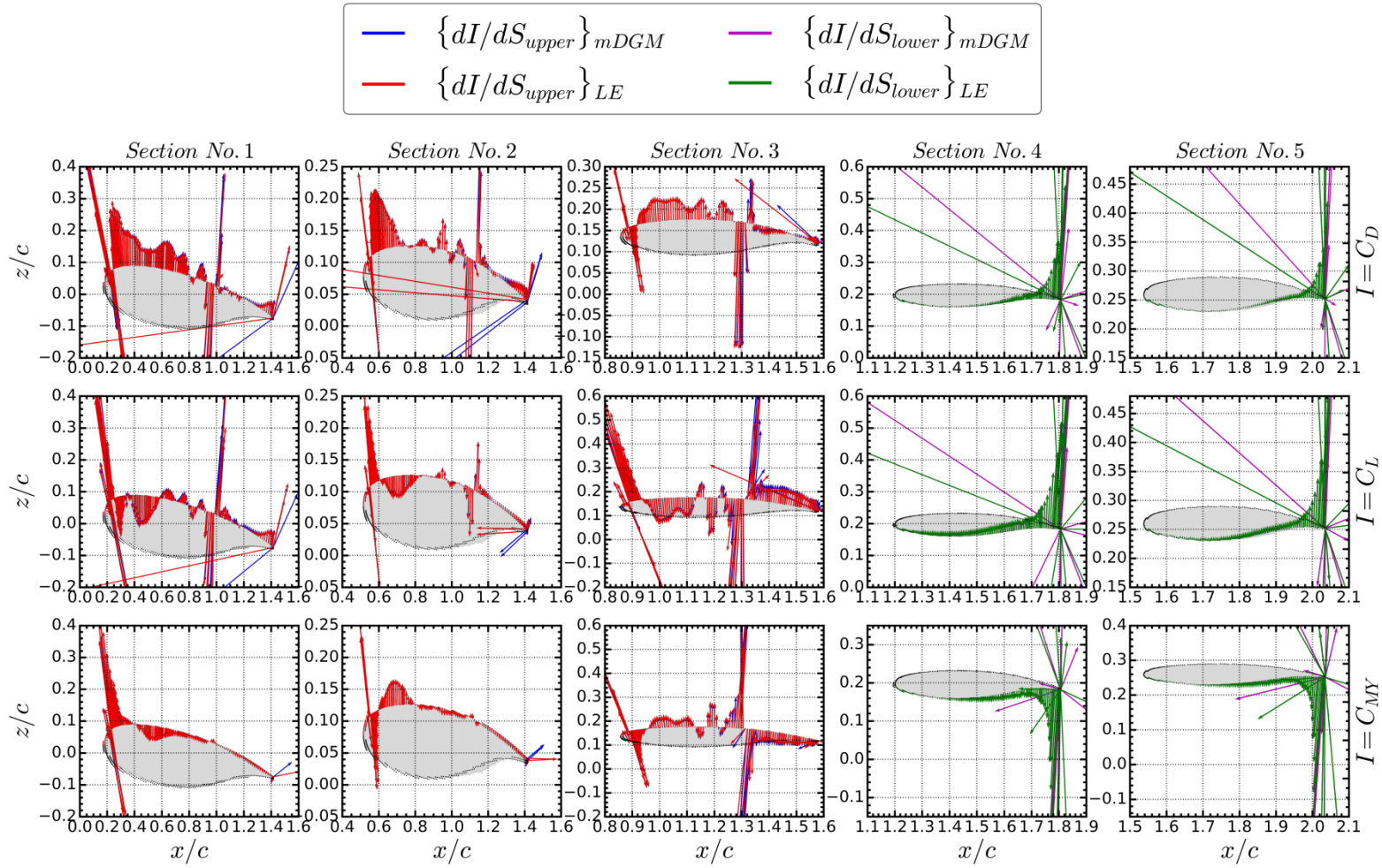


Figure 5.23 Point-to-point gradient comparison for the DLR F6 wing.

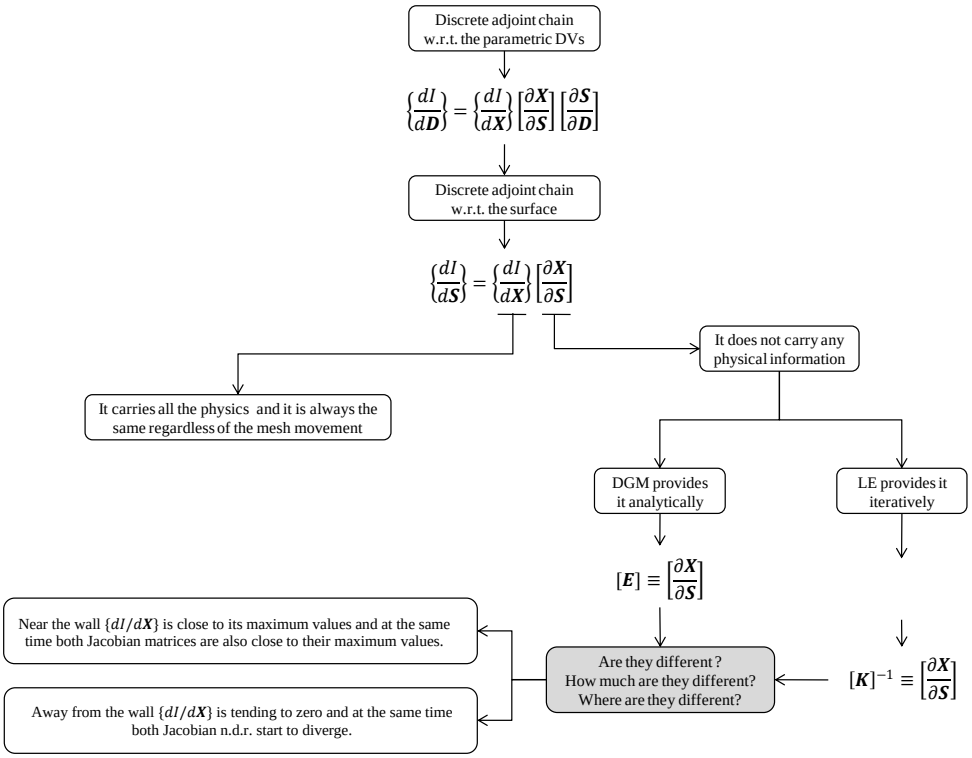


Figure 5.24 Scheme of the analysis used for the gradients comparison.

A similar study was published also by Wang *et al.* [51] where the authors investigated the influence of different mesh movements such as RBF, spring and LE analogy on the resulting gradient. Their study concentrated directly on the sensitivity w.r.t. the parametric design variables, i.e. $\{dI/d\boldsymbol{D}\}$, whereas in the present work, the attention was on $\{dI/d\boldsymbol{S}\}$. Nevertheless, the study published by Wang *et al.* [51] confirms the conclusion made above. In other words, even though the mesh movements are different, their grid sensitivities lead to the same value of the gradient.

5.8.2 CPU Time and Memory Comparisons

In this chapter the differentiated mesh movement performances of two methods, namely LE and mDGM are discussed. This is assessed by monitoring the CPU time and the memory requirements, while computing the gradient expressed as in $\{dI/d\boldsymbol{S}\}$. The initial discussion presented in Chap. 4.7.1 on how the comparison was built still holds here.

It was already mentioned that the LE and mDGM are different in their formulation and finding a common ground where they can be both fairly tested is not easy. These two methods compute the $\{dI/d\boldsymbol{S}\}$ map in different steps. When using the mDGM, matrix $[\boldsymbol{E}]$ and vector $\{dI/d\boldsymbol{X}\}$ are first computed separately and then multiplied together. Therefore, the CPU time reported in Tab. 5.3 is the sum of the time required to perform these two operations. On the other hand, although the LE makes use of vector $\{dI/d\boldsymbol{X}\}$, this is not saved on disk, but its entries are computed recursively as the system of equations (see Eq. (5.31)) is solved iteratively.

From the results reported in Tab. 5.3, three conclusions can be made regarding the memory, CPU time and disparity registered in these values for different objective functions.

Table 5.3 CPU time and memory requirements comparison between the differentiated mDGM and LE.

Differentiated mesh movement	**Geometry**	**Objective function**	**Memory (Gb)**	**CPU time (s)**
mDGM	M6 wing	$C_{D,L,MY}$	1.2	179 (=130+49)
	F6 wing	$C_{D,L,MY}$	2	337 (=255+82)
LE	M6 wing	C_D	15	695
		C_L		1,031
		C_{MY}		839
	F6 wing	C_D	27	1,655
		C_L		1,759
		C_{MY}		1,594

In particular, the memory needed to solve the LE-based linear system of equations is one order of magnitude bigger than that requested by the mDGM. A similar trend is registered also in terms of CPU time. For instance, considering the DLR F6 wing case, the mDGM takes 2Gb of memory and 82s of CPU time to perform the product. Interestingly, the largest share of the total CPU time is not represented by the product itself, but by the computation of vector $\{dI/d\boldsymbol{X}\}$

which takes 255s. The same map is delivered by the LE in about 1,669s (mean between the three objective functions) and 27Gb of memory. This makes the LE-based gradient approach 4.95 times in terms of CPU time and 13.5 times in terms of memory more expensive than the mDGM. Similar conclusions can be made for the ONERA M6 wing.

Three objective functions were tested, namely drag, lift and longitudinal moment coefficient. What is interesting to note is that both mDGM and LE require the same memory as the objective function is changed. On the other hand, the LE-based gradient shows CPU time values which depend on the choice of the objective function. Since the same solution algorithm is used, one explanation for the disparities in CPU times, is that the memory is proportional to the size of the system, i.e. proportional to $N_{VM} \times N_{SM}$ which is the same for all the objective functions, whereas the CPU time is proportional to the linear system stiffness (conditioning) which cannot be considered constant as one changes the objective function.

5.8.3 Performance Scaling with the Mesh Size

In order to make a final judgment on the method, it is interesting to see how the mDGM scales with the N_{VM}. To this end, the DLR F6 wing-body with the FX2B bump fairing geometry is chosen [131]. The supporting box used can be seen in Fig. 4.13 and was constructed selecting some suitable volume mesh points as described in Chap. 4.5.3. The final sensitivities can instead be seen in Fig. 5.25 and it was computed accordingly to Eq. (5.12). The flow solution was computed at Mach number of 0.75 and target lift coefficient of 0.5 with the flow settings described in Chap. 3.6.

From the result reported in Tab. 5.4, it is evident that the increase on the mesh size is very similar to the increase in the memory required. In fact, considering the medium mesh, with a N_{VM} equal to 6.1 Mil., the increase w.r.t. the coarse mesh is of 110.34% which is then reflected in an increase in memory of 109.52%. However, the same trend is not registered for the CPU time where for instance given a delta in the N_{VM} of +110.34% and +244.82% this is reflected in a variation of CPU time of +163.78% and +378.30% respectively.

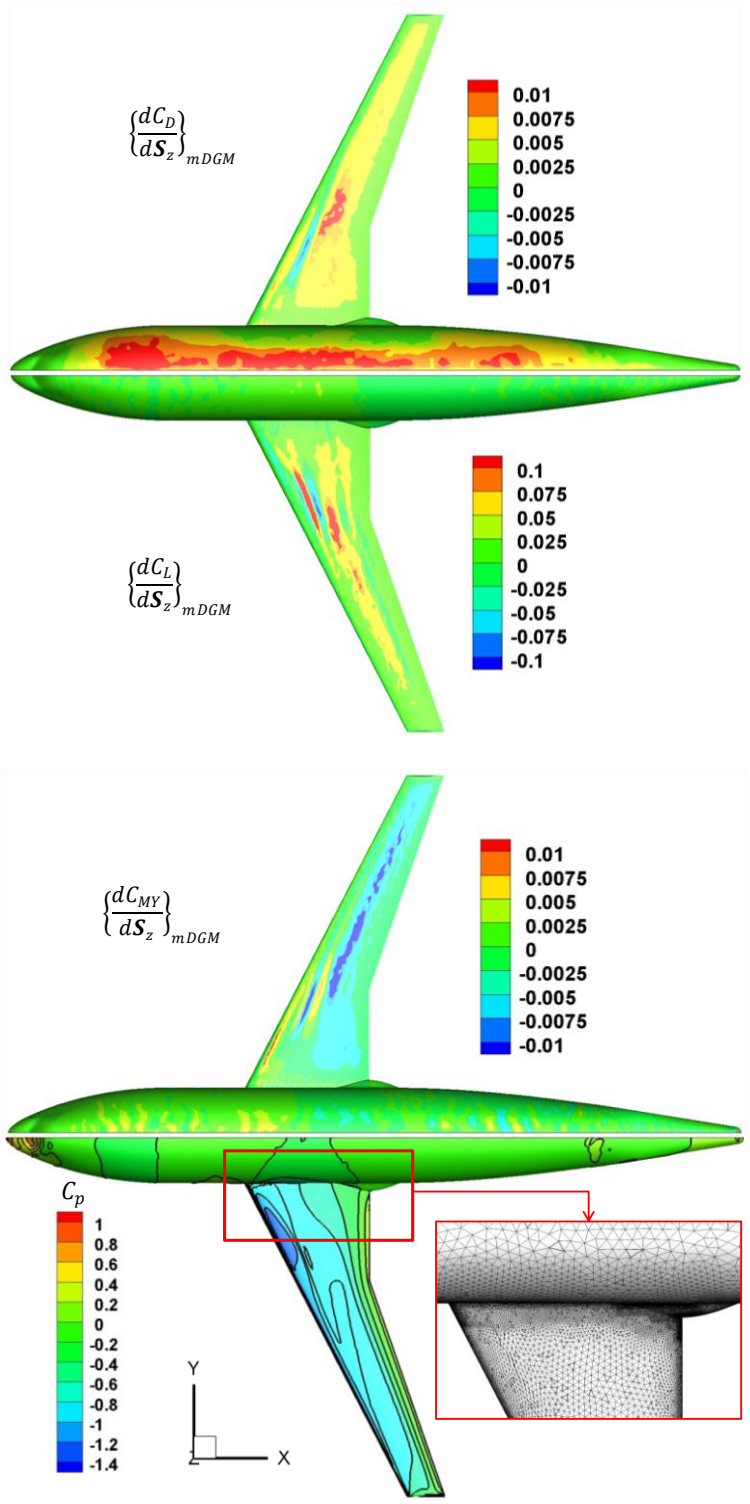


Figure 5.25 Different types of gradients for the DLR F6 wing-body geometry.

Table 5.4 mDGM CPU time and memory requirements scaling with the mesh size.

N_{VM} **(Mil)**	Δ%	**Memory (Gb)**	Δ%	**Objective functions**	**CPU time (s)**	Δ%
2.9	Datum	2.1	Datum	$C_{D,L,MY}$	544	Datum
6.1	+110.34	4.4	+109.52		1,435	+163.78
10	+244.82	7.4	+252.38		2,602	+378.30

However, care should be exercised while commenting these data. The CPU time reported in Tab. 5.4 is the sum of the CPU time required to perform two steps, namely, computation of $\{dI/d\boldsymbol{X}\}$ and $\{dI/d\boldsymbol{X}\}[\boldsymbol{E}]$. Since the computation of vector $\{dI/d\boldsymbol{X}\}$ does not depend on the mesh movement chosen, to truly understand the benefit provided by an algebraic method such as the mDGM one should consider only the CPU time require to perform the product. In this case, the CPU time is much lower. For the three grids tested in Tab. 5.4 given the total CPU time of 544s, 1,435s, 2,602s only 90s, 185s, 317s are actually needed to perform the product and the remaining time is actually used to compute $\{dI/d\boldsymbol{X}\}$.

5.9 Summary

A method based on the DGM mesh movement was proposed to efficiently compute the grid sensitivity in the discrete adjoint framework. The method results in a one-to-one explicit algebraic mapping between the volume and solid wall mesh nodes. This allows a straightforward computation of the gradient without the need to invert a large and sparse matrix or in other words, to converge a linear system of equations generally associated with any implicit iterative mesh movement such as LE.

Two strategies were presented to derive the DGM-based gradient chain. While using the augmented objective function approach, it was proven how the solution of the second linear mesh adjoint system of equations can be eliminated. The same result was shown to be possible even if the Lagrangian multiplier associated with the mesh movement constraint is not used. To

be more precise, this second option can be used only if the differentiated mesh movement provides $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$ explicitly.

The method was verified for the 2D RAE 5243 aerofoil and for the 3D ONERA M6 wing for two cases: sensitivity of the objective function w.r.t. each surface mesh point and the AoA compared against the FDs and linearised values respectively.

The issue related to the DGM-based gradient was addressed and discussed. A solution was proposed which consists in the construction of a supporting box. This improves the quality of the Delaunay elements and in turn respects the surface-to-volume map established by Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$.

A comparison against the LE in terms of memory and CPU time requirements was also provided, where it was highlighted the advantages offered by the DGM. The saving in CPU time and memory requirements compared to the LE are significant and can be quantified in a saving of up to one order of magnitude.

Chapter 6 Surface Mesh Parameterisations

6.1 Introduction

In Chap. 5 it was discussed how, using the DGM, it is possible to efficiently map the variation of any objective function as a function of a movement of the surface. This approach, although very powerful, suffers from smoothness issues both in the shape updates and in the gradients. Another approach is to use a parameterisation technique to control the surface mesh nodes. In this case, given a sensible parameterisation, both the surface updates and the gradients are guaranteed to be smooth.

The FFD is chosen to parameterise the 3D wing due to its simplicity and the capability to recover the original shape without the need to perform any geometric inverse fitting, which was one of the requirements of the study to be presented in Chap. 7.4. However, since the FFD is based on BP it offers poor local control unless the control box is constructed in a small localised area.

In the literature, there are always been a dichotomy between the free-nodes and the parametric approach. The first allows the exploration of the entire design space, but suffers from poor deformation smoothness. On the other hand, the latter offers always a smoothed shape, but with a reduced design space exploration capability. Thus, it is natural to seek a formulation which allows the maximum space exploration while maintaining a smoothed shape, something that neither the discrete nor parametric approach is able to offer.

To bridge the gap between shape smoothness and design space exploration, the CST was modified to address both requirements. The newly modified CST formulation, termed *l*-CST, as in “local” CST, is based on the corrected BPs which confers the CST parameterisation method a strong on-demand local control. Various possible applications are discussed along with the effect of the parameterisation on the resulting pressure distribution, aerodynamic coefficients and on an inverse geometric fitting.

Within these two choices, namely parameterised surface and free-nodes approach, the smoothing issue is addressed from both the gradient and the shape point of views.

The chapter concludes with a summary of the most important findings.

6.2 Survey of the Most Important Existing Methods

Generally in aerodynamic optimisation where parameterisation is a fundamental part, curves and surfaces can be manipulated in many ways such as PDE, camber, NACA aerofoil, CPs (Control Points) or direct manipulation parameterisation. Samareh [144] provides an extensive review of some of these options. In this work, the discussion is restricted to the CPs and the direct manipulation methods because, as noted by Yamazaki *et al.* [145], these are the most common forms of parameterisation used in aerodynamic optimisation.

The former is the classical method and probably the most implemented in aerodynamic optimisation exercises. CST, Bézier and B-Splines (Basis) parameterisation are all good examples of this category. The main drawback of this types of methods is the lack of intuitiveness. In other words, there is no direct or intuitive relationship between the displacement of the CPs and the corresponding surface deformation. As a consequence, the bounds of each CP are difficult to define as they are the result of tedious cut-and-try solutions. There are two solutions to this problem. The first option is to increase the number of CPs in such a way to redistribute more locally the deformation, or alternatively local control can be imposed directly to the curve. The latter is not easy as most parameterisation techniques do not have this property or, if they do, this is done to the expense of an increase of the formulation complexity.

On the other hand, the direct manipulation method [146] maintains the CPs as a means to control the curve, but perturbations are made directly using the curve points as the location of the CPs are recomputed automatically each time. A method such as this gives a strong intuitive property to the system. However, if the number of CPs are too low, the method is generally under constrained and an infinite number of solutions is an inevitable consequence as there are infinite CP values that satisfy the user specified displacements [145].

How the parameterised curve is controlled is therefore very important as it determines some other features such as: parsimony (conciseness), intuitiveness, orthogonality, physical meaning and local/global control. Many methods were proposed and some of the most important are

briefly classified in Fig. 6.1. Each one of them has its own advantages and disadvantages in relation to the points presented above. The most used methods are by far the free-nodes approach, H-H (Hicks-Henne) bump, Bézier, B-splines, NURBS (Non-Uniform Rational Basis-Spline), CST and FFD. For this last two, a dedicated discussion is provided for reasons that are made clear later on in this chapter.

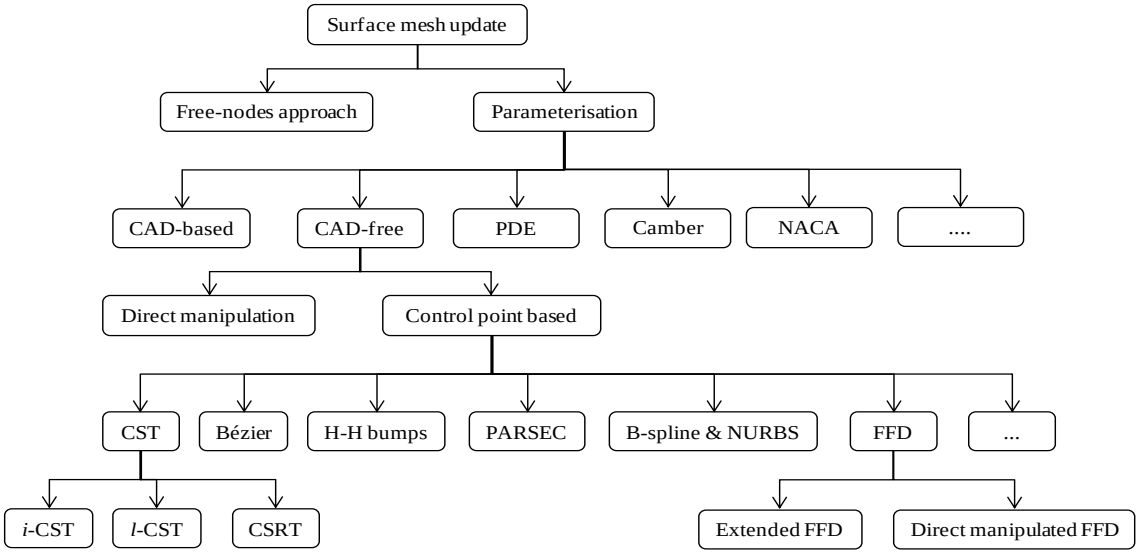


Figure 6.1 Classification of the parameterisation techniques.

The free-nodes method (also known as discrete method) works directly on the surface mesh nodes making it one of the most intuitive. Jameson [6] and Campbell [147] noted that this method allows one to explore all the design point of the design space with strong local control, but the issues related to its smoothness and the large N_{DV} need to be addressed. Jameson and Kim [9] provided some examples and a method to address its inherent noisy trend. The method is based on the concept of Sobolev gradient for which a detailed discussion is offered in Chap. 6.6. Of course, having access to the entire design space poses a question on the parsimony of the method in terms of the Hessian memory requirement, if a second order optimiser is used. Another interesting point to note is its inherent lack of intuitive parameters, such as leading radius, minimum thickness and so on.

The idea behind the H-H (Hicks-Henne) bump [2] is to add smooth deformations to the original shape through a weighted sum. H-H used a trigonometric bump function, but other options can be used such as Wagner, Legendre, BPs or even NACA functions. Khurana *et al.* [148] and Eyi and Lee [149] tested different combinations of the shape functions listed above in a geometric inverse fitting and inverse design exercise. They concluded that the H-H and Legendre functions offered the best performance in terms of flexibility and convergence speed.

The Bézier curve is defined as the dot product between the BP blending functions and the CPs vector. The BPs are then defined as the dot product between a binomial and a polynomial term. This combination offers a smooth parameterised curve. One of the largest advantage, which is often overlooked, is the property of being able to parameterised the same curve using different degrees, i.e. degree elevation. On the downside, there is the issue that the order is directly linked with the N_{CP}. In fact, for the case where a high N_{CP} is necessary, numerically instability in form of curve oscillations is a notable issue of the method. Nevertheless, due to their simplicity, there are many applications available in the literature and an example can be found in Cosentino and Holst [150].

B-splines are very similar in their mathematical formulation to Bézier formulation. In fact, they also feature a dot product between a CPs vector and a series of blending functions. The main differences is that, contrary to Bézier parameterisation, the blending functions are non-zero only on a specific part of the curve [151] accordingly to the distribution enforced by the knot vector. If these are equally spaced, the B-splines are called *uniform*, otherwise are referred to as *non-uniform*. A disadvantage of this implicit form is that circles and conics cannot be represented unless one used the concept of rational B-splines. If this last mentioned formulation is used along with the non-uniform knots one arrives to the so called NURBS definition. Both B-spline and NURBS have two very important and distinguishing features when compared to the Bézier curve method. Firstly, the degree of the curve is not determined by the N_{CP}, whereas a Bézier curve features a curve degree that depends to the N_{CP}. Secondly, the local control of the curve is possible unlike Bézier, where moving one CP affects the whole curve. Examples can be found in Sasaki and Obayashi [152] for a comparison between B-spline and Bézier curve, Lepine *et al.* [153] for application of the NURBS curves in inverse design and Bentamy *et al.* [154] for wing retrofitting.

Although NURBS feature a complex formulation, they have become the de-facto standard in nearly all the CAD (Computer Aided Drawing) softwares [144]. Speaking of CADs, the reasons of their widespread use and applications can be briefly reduced to the capability to deal with complex geometries, provide intuitive parameters, knowledge of the intersection lines and more importantly a common framework for different disciplines. This last feature is very important for MDO applications. On the downside, there are a series of issues, such as gaps for which a CAD repair routine needs to be used and the inherent flaw of not providing directly the sensitivity which is fundamental for any gradient-based optimiser [144]. To be more precise, the CAD sensitivities can be provided if the CAD engine is available to the end user. Since, most of the time, the CAD engine is a proprietary software, this capability is not readily available. Nevertheless, an alternative solution was proposed with the software called CAPRI (Computational Analysis PRogramming Interface) [155] which provides a flexible interface between the CAD-engine and its differentiation. Furthermore, other non-commercial CAD libraries can be used as suggested and tested by Nemec *et al.* [156].

Sobieczky [157] proposed an intuitive 2D method called PARSEC (PARAmetric SECtion). It is based on 11 geometric parameters, each one of them describing a specific feature of the aerofoil. These are the leading edge radius, upper and lower crest position and curvature along with trailing edge location and respective angles. Two polynomials are used, one for the upper and one for lower surface. Although the PARSEC built-in intuitive parameters are considered by some an advantage, there is also a subtle issue that must be highlighted. In aerodynamics there are many non-linearities and linking them to a physical aerofoil properties may not be a straightforward exercise. Examples of how the PARSEC method is used can be found in Fuhrmann [158] and Khurana [159].

Some of the features needed for this work could not be found in the methods surveyed so far. From here, the decision to restrict the attention to other two methods, CST and FFD. The CST provides a nice framework to build an on-demand local control. On the other hand, the FFD allows an easy set up for a 3D geometry with the advantage that it does not require to retrofit the original profile. This last capability is a very important requirement for the non-consistent mesh movement optimisation cases tested in Chap. 7.4. These two methods are duly described in the following sections.

6.3 Free-Form Deformation

The FFD is a parameterisation technique popularised in the aerodynamic literature by Samareh [144], but was originally proposed by Sederberg and Parry [12]. Its main feature is that it parameterises the displacement rather than working directly on the surface or volume. Many modifications were proposed, such as the *extended FFD* [160] and the *direct manipulated FFD* [161]. The reader is referred to the valid literature review by Amoiralis and Nikolos [162] for a full account of the available different formulations.

The FFD consists of creating a parallelepiped box-like lattice around the object which is then mapped using a linear combination of some basis functions and the parametric CPs vector. For instance, Sederberg and Parry [12] used the BPs, but other basis can be used such as NURBS [163].

To formulate the fundamental FFD expression, a local coordinates system is first created:

$$\{\boldsymbol{P}\} = \{\boldsymbol{P}_l\} + u\{\boldsymbol{U}\} + v\{\boldsymbol{V}\} + w\{\boldsymbol{W}\} \tag{6.1}$$

where $\{\boldsymbol{P}\}$ contains the coordinates w.r.t. the object reference system, $\{\boldsymbol{P}_l\}$ contains the coordinates of the origin and $\{\boldsymbol{U}\},\{\boldsymbol{V}\},\{\boldsymbol{W}\}$ are the local unit vectors (for instance $\{\boldsymbol{U}\} = \{1,0,0\}$). The u, v, w local coordinates can then be found as:

$$\begin{aligned} u &= \frac{\{\boldsymbol{V}\} \times \{\boldsymbol{W}\}(\{\boldsymbol{P}\} - \{\boldsymbol{P}_l\})}{\{\boldsymbol{V}\} \times \{\boldsymbol{W}\} \cdot \{\boldsymbol{U}\}} \\ v &= \frac{\{\boldsymbol{U}\} \times \{\boldsymbol{W}\}(\{\boldsymbol{P}\} - \{\boldsymbol{P}_l\})}{\{\boldsymbol{U}\} \times \{\boldsymbol{W}\} \cdot \{\boldsymbol{U}\}} \\ w &= \frac{\{\boldsymbol{U}\} \times \{\boldsymbol{V}\}(\{\boldsymbol{P}\} - \{\boldsymbol{P}_l\})}{\{\boldsymbol{U}\} \times \{\boldsymbol{V}\} \cdot \{\boldsymbol{U}\}} \end{aligned} \tag{6.2}$$

where symbol "·" refers to the dot product, whereas symbol "×" refers to the cross product. By doing so, all the physical coordinates (x, y, z) are mapped into local parametric coordinates $(u, v, w) \in [0,1] \times [0,1] \times [0,1]$. The parameterised FFD coordinates are then elegantly expressed as a trivariate volume tensor-product:

$$\{\boldsymbol{S}(u,v,w)\} = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{p} D_{i,j,k}\ BP_i^n(u)\ BP_j^m(v)\ BP_k^p(w) \tag{6.3}$$

where and n, m, p are equal to the degree of the curve along the *x*, *y*, *z* directions and the N_{CP} are equal to $n+1, m+1, p+1$. Each BP is then defined as follows:

$$\begin{aligned} BP_i^n(u) &= \binom{n}{i}(1-u)^{n-i}u^i = \frac{n!}{i!\,(n-i)!}(1-u)^{n-i}u^i \\ BP_j^m(v) &= \binom{m}{j}(1-v)^{m-j}v^j = \frac{m!}{j!\,(m-j)!}(1-v)^{m-j}v^j \\ BP_k^p(w) &= \binom{p}{k}(1-w)^{p-k}w^k = \frac{p!}{k!\,(p-k)!}(1-w)^{p-k}w^k \end{aligned} \tag{6.4}$$

It is known that, the BP basis functions suffer from poor local control. However, for the purpose of this work (presented in Chap. 7.4), this choice is not of any concern.

One of the most important feature of the FFD method is that all the points embedded in the control volume lattice (see Fig. 6.2) are parameterised and effectively treated as a cloud of points. Therefore, no information regarding connectivity and mesh type is required. Another noteworthy feature is that FFD, contrary to CST, B-splines and Bézier, does not need to perform any additional geometric inverse fitting.

As explained in Chap. 2.4.1, the derivatives of the parameterised surface w.r.t. the parametric DVs are needed. Thus, taking the derivative of Eq. (6.3) w.r.t. $\{\boldsymbol{D}\}$ yields:

$$\left[\frac{\partial \boldsymbol{S}(u,v,w)}{\partial \boldsymbol{D}}\right] = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{p} BP_i(u)\ BP_j(v)\ BP_k(w) \tag{6.5}$$

There are different ways to construct the lattice box and it is acknowledged that different ways of constructing the control box would affect the minimum or better it would affect the way the parameterisation surfs the design space. On this matter, it is also important to highlight that, as noted by Samareh [164], the lattice CPs do not have a direct link to the geometry. Therefore, a change in one of the DV may deform the parameterised object in a non-intuitive way.

In terms of aerodynamic applications, the FFD method was widely used and tested against a different variety of other methods. For instance, Bloor *et al.* [165] compared the FFD against the PDE in a pure geometry modelling exercise, Amoiralis and Nikolos [162] against the FFD based on the B-splines in an inverse design exercise and Kenway *et al.* [117] used it for a MDO optimisation application.

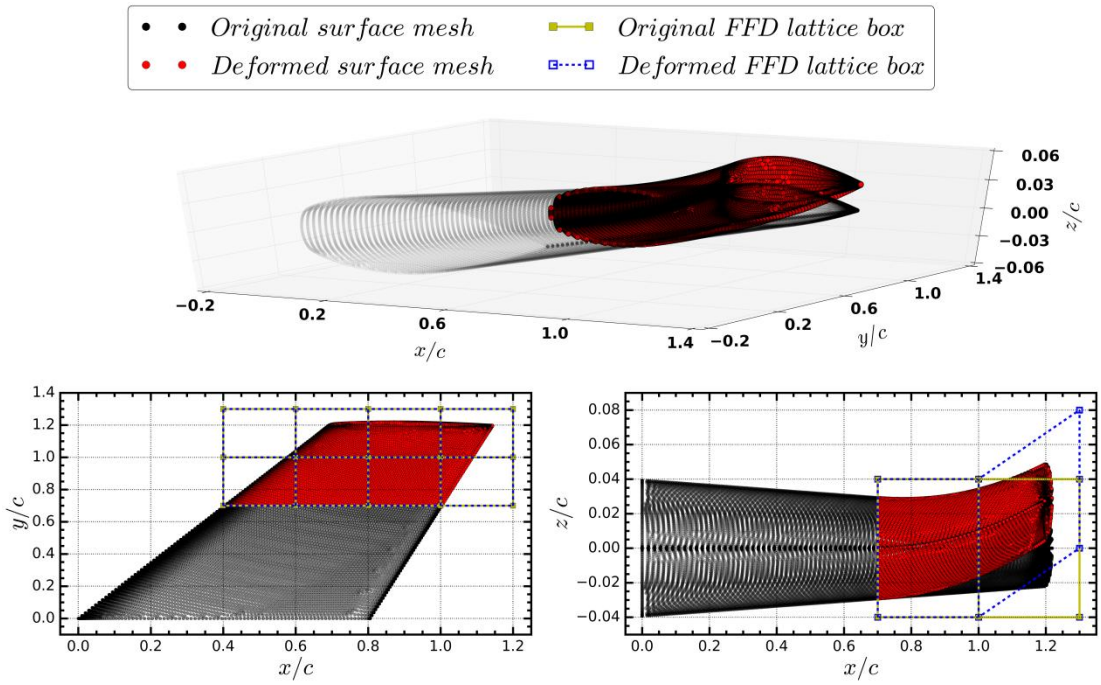


Figure 6.2 Example of a local deformation governed by the FFD.

6.4 Class Shape Transformation

The CST is a relatively new parameterisation technique introduced by Kulfan [11]. Its main advantage is that is able to impose directly some aerodynamic key geometric features by using a Class shape Function (CF). This means that aerofoil-like shapes can be obtained directly and then optimised for a specific use. These are leading edge radius and trailing edge thickness, whereas the other parameters are not intuitive. A solution, termed *i*-CST (intuitive), to this issue was proposed by Zhu and Qin [166]. The CST curve, C_{CST}, is defined as:

$$C_{CST}(u) = CF(u) \cdot \left(\sum_{i=0}^{n} A_i \, BP_i^n(u) \right) + u_{TE} u \tag{6.6}$$

where n is the degree of the curve, i is the index and u is the non-dimensional chord length position, i.e. x/c with c being the chord, and u_{TE} is the trailing edge thickness correction (equal to zero when a sharp leading edge is considered). The degree of the curve is equal to the N_{CP} (number of CPs) minus one ($n = N_{CP} - 1$), whereas the order is the same as per the N_{CP}. The CPs are also generally called coefficients defined as $A = \{A_1, A_2, \ldots . A_{N_{CP}}\}$. The BPs are defined as the dot product between a binomial and a polynomial term as follows:

$$BP_i^n(u) = \binom{n}{i} (1-u)^{n-i} u^i = \frac{n!}{i!\,(n-i)!} (1-u)^{n-i} u^i \tag{6.7}$$

They effectively work as a smooth basis blending function and there are as many BPs as CPs. If a higher order of accuracy is needed, one is left with no choice but to increase the N_{CP}. This is because each BP has a global influence on the shape of the curve. In fact, they are non-zero all along the aerofoil chord as can be seen in Fig. 6.3 (centre). The CST parameterisation makes use of the BPs and as such suffers from the same issues. The CF depends on two parameters, namely N_1 and N_2, and is defined as:

$$CF_{N_2}^{N_1}(u) = u^{N_1}(1-u)^{N_2} \tag{6.8}$$

where for aerofoil-likes shape, N_1 and N_2 take the value of 0.5 and 1 respectively, allowing to impose directly a round leading edge. The CF is not involved in the parameterisation process because is used to enforce a known aerodynamic shape. In fact, Ceze *et al.* [167] noted that it is the only source of non analyticity in the CST formulation. To this regard, an interesting solution was proposed by Marshall [168] where a solution to represent the CST in terms of exact Bézier curve (analytical) was suggested. On the other hand, the BPs are analytical functions and are solely responsible for the shape parameterisation because they work as a tuning functions on CF. Substituting Eq. (6.8) back in Eq. (6.6) yields:

$$C_{CST}(u) = (u^{N_1}(1-u)^{N_2}) \cdot \left(\sum_{i=0}^{n} A_i \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i} \right) + u_{TE} u \tag{6.9}$$

Two interesting points can be made here. Firstly, the resulting BPs have a consistent lower value as depicted in Fig. 6.3 (right). The second point is the reduced sensitivity w.r.t. the coefficients as depicted in Fig. 6.3 (centre) as compared to Fig. 6.3 (right). In both cases, this is happening because the CF is effectively a power basis.

6.5 *local*-Class Shape Transformation

A truly useful optimisation environment must be able to find subtle beneficial changes that are non intuitive when starting from a fairly good baseline design. Wu and Padula [169] noted that if an optimisation is set up to try to improve an already fine-tuned aerofoil with insufficient low-resolution design space any improvement can be very difficult unless by luck. This is why a turn on/off locality feature is highly desiderable especially at the transonic regime where improvements take the shape of highly localised tweaks that aerodynamic parameterisation should be able to enforce [170, 171].

As far as it regards the local support, valid alternatives are represented by B-splines and NURBS which require the definition of a knot vector. If more knots are added, the B-spline acquires more and more local control. There are two shortcomings in this process. First adding more and more knots increases the possibility of spurious oscillation and secondly, will likely change the other knot positions something which is not recommended due to the unpredictability of the resulted parameterised curve.

In the literature, to the best knowledge of the author there is only one application where the CST method was used in combination with the B-spline in an effort to give more local control to the curve. Straathof *et al.* [172] proposed to use the B-spline basis to locally modify the curve and their solution was named CSRT (Class-Shape-Refinement-Transformation). However, this is happening by increasing the N_{CP} due to the addition of a refinement function.

This work proposes a modified version of the CST, termed *l*-CST, where *l* stands for *local*. The same N_{CP} are maintained in order not to increase the degree of the curve while conferring the curve a strong on-demand local control. If Bézier curves are used, the degree of accuracy is linked to the N_{CP} and this is why it is not trivial to find a low dimensional parameterisation featuring high accuracy. This is a long lasting problem related to the Bézier and CST

formulation based on the BPs where an increase in the N_{CP} is reflected by an increase in the curve's degree.

The idea developed in this chapter has been inspired by the work of Punj *et al.* [173] where the authors introduced a cosine function that gives various degree of local control to each one of the rational Bézier CP. In the present work, a different correction function is proposed and extending it to the CST method based on the non-rational BPs. The *l*-CST parameterisation is defined as:

$$C_{l-CST}(u) = CF(u) \cdot \left(\sum_{i=0}^{n} A_i \; BP_i^n(u) \; LF_i(u) \right) + u_{TE} u \tag{6.10}$$

where the LF (Local Factor) function is defined as:

$$LF_i\left(u_{CP_i}, l_i, \delta_i\right) = \delta_i (\cos(\Delta u_i))^{l_i} \tag{6.11}$$

where Δu_i is defined as the difference between the non-dimensionalised coordinate and the point along the chord where local control is needed, i.e. u_{CP_i}:

$$\Delta u_i = u - u_{CP_i} \tag{6.12}$$

There are as many δ_i, l_i and u_{CP_i} as the N_{CP}. So, if it is desired to give strong local control to each one of the DV, the total number of extra variables is equal to $3 \times N_{CP}$. Alternatively, local control can also be applied to a single DV at the time which equates to an increase in the N_{DV} equal to 3. These parameters establish where the LF modifies the curve and how strong the local control would be respectively. Substituting Eq. (6.11) back in Eq. (6.10), the following relation holds:

$$C_{l-CST}(u) = (u^{N_1}(1-u)^{N_2}) \cdot \left(\sum_{i=0}^{n} A_i \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i} \left(\delta_i (\cos(\Delta u))^{l_i}\right) \right) + u_{TE} u \tag{6.13}$$

If exponents l_i take values greater than one they introduce more locality, whereas the vice-versa is true if they take values smaller than 1 with the exception of 0. Obviously when the exponents l_i take a value equal to zero and the $\delta_i = 1$, the original CST parameterisation is recovered.

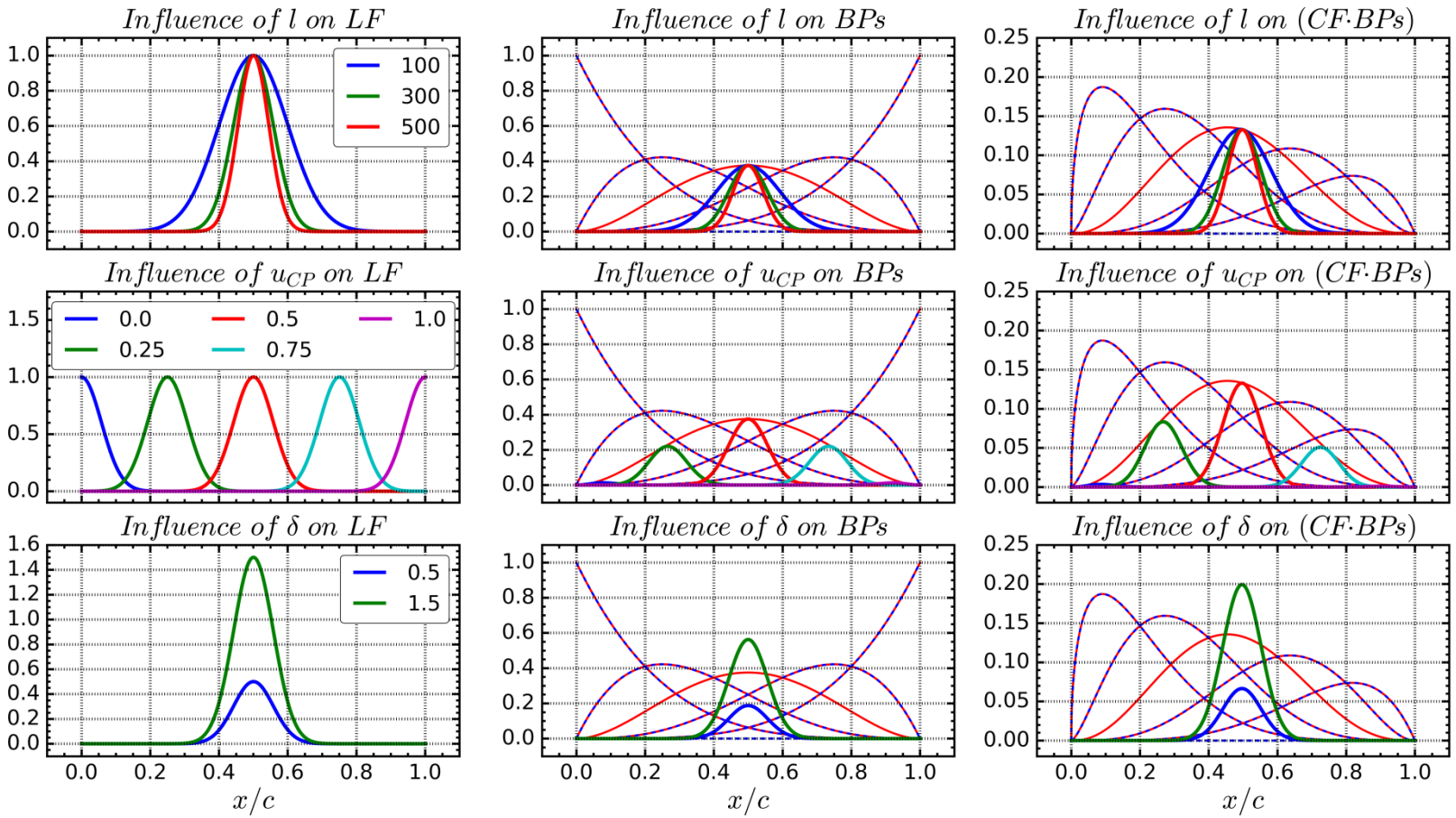


Figure 6.3 Plot of the influence of the LF control parameters.

This gives an on-demand added control to the curve leaving unchanged the original curve degree. Parameters δ_i tune the distance between the curve and the local modified curve crest. This is a feature that only rational polynomial (for instance NURBS) can offer. These four extra coefficients confer intuitiveness to the curve because one can specify the value of u_{CP_i} where a greater a priori local control is desired.

As can been seen from Fig. 6.3 (centre) the LF is built in such a way to follow the u_{CP} associated to its basis function considered. This gives the possibility to choose a specific area over the aerofoil and enforce a local perturbation while maintaining the global control features of the other CPs untouched. This is particularly important as this is an additional feature conferred to something that was generally thought to have only global control. This gives a reason to use the *l*-CST instead of either B-spline or NURBS.

6.5.1 Comparison Between the *l*-CST and the H-H Bump

At this point, it is important to highlight the differences between the *l*-CST and the H-H [2] bumps. This is necessary because the two formulations seem to be very similar and their fundamental difference is not trivial. The H-H parameterisation enforces a deformation on the baseline profile by linearly adding a series of basis functions. In formula:

$$y_{u,l} = y_{u,l}^{basis} + \left(\sum_{i=0}^{n} a_i f_i \right) \tag{6.14}$$

where $y_{u,l}^{basis}$ are either the upper or the lower discrete surface coordinates of the original profile and f_i are the bump functions. The most used bump function is the sinusoidal function defined as:

$$f_i(u) = \sin\left(\pi u^{\frac{log_{10}(0.5)}{log_{10}(t_{1_i})}} \right)^{t_{2_i}} \tag{6.15}$$

where t_{1_i} and t_{2_i} are two constants defined a-priori. The former defines the position along the chord of the maximum *i-th* bump peak, whereas the latter defines the width of the *i-th* bump. Thus, coefficients t_{1_i} and t_{2_i} enforce a change in the parameterised curve in a similar manner as

coefficients u_{CP_i} and l_i. However, by comparing Eq. (6.10) and Eq. (6.14), it can be noticed that where the LF works directly on each BP basis using a product, the H-H bump adds to the baseline profile a series of different bumps. In conclusion, a very localised deformation can be obtained by both methods, but using two different strategies.

6.5.2 Sensitivity Analysis

The analytical sensitivities of both CST and *l*-CST parametric curves is uniquely defined and easy to get. Differentiation of Eqs. (6.6) and (6.10) w.r.t. the CPs leads respectively to:

$$\frac{\partial C_{CST}(u)}{\partial A} = \frac{\partial}{\partial A}\left(CF(u)\cdot\left(\sum_{i=0}^{n} A_i\, BP_i^n\,(u)\right) + u_{TE}u\right) = CF(u)\cdot\left(\sum_{i=0}^{n} BP_i^n\,(u)\right) \tag{6.16}$$

$$\frac{\partial C_{l-CST}(u)}{\partial A} = CF(u)\cdot\left(\sum_{i=0}^{n} BP_i^n(u)\, LF_i(u)\right) \tag{6.17}$$

Interestingly, even if the control coefficients are changed the derivatives stay constant as they do not have any explicit dependency upon them. Physically these equations show the parameterised surface points affected if a change is made on each one of the CP. This is a valuable tool as it tells where to move the point and the magnitude of the movement if a predetermined perturbation is desired.

The derivative of the CST curve w.r.t. its CPs is identical to its CF times the BP and this explains the reduced sensitivity as depicted in Fig. 6.3 (centre). Furthermore, from Fig. 6.3 (centre and right) it is evident that the sensitivities show a global behaviour as expected. Finally, performing the same derivative also for the *l*-CST, the LF remains and this explains how it imposes the local control to the curve.

The other type of sensitivity which is generally needed is the sensitivity of the curve w.r.t. the points along the chord, hence the slope and curvature of the parameterised curve. Also in this case, the derivatives w.r.t. u can be obtained analytically as:

$$\frac{\partial C_{CST}(u)}{\partial u} = \frac{\partial}{\partial u}\left(CF(u)\cdot\left(\sum_{i=0}^{n} A_i\, BP_i^n\,(u)\right) + u_{TE}u\right) \tag{6.18}$$

$$\frac{\partial C_{l-CST}(u)}{\partial u} = \frac{\partial}{\partial u}\left(CF(u) \cdot \left(\sum_{i=0}^{n} A_i \; BP_i^n(u) \; LF_i(u) \right) + u_{TE} u \right) \tag{6.19}$$

It is known that the derivatives of the BPs are continuous except at the extrema, whereas the CST and consequently also *l*-CST, have a singularity point only at the leading edge due to the square-root term needed to impose a round leading edge [11]. Eqs. (6.18) and (6.19) are derivatives which can be found in Farin [151]. For the sake of simplicity only the first and second derivative of the newly introduced LF is given here:

$$\frac{\partial (LF_i)}{\partial u} = -\delta_i \; l_i \; \sin\left(u - u_{CP_i}\right) \left(\cos\left(u - u_{CP_i}\right)\right)^{l_i - 1} \tag{6.20}$$

$$\frac{\partial^2 (LF_i)}{\partial u^2} = -\delta_i \; l_i \; \left(\left((l_i - 1) \left(\sin\left(u - u_{CP_i}\right)\right)^2 \left(\cos\left(u - u_{CP_i}\right)\right)^{l_i - 2} \right) + \left(\left(\cos\left(u - u_{CP_i}\right)\right)^{l_i} \right) \right) \tag{6.21}$$

The first and second derivative as expressed in Eqs. (6.20) and (6.21) are depicted in Fig. 6.4. Since LF is a trigonometric function, it is infinitely times differentiable.

To analyse the effect of the LF on both the BPs and the CST, let us consider a fifth degree (order 6) BP having 6 coefficients with u_{CP} located at midchord. The attention is on the third coefficient located at mid span for which it is easier to show how the LF works and how the other BPs are left unchanged. The influence is first shown on the BPs and secondly on the CF times BPs curve so to highlight the effect on each factor.

The effect of a change in any exponent l_i on the LF, BPs and the CF times BPs is depicted at the top of Fig. 6.3. As can be seen, an increase in the exponent forces the curve to take values closer to zero away from the position of u_{CP}. The effect of this LF on the third BP is evident as it is taking the curve from having a control all over the chord length to a small localised region of the aerofoil. The same effect is seen also on the CF times BPs curve.

For the sake of simplicity, an u_{CP} positioned at midchord was chosen, but as shown in Fig. 6.3 (centre) different locations can be chosen while working always on the same third BP basis. The effect on the selected BP (Fig. 6.3 centre) and CST Fig. 6.3 (centre left) curve is as described before.

The next terms to analyse are δ_i. If these terms are equal to one neither the BPs nor the CF times BPs curves are affected by the LF. When these terms are different from one, it is clear that the effect is very similar to what rational BPs can do. In fact, these give the possibility to modify the distance between the curve and the local curve crest as depicted in Fig. 6.3 (bottom).

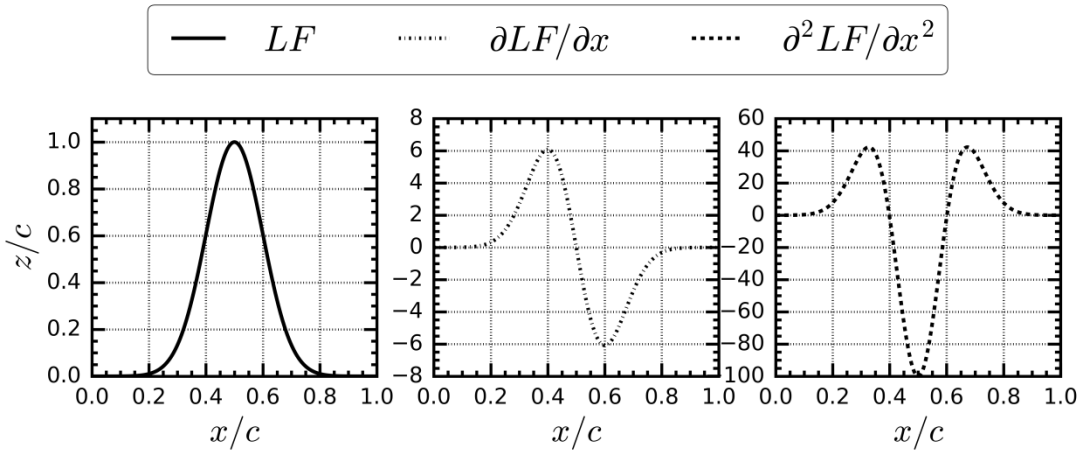


Figure 6.4 LF function and its derivatives.

6.5.3 Examples of Possible Applications

In this section two examples are given to show the capability of the *l*-CST method. The first example compares the *l*-CST against the CST method for relatively large visible deformations. On the other hand, the second example shows how a quasi-discrete local control can be achieved. This is done to show that, although a parameterisation technique is used, a very strong local control (very much like that allowed by the free-nodes approach) can still be achieved.

The first step that needs to be performed is a geometric fitting of the upper and lower profiles separately using a 11th degree curve for both the CST and the *l*-CST parameterisations. As a second step only one coefficient is modified in both cases. An 11th degree CST curve has 12 coefficients (degree of the curve plus 1), however the number of free parameter used by the *l*-CST (in Figs. 6.5-6.7) with local control given only to one coefficient is $12 + 3$, where the three extra coefficients chosen for this case are l_6, δ_6 and u_{CP_6}.

Referring to Fig. 6.5, two deformations were imposed, one on the upper side and the other on the lower one where the following settings were used: $\Delta a_6 = 0.3$, 11th degree curve, $u_{CP_6} = 0.5$, $l_6 = 200$ and $\delta_6 = 1.001$. As can be seen, the CST has an effect that spans more than a half chord, whereas the *l*-CST reduces the extent of the perturbation to a much smaller area. This can be seen in Fig. 6.5 (bottom) where the point-by-point differences w.r.t. the original fitted CST and *l*-CST curve are plotted. Two points can be made here, namely the CST and *l*-CST curves at the perturbed *x*-location are different because $\delta_6 \neq 1$ and the difference between the two perturbations are greater at the peaks and where the curves recover the original shape.

For the second example, Fig. 6.6 depicts how a quasi-discrete local control can be obtained where the following settings were used: $\Delta a_6 = 0.3$, 11th degree curve, $u_{CP_6} = 0.5$, $l_6 = 2000$ and $\delta_6 = 1$. To make this point clearer a test was performed to find what is the degree of the original CST curve that recovers the local control property provided by the *l*-CST. As can been seen from Fig. 6.7 (using the same settings as per Fig. 6.6), not even a CST curve featuring a 41st degree can recover the local-control capability offered by the *l*-CST using an 11th degree. Also note that, as the BP degree is increased (a consequence of the increased N_{CP}), the magnitude of the deformation is reduced, whereas the *l*-CST matches exactly that of the original 11th degree CST.

On this matter, Painchaud-Ouellet *et al.* [174] observed that adding more local control has the same effect of adding more CPs, which leads to modifications occurring at a much smaller scale. This can possibly lead to undesirable bumpy aerofoils and can be viewed as a potential problem for the *l*-CST formulation. However, there is a fundamental difference. While increasing the order of the CST curve increases the risk of obtaining localised bumps all over the profile, the addition of a strong local control affects only a specific area. While one is undesirable, the other is intentional.

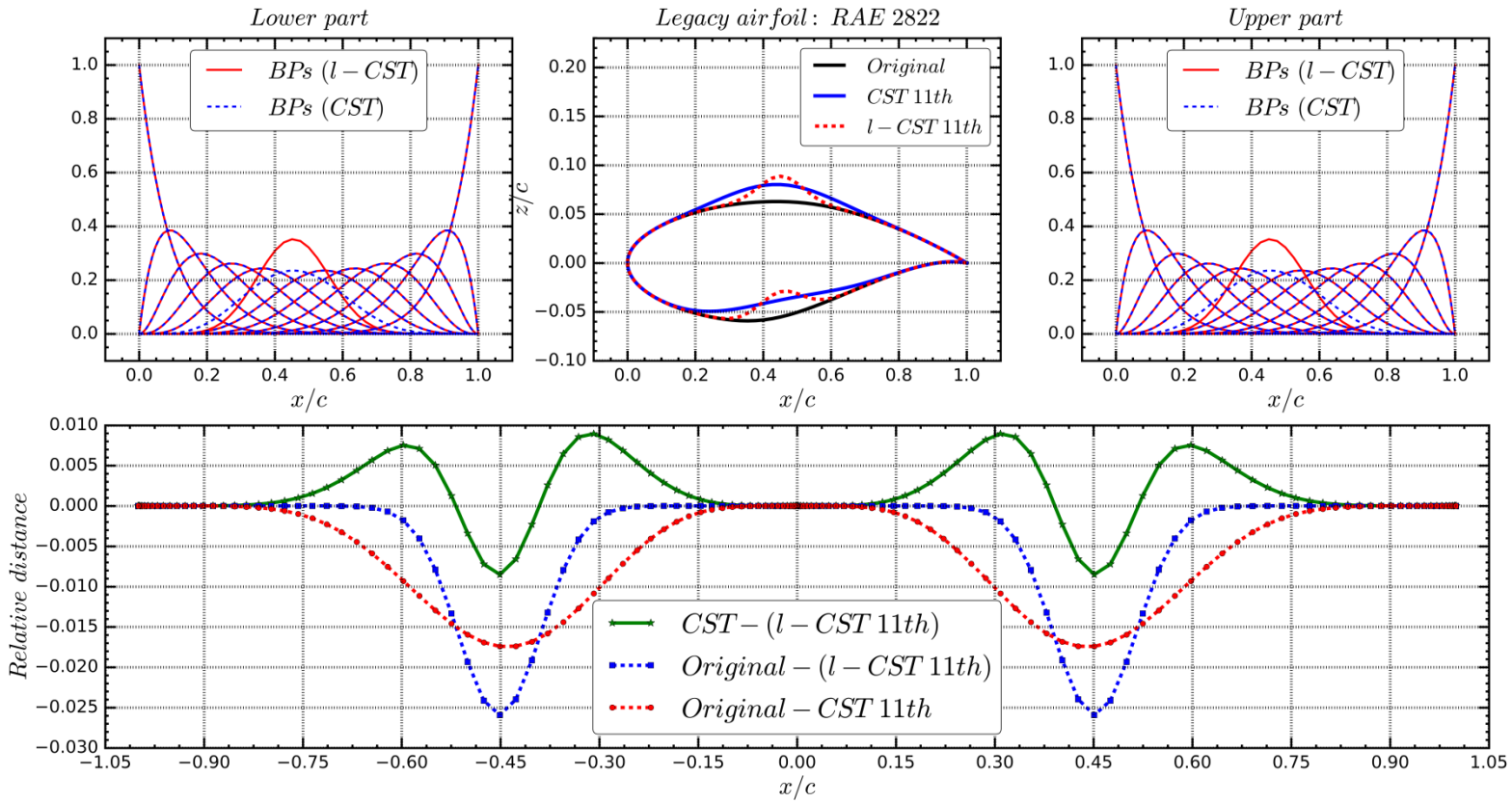


Figure 6.5 Example of local deformations.

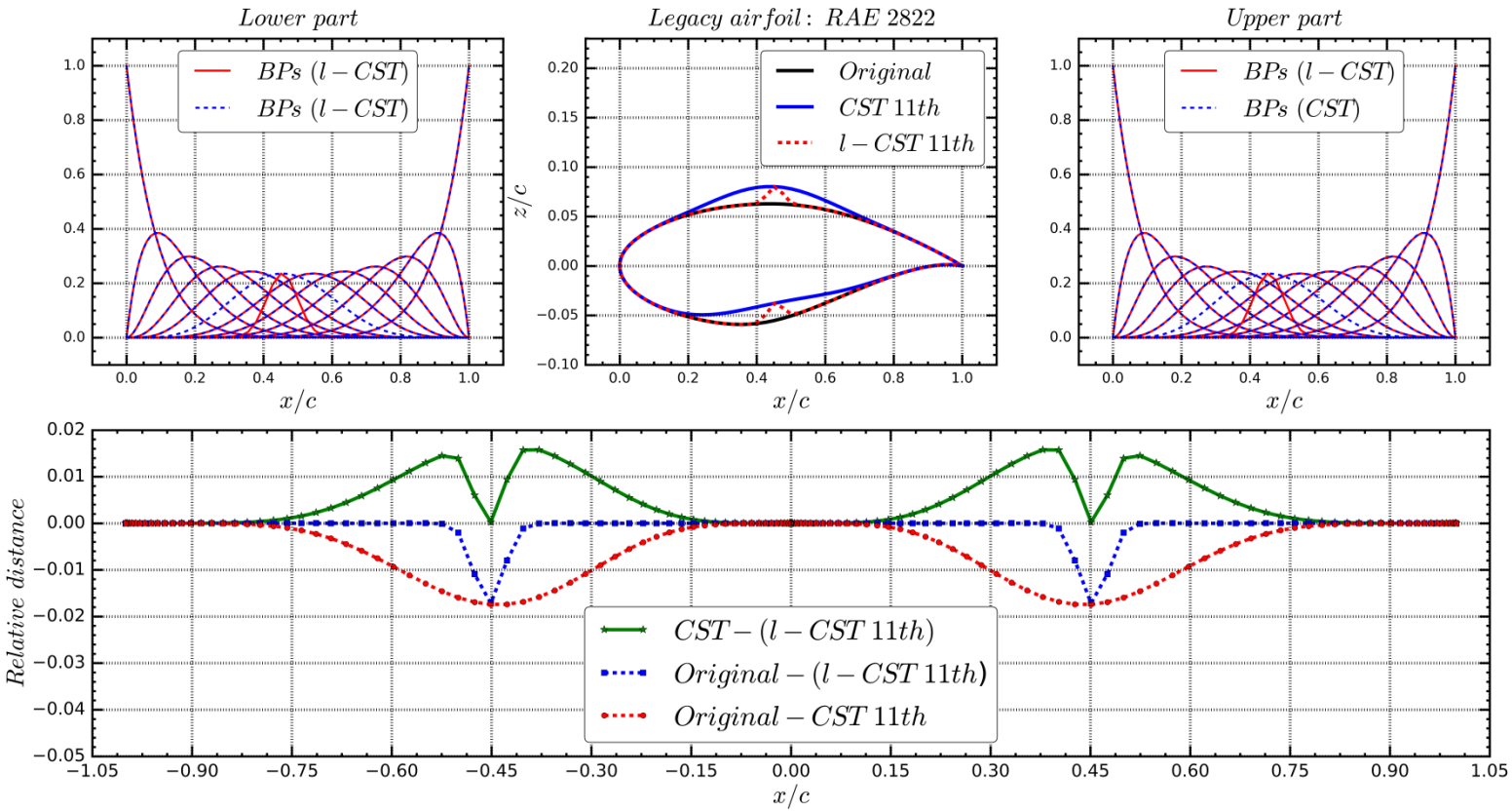


Figure 6.6 Example of almost discrete-like deformations.

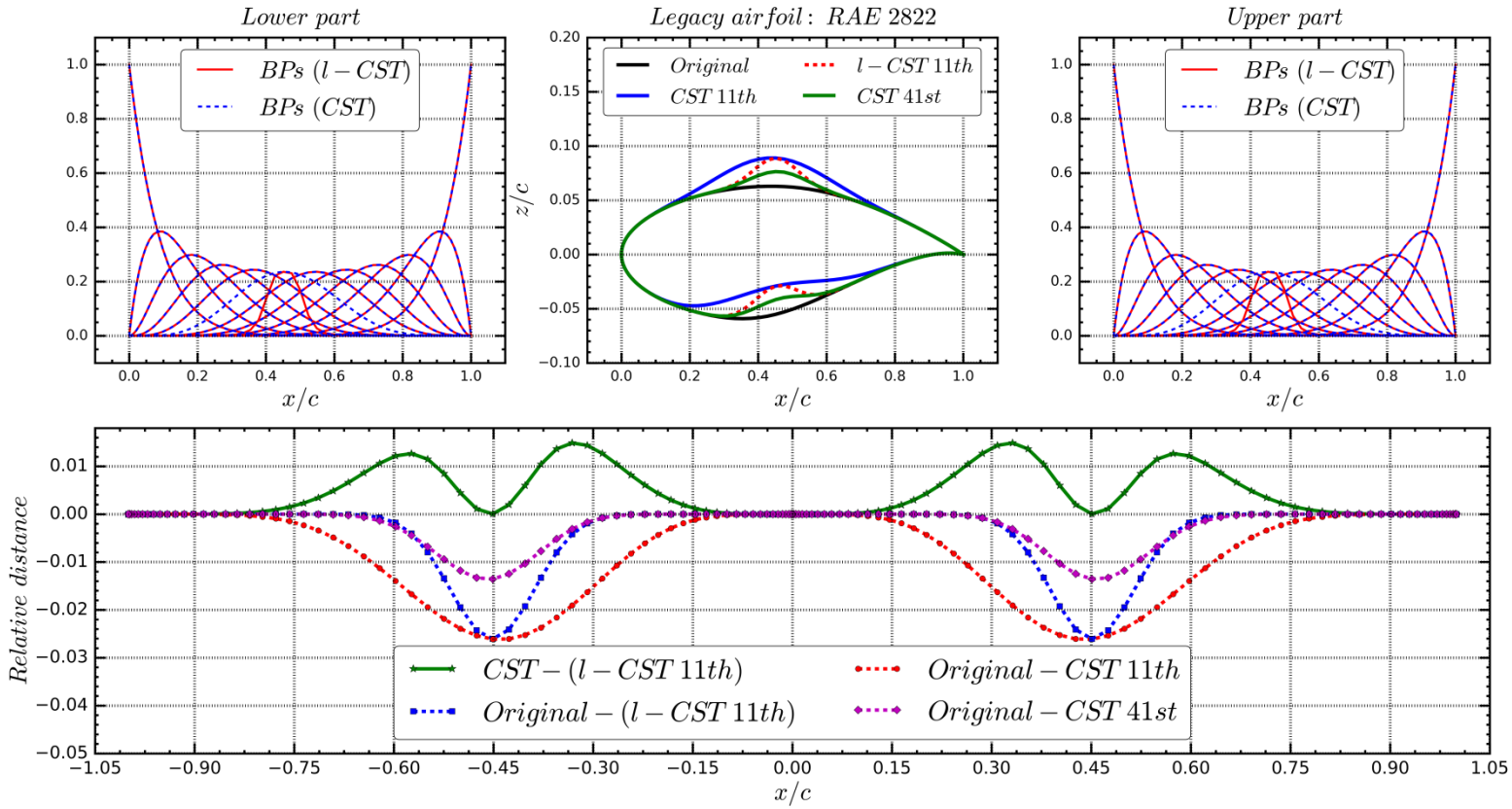


Figure 6.7 Example of the effect of an increase in the curve degree.

6.5.4 Geometric Inverse Fitting

The task of curve fitting is to find a smooth curve that fits the data points on the average. Fitting was chosen in place of interpolation because the risk of having high frequency oscillations increases when the interpolation process is tested against closely spaced points such as those at the leading edge. In fact, while interpolation stays true to the training points, fitting does not and the smoothing capability is stronger [171].

One of the most important features that a parameterisation should have while performing a geometric inverse fitting is the capability to increase the accuracy. This is generally done by increasing the number of CPs which allows one to refine the fitting, a feature commonly referred to as hierarchy of the parameterisation [170, 171]. This cannot always be done without impairing other good features such as stability in particular. In fact, the curve should have a simple form, e.g. a low-order polynomial, so as not to reproduce the noise [175]. Having said that, the goal of this exercise is to show how the *l*-CST allows an increase of the retrofitting accuracy without increasing the degree of the curve.

The most commonly used method to measure the fitting error is the least-squares fit because squaring emphasises larger errors relative to smaller ones. Therefore, the L_2 norm is used to measure how good the fit is:

$$L_2 = \sqrt{\sum_{i=1}^{N_{SM}} \left(C(u_i)_{originial} - C_{para\,,fitted}\,(u_i) \right)^2} \tag{6.22}$$

where $C_{para\,,fitted}$ designates either $C_{l-CST,fitted}$ or $C_{CST,fitted}$. The fitting is not generally done for the horizontal coordinates because the sensitivity is much lower when compared to the vertical direction as reported by Amoiralis and Nikolos [162]. However, it is known that this choice is not ideal at the leading edge where the behaviour is very similar to the local aerofoil tangent, hence going to infinite.

The aerofoil is split in two parts, an upper and lower part and Eq. (6.10) is applied to each one of them. Three different N_{CP} equal to 4, 8, 12 corresponding to 3rd, 7th, 11th BP degree curve were tested so as to facilitate the comparison between the CST and *l*-CST curves as the N_{CP} is increased.

As can been seen from Figs. 6.8 to 6.11, two types of fluctuations in the error plot can be clearly identified. Big scale fluctuations show where the aerofoil is stiff to fit and this type of error can be reduced by simply increasing the N_{CP} or using the newly proposed *l*-CST. On the other hand, small scale fluctuations are not due to fitting problems and the reason of these deviations needs to be investigated on the way the data were collected and passed over. One way to reduce this noise is to use analytically defined aerofoils (such as the 4, 6 digits NACA profiles).

From the series of plots presented in Figs. 6.8-6.11, it is clear how the *l*-CST differs from the CST. As the BP degree is increased the advantages offered by the *l*-CST diminish because the CST can almost recover the fitting accuracy offered by the *l*-CST. This means that the *l*-CST allows to use a lower order BPs without the expense associated with the high order BPs. It is remarkable that the 7th degree *l*-CST can offer, in some area of the aerofoil, the same accuracy of a 11th order CST (see for instance Fig. 6.10). However, there are points where the *l*-CST does not perform as well as the CST, namely in Fig. 6.9 for the 7th degree *l*-CST. This is not really an issue because it can be solved either by increasing the BP degree as shown in the same plot or by turning off the local control.

An interesting thing to remark is how the BP basis functions for the CST and the *l*-CST method differ as depicted in Figs. 6.8-6.11 top right and left. In particular, a closer look reveals that some of the BP basis functions feature a reduced local factor making a clear example of the possibility that not always more local control leads to a better fitting.

To investigate more this aspect, Tab. 6.1 tabulates the values of the NLR-7310 aerofoil local coefficients as the degree of the curve is increased. In Tab. 6.1, the first coefficients control the area close to the leading edge, whereas the last control the area close to the trailing edge. It is clear that, as the curve degree is increased, most of the exponent l_i controlling the aerofoil trailing edge take null values. This means that these DVs do not need more local control as far as it regards the geometric fitting. This can be explained with the fact that, as one increases the degree, the control is distributed more evenly on the entire aerofoil reducing the need for more local support. However, the leading edge still remains a critical and difficult area to retrofit and this explains the higher value of the first coefficients l_i.

In Figs. 6.8-6.11, the degree of the curve was kept constant, which is good to avoid curve oscillations (stability), but the number of free parameters used by the *l*-CST is higher (double)

than the CST because coefficients l_i were allowed to change. However, by looking at the results reported in Tab. 6.1, it is possible to see which area of the aerofoil requires more local control. Using this a-posteriori analysis, it is then possible to reduce the number of the *l*-CST free variables by turning off the local control where it is not needed. This is shown in Fig. 6.12 by comparing the 3rd degree *l*-CST with the 7th degree CST. These last two have now the same number of free-variables, i.e. eight. To be more precise where the *l*-CST has 4 A_i along with 4 l_i, the CST has 8 A_i. As can be seen the 3rd degree *l*-CST is able to provide a comparable level of fitting accuracy.

Table 6.1 Optimised values of the exponent coefficients.

DV	**3rd degree *l*-CST curve exponent coefficients**		**7th degree *l*-CST curve exponent coefficients**		**11th degree *l*-CST curve exponent coefficients**	
	upper	**lower**	**upper**	**lower**	**upper**	**lower**
1	-9.921	-8.810	11.160	-17.578	2.820	186.732
2	-14.304	-18.830	76.963	29.078	-37.290	102.483
3	-8.621	-6.549	13.866	-26.866	4.443	-39.925
4	0.0	0.0	0.0	0.0	0.0	0.0
5	There are 4 local coefficients for a 3rd degree curve		0.0	0.0	0.0	0.0
6			0.0	0.0	0.0	0.0
7			0.0	0.0	0.0	0.0
8			0.0	0.0	0.0	0.0
9			There are 8 local coefficients for a 7th degree curve		0.0	0.0
10					0.0	0.0
11					0.0	0.0
12					0.0	0.0

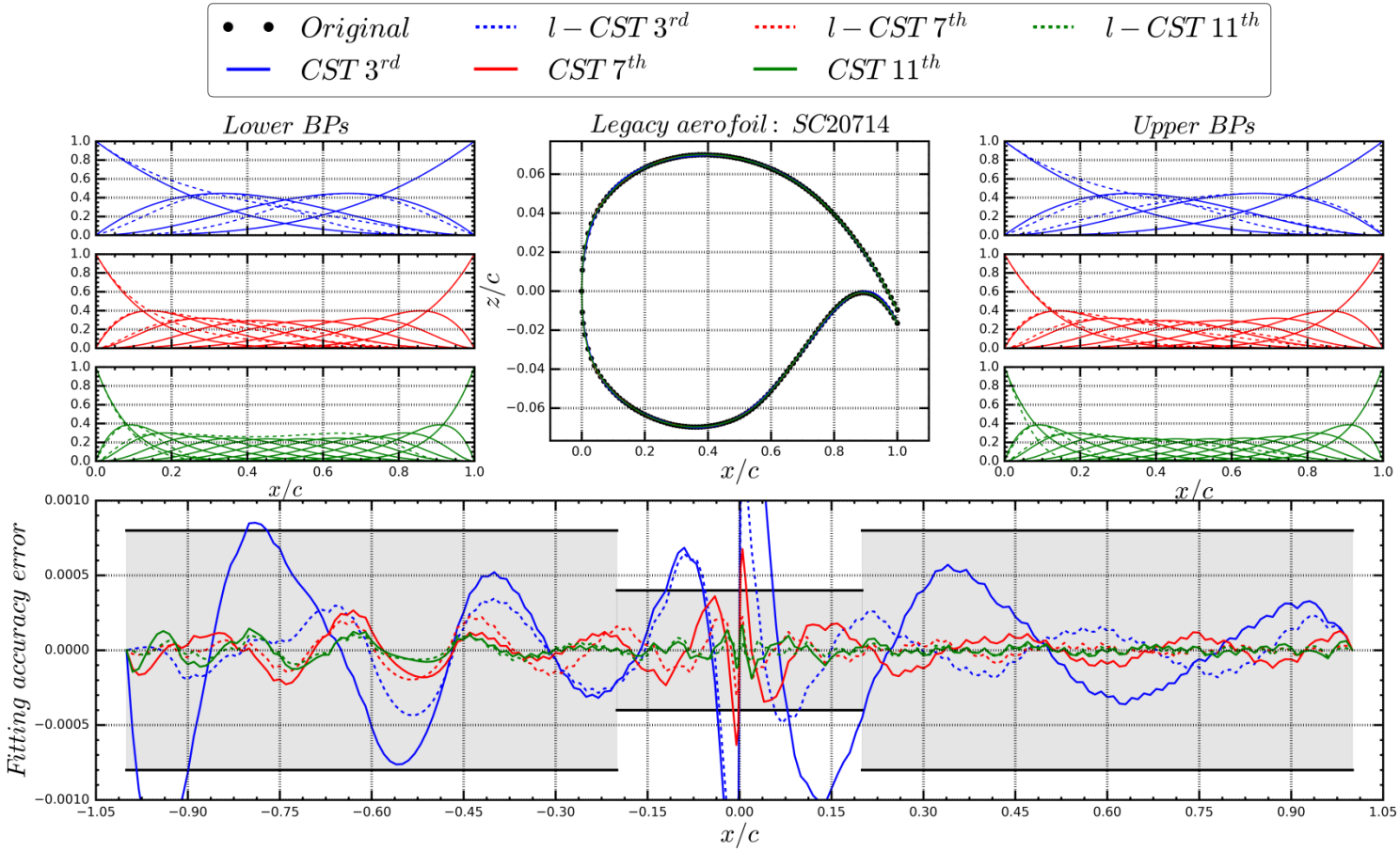


Figure 6.8 Geometric fitting for the SC20714 aerofoil.

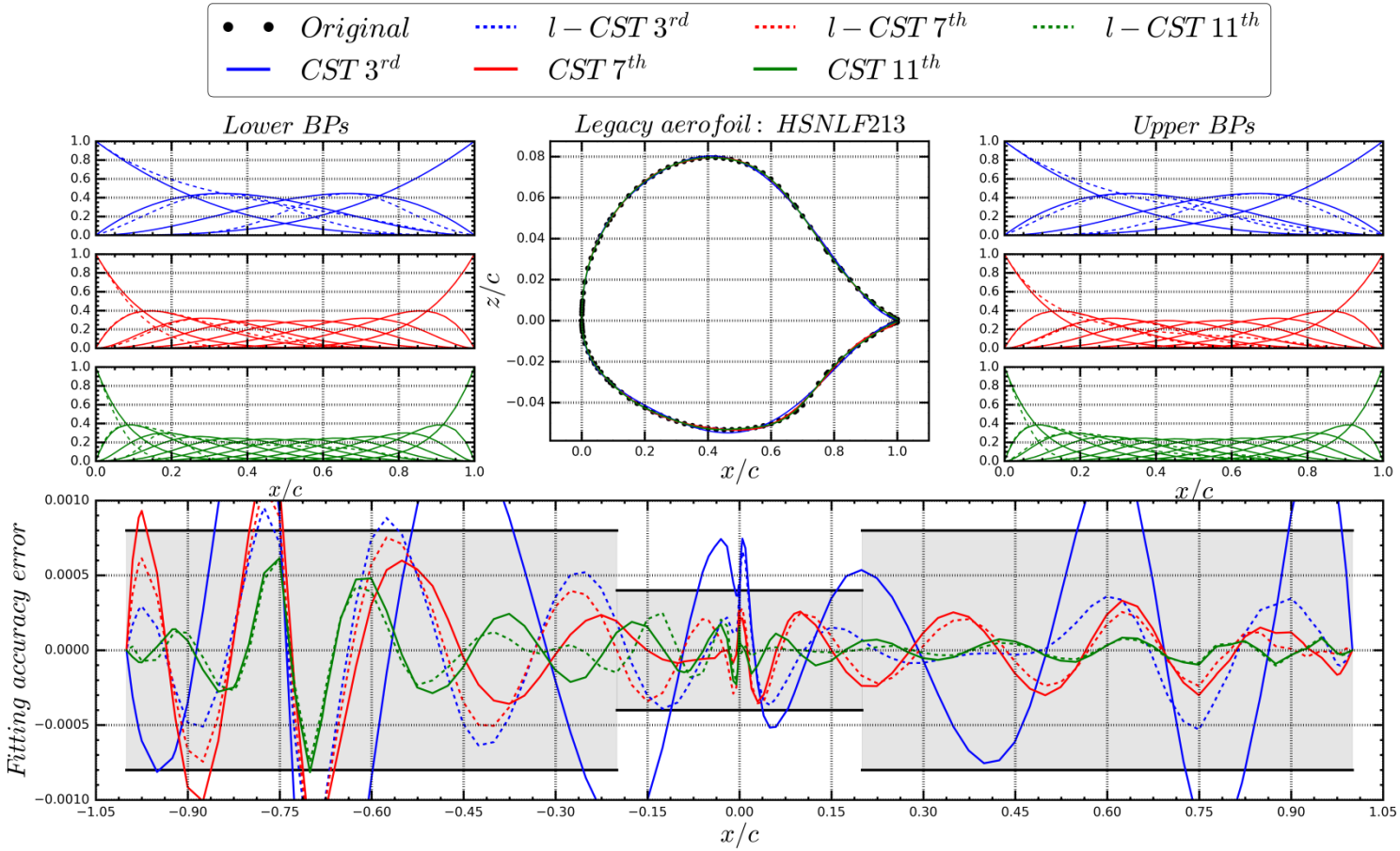


Figure 6.9 Geometric fitting for the HSNLF213 aerofoil.

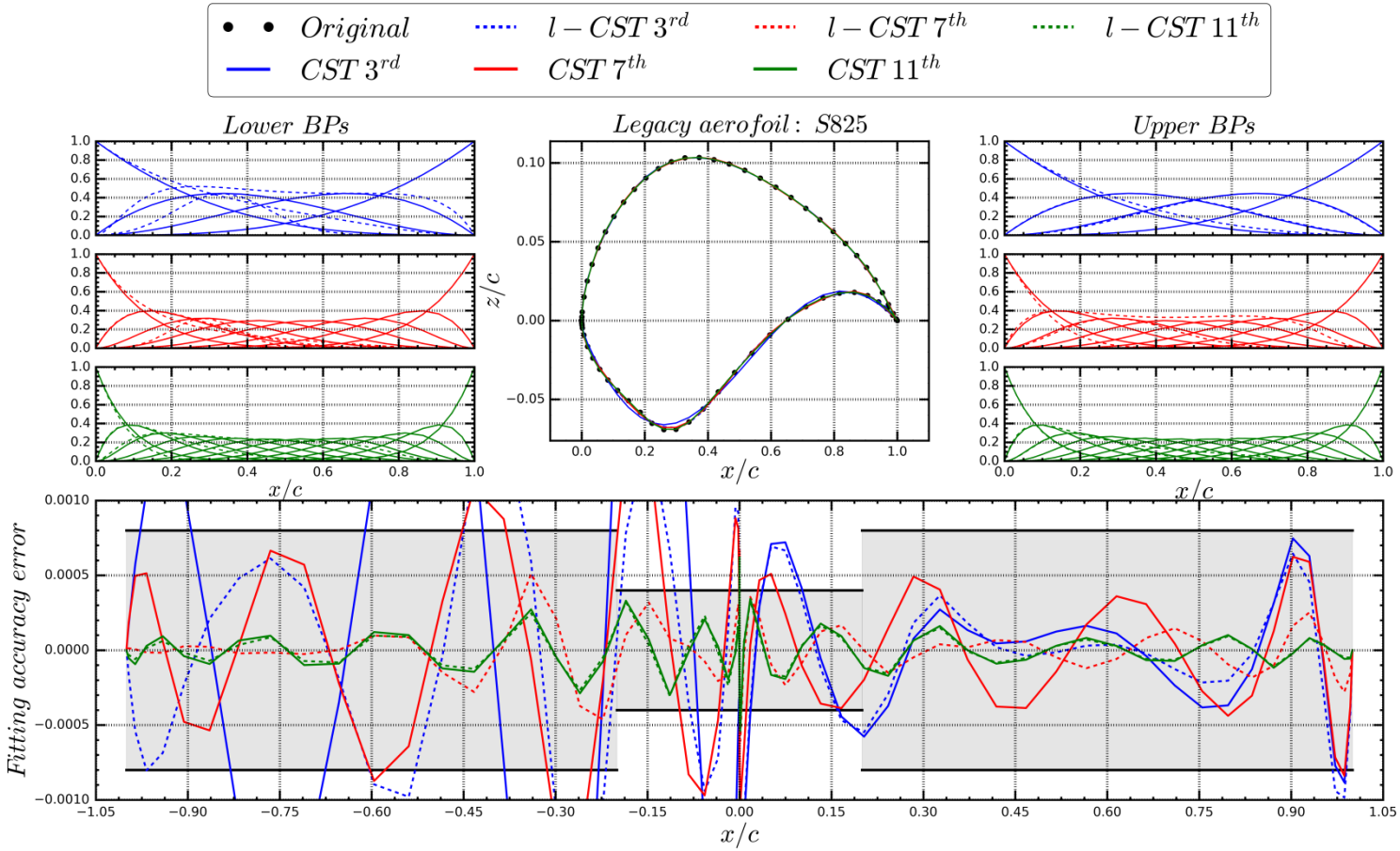


Figure 6.10 Geometric fitting for the S825 aerofoil.

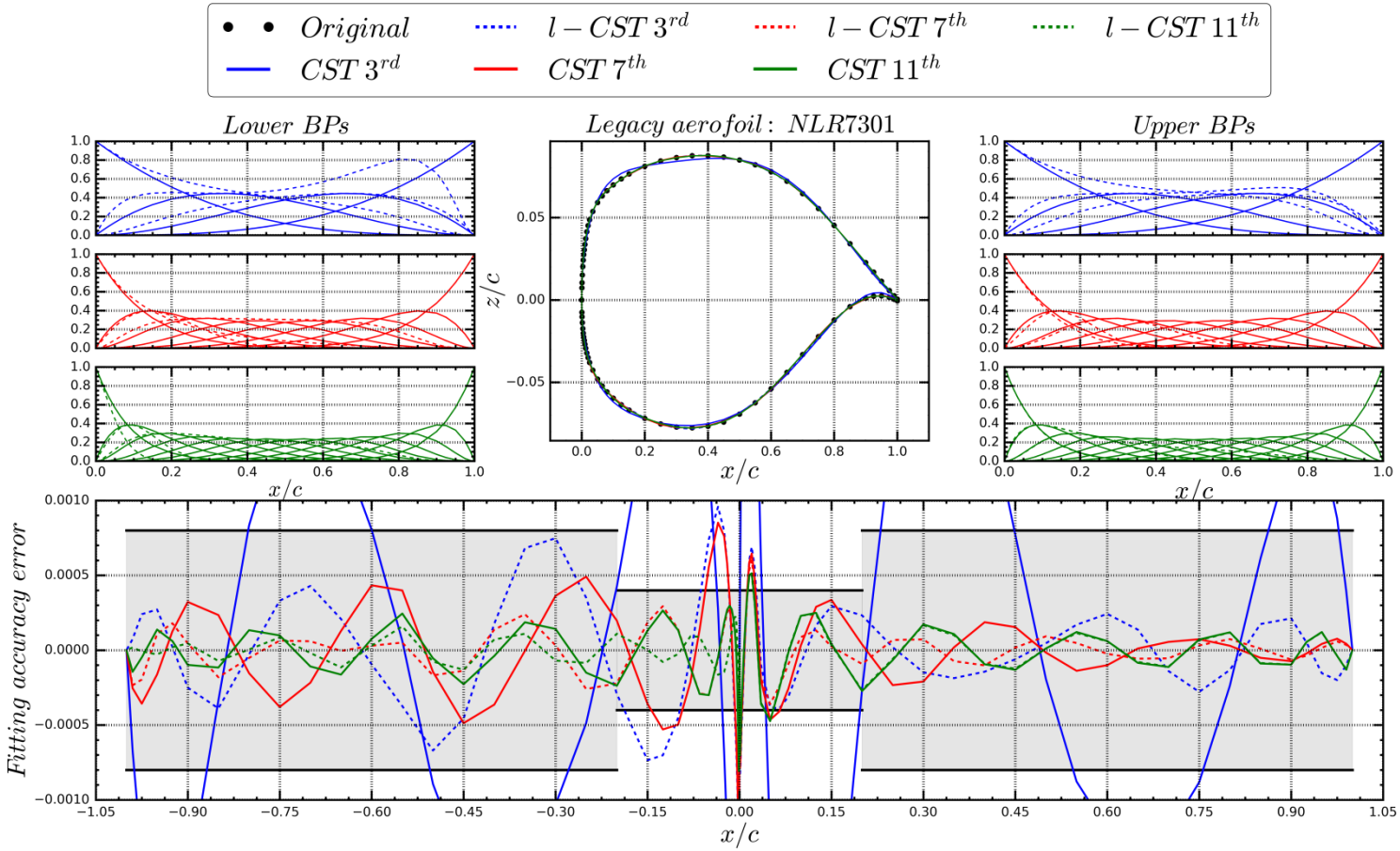


Figure 6.11 Geometric fitting for the NLR7301 aerofoil.

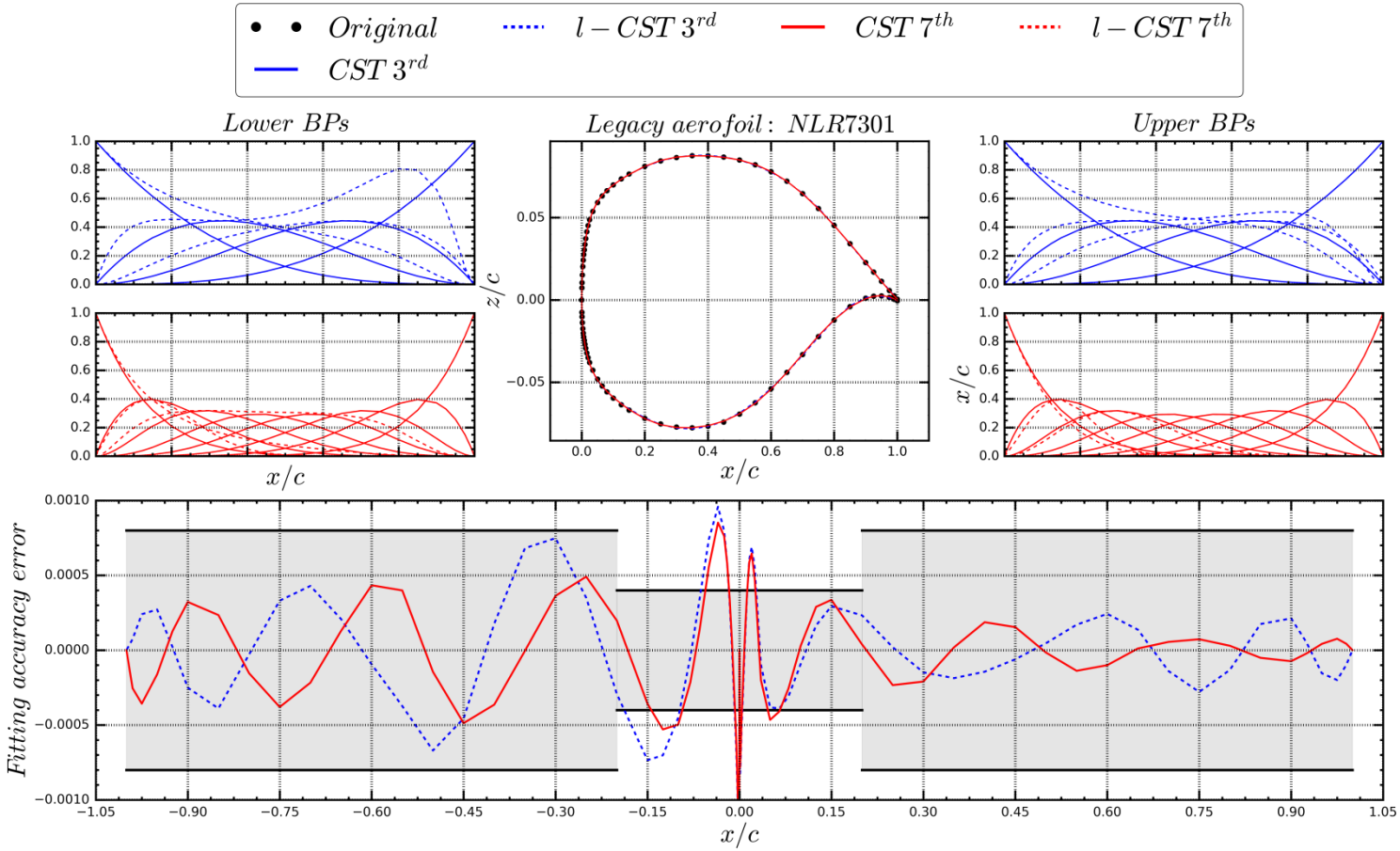


Figure 6.12 Geometric fitting comparison for the NLR7301 aerofoil using the same number of free variables.

6.5.5. Importance of Geometric Fitting for the Aerodynamic Coefficients

The RAE 5243 aerofoil [132] was chosen to study the impact of the *l*-CST retrofitted profile on the aerodynamic coefficients and the pressure distribution. The flow solution was computed at an AoA equal to zero and a Mach number equal to 0.68, using the DLR TAU-Code flow solver and the mesh described in Chaps. 3.6 and 5.4.1 respectively.

It is known that the leading edge and the shock region are one of the areas where the flow experiences high gradient. In these regions, small geometric variations can have a large impact in both aerodynamic coefficients and pressure distribution. The trailing edge also experiences a high gradient, however since this is more a singularity point rather to be a continuously varying feature, is difficult to use for comparison purposes.

It is evident that the *l*-CST outperforms the CST in terms of both pressure distribution and aerodynamic coefficients recovery shown respectively in Fig. 6.13 and Tab. 6.2. The 7th degree CST curve tested misses significantly where the shock starts, whereas the 7th degree *l*-CST is much more accurate in reproducing the same feature. The same conclusion can be also drawn by looking at the leading edge pressure distribution depicted in Fig. 6.14. In this last figure, in particular, is evident the oscillatory behaviour of the fitting due the BP basis used. This situation is particularly important in an optimisation loop, at the first iteration when the parameterisation is called to retrofit a surface mesh nodes created accordingly to the RANS requirements. These last, do not necessarily coincide with the geometric slope/curvature requirements and as a result there is very often a mismatch between the aerodynamic coefficients computed on the original non-parameterised profile and the retrofitted parameterised shape.

Table 6.2 Aerodynamic coefficients comparison.

Parameterisation	C_d	ΔC_d[%]	C_l	ΔC_l[%]	C_{my}	ΔC_{my}[%]
Original mesh	**0.008**72	**Datum**	**0.388**17	**Datum**	**-0.091**8	**Datum**
CST 7th (retrofitted mesh)	**0.008**84	+1.376	**0.388**43	0.0669	**-0.09**21	-0.326
l-CST 7th (retrofitted mesh)	**0.0087**4	+0.229	**0.388**06	0.0283	**-0.091**8	0

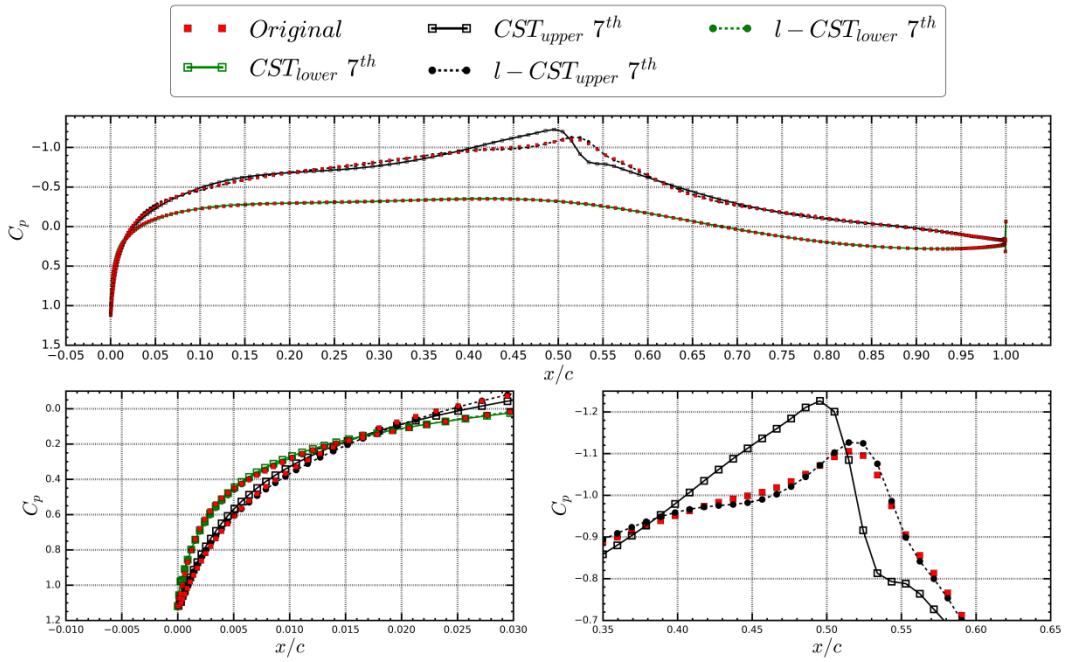


Figure 6.13 Coefficients of pressure comparison.

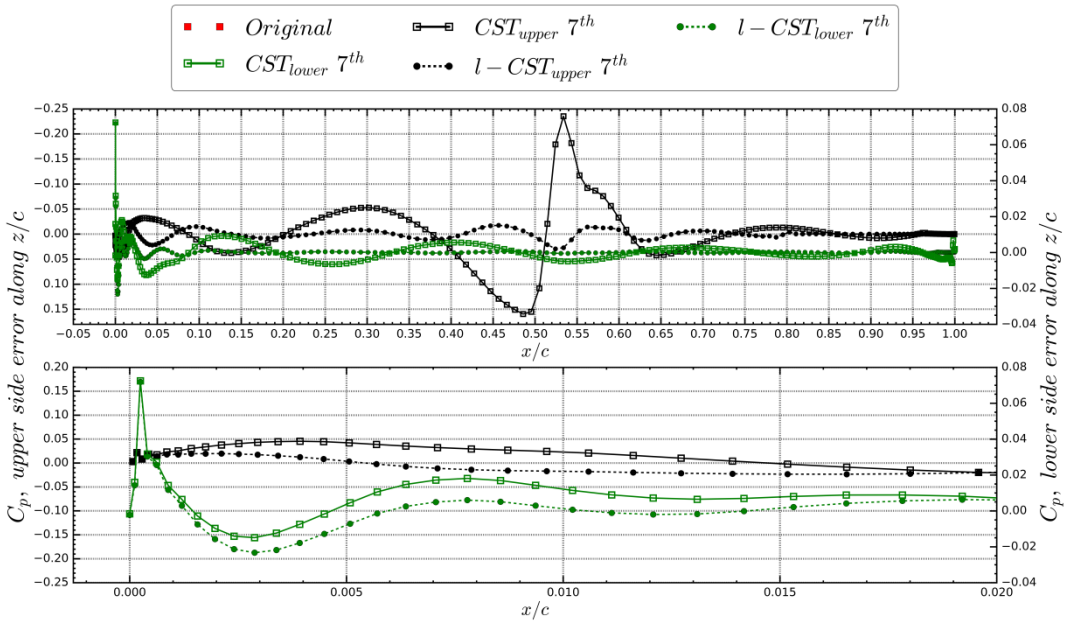


Figure 6.14 Comparison of the coefficients of pressure error.

6.6 Smoothness Issues

There are generally two broad general approaches available to control the surface mesh updates, namely parameterisation or free-nodes approach. Their major differences were discussed in Chap. 6.1, but here the discussion is focused on their smoothness properties. It is known that, whenever parameterisation is used, both shape and gradient are smooth (see also Fig. 6.15). On the other hand, whenever the free-nodes approach is used neither the shape nor the gradient is guaranteed to be smoothed. In this chapter, it is desired to investigate more this matter.

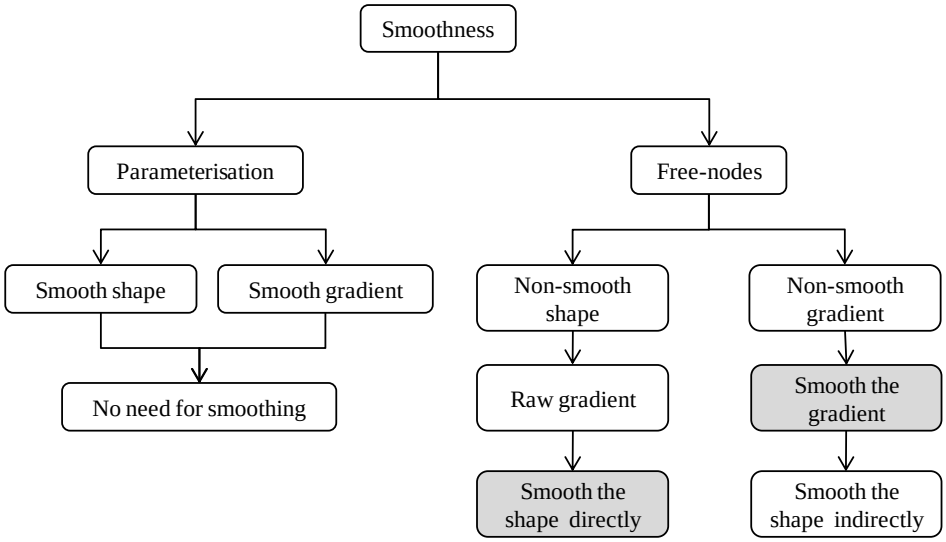


Figure 6.15 Parameterisation versus free-nodes approach comparison from the point of view of shape and gradient smoothness.

Referring to Fig. 6.15, whenever the free-nodes approach is chosen, one can either smooth the shape or the gradient as suggested by Jameson and Kim [9] and Mohammadi [10] respectively. Let us consider the first approach where the gradient is generally smoothed in 2D using the Sobolev approach [9] or in 3D using its extension termed Laplace-Beltrami operator [176]. By doing so, the gradient is effectively changed arbitrarily because one has to choose the smoothing factor beforehand. This, in turn, changes the gradient direction, but improves significantly the convergence. This guarantees within a certain limits a smoothed shape because the smoothing applied is a second order operator and the gain in the regularity w.r.t. the raw gradient is of two

orders. For this reason, the shape is smoothed indirectly because the modified gradient is fed to the optimiser which in turn provides the DV updates.

On the other hand, whenever the shape is smoothed directly, the shape is regularised using the Sobolev operator starting directly from an unsmoothed geometric. In other words, the gradient is left unchanged and is fed directly to the optimiser which in turn provides the DV updates.

The bottom line is that, if it is desired to have a final smoothed shape, there are three possible strategies: parameterisation or smoothing either the gradient or the shape. Since the first option provides by definition a smoothed shape and gradient at the same time, it is felt that the other two approaches should be investigated separately.

6.6.1 Smoothing the Gradient

The map provided by $\{dI/d\boldsymbol{S}\}$ is a valuable source of information as it maps the objective function against each surface mesh point. However, as depicted in Fig. 6.16, this map is extremely noisy. Therefore, it can be rarely used directly unless a smoothing procedure is used [177]. Wu and Padula [178] noted that the chief reason why one wants to use the maximum design space available is dictated by the knowledge that an optimisation may be badly affected by an insufficient N_{DV}.

It is interesting to list the reasons why for the gradients of aerodynamic nature, when the DVs coincide with the surface mesh nodes, it is difficult to maintain an acceptable smoothness level. Without any loss of generality, poor smoothness level can be traced back to any of the following non-mutually excluding reasons:

- Ill-conditioning of the problem, in form of a poorly conditioned Hessian matrix, may cast the framework in a way that leads to gradients that are not smoothed.

- Protas *et al*. [179] noted that fluid dynamics is characterised by a great variety of length scales, a feature which is also reflected in the output.

- Eppler *et al.* [180] noted that the ordinary gradient is a discretisation of Euclidean inner product and as such lacks of smoothness. Furthermore, Barger [181] concluded that for

aerodynamic derivatives (drag and lift for instance), a smoothed profile is generated only if the free-nodes approach is not used.

- The surface mesh is used to discretise both the near-wall flow and the aerodynamic surface at the same time. In fact, the surface mesh is the result of the volume mesh which is done in accordance to the RANS requirements close to the wall. Unfortunately, the discretisation of the geometry may require a different kind of resolution, mainly driven by the local curvature, which is often overlooked in favour of a better near-wall resolution. Stuck and Rung [182] noted that this leads to situations where, the geometry is described by an exaggerate N_{SM}, making the characteristic geometric length very small which is then reflected to a fine-scale gradients.

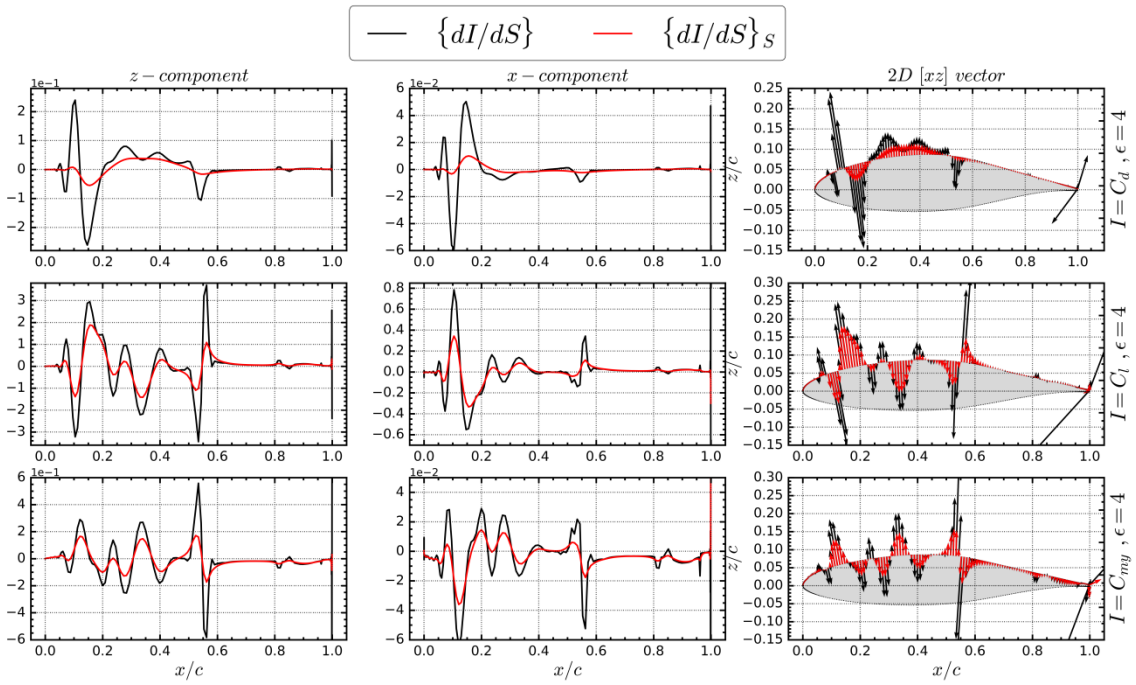


Figure 6.16 Comparison between the raw and the Sobolev (smoothed) gradient.

One solution is to smooth the gradient, a procedure sometimes referred to as gradient preconditioning. One of the most used preconditioning is the Sobolev gradient (as it is known in the literature). The term *Sobolev gradient* was coined by Neuberger [183] and there exist many

applications not necessarily restricted to aerodynamics. This discussion focuses on problem of aerodynamic nature for which its biggest advocates were Jameson and Kim [9].

It can be shown that a smooth gradient can be obtained using the following formulation:

$$[\boldsymbol{S}]\left\{\frac{dI}{d\boldsymbol{S}}\right\}_S = \left\{\frac{dI}{d\boldsymbol{S}}\right\} \tag{6.23}$$

where the subscript S stands for Sobolev and $[\boldsymbol{S}]$ is the smoothing matrix defined as:

$$[\boldsymbol{S}] = \left[1 - \varepsilon \frac{\partial^2}{\partial x^2}\right] \tag{6.24}$$

where ε is the smoothing factor chosen arbitrarily a priori. Matrix $[\boldsymbol{S}]$ is effectively a Laplace operator which can be estimated by second-order central second FDs as suggested by Kim *et al.* [184]. Therefore, considering a 2D case, each direction is smoothed separately as:

$$\begin{aligned}
\left(\frac{dI}{dS_{x,i}}\right)_s - \varepsilon\left(\frac{dI}{dS_{x,i-1}} - 2\frac{dI}{dS_{x,i}} + \frac{dI}{dS_{x,i+1}}\right)_s &= \frac{dI}{dS_{x_i}}, \quad 1 \leq i \leq N_{SM} \\
\left(\frac{dI}{dS_{z,i}}\right)_s - \varepsilon\left(\frac{dI}{dS_{z_{i-1}}} - 2\frac{dI}{dS_{z,i}} + \frac{dI}{dS_{z,i+1}}\right)_s &= \frac{dI}{dS_{z,i}}, \quad 1 \leq i \leq N_{SM}
\end{aligned} \tag{6.25}$$

where $\left(dI/dS_{x,z,i}\right)_s = 0$ at the end points. This can be organised into a matrix as:

$$[\boldsymbol{S}] = \begin{bmatrix} 1+2\varepsilon & -\varepsilon & 0 & . & 0 \\ -\varepsilon & . & . & & \\ 0 & . & . & . & \\ . & & . & . & -\varepsilon \\ 0 & & & -\varepsilon & 1+2\varepsilon \end{bmatrix} \tag{6.26}$$

Which is a tri-diagonal smoothing matrix having dimension equal to $[N_{SM} \times N_{SM}]$ which needs to be inverted to get the value of the smoothed (Sobolev) gradient:

$$\left\{\frac{dI}{d\boldsymbol{S}}\right\}_S = [\boldsymbol{S}]^{-1}\left\{\frac{dI}{d\boldsymbol{S}}\right\} \tag{6.27}$$

Let us considered the RAE 5243 [132] at the same flow conditions described in Chap. 5.4.1. Referring to Fig. 6.16, the elliptic operator smoothes the peaks and fills the valleys but leaves unchanged areas of the aerofoil where the gradient shows already a good degree of smoothness.

Kim *et al.* [185**]** noted that, although the Sobolev approach successfully reduces the gradient amplitude, it can still produces peaky variations even for very large values of the smoothing factor. Furthermore, Wu *et al.* [186] noted that this solution does not guarantee that the same gradient direction is preserved (see Fig. 6.16) and proposed a modification of the Sobolev space capable of maintaining the same steepest descent direction.

6.6.2 Smoothing the Shape

The second order elliptic smoothing operator defined in Eq. (6.26) can also be applied to the shape as suggested by Mohammadi [10]. Considering only the upper z-component of a 2D profile, z_{upper} , the shape is smoothed as follows:

$$[\boldsymbol{S}]\{z_{upper}\}_S = \{z_{upper}\} \tag{6.28}$$

The effect on the shape is shown in Fig. 6.17 for two arbitrary discrete shape updates where it is clear that the application of a smoothing matrix has an effect similar to what is shown in Fig. 6.16.

One interesting aspect is that the neighbouring points are also displaced and the smoothing, if applied to the entire shape, smoothes also the area where no modifications are enforced. Stück and Rung [182] suggested this last case is representative of situations where the initial surface discretisation is not smooth, a consequence of the fact that the surface discretisation follows the RANS requirements at the wall and not the curvature requirements of the shape.

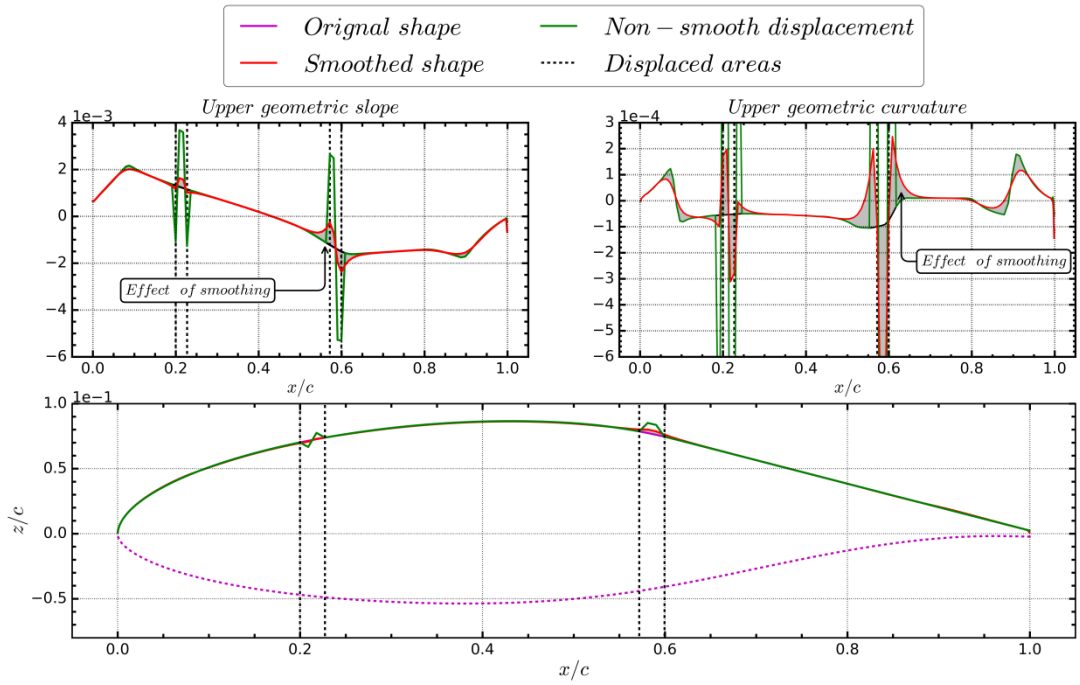


Figure 6.17 Examples of shape smoothing for two arbitrary deformations.

6.7 Summary

In this chapter, two parameterisation methods were presented, namely the FFD and the *l*-CST. In this framework, the gradient and shape smoothness properties were addressed by comparing it against the free-nodes approach.

It was discussed how the FFD, thanks to its local coordinates mapping, allows to retrofit the initial geometry without the need to perform any inverse fitting. A requirement of fundamental importance for the subsequent studies presented in Chap. 7.4 where different mesh movements are tested on the same test case.

On the other hand, it was demonstrated how the *l*-CST method allows a strong on-demand local control capability. The key point is its ability to offer this feature without affecting the advantages generally associated to the CST method. The capabilities of the method was tested against a geometric fitting exercise for different aerofoils ranging from supercritical to wind turbine aerofoils. Therefore, it was demonstrated how the *l*-CST can deliver a better retrofitting using more (i.e. $2 \times N_{CP}$) free parameters but still using the same curve degree. The impact on

the aerodynamic coefficients and resulting pressure coefficient was also analysed. From the results presented, it is clear that local control can be placed in an on-demand fashion along the profile controlling a few parameters. This is done by maintaining the same number of control coefficients and hence the same BP degree which is good for stability.

Within the surveyed parameterisation techniques, the free-nodes approach was given particular attention. This introduces other issues in terms of smoothness both at the shape and gradient level. Two strategies were presented where the Sobolev smoothing was applied directly either to the gradient or the geometry. It was highlighted where the two approaches differ and what are the possible consequences

.

Chapter 7 Aerofoil and Wing Shape Optimisation

7.1 Introduction

Two new techniques were covered from a theoretical point of view in Chaps. 5 and 6, namely the differentiation of the DGM mesh movement and the *l*-CST parameterisation. This chapter proceeds to evaluate their performances in an optimisation loop.

It was discussed how, using the DGM, it is possible to obtain a considerable saving in memory and CPU time requirements. However, the introduction of a new method poses additional research questions that can be addressed only in an optimisation loop. The effect of the supporting box, grid sensitivity extent over the computational domain, consistent versus non-consistent mesh movement are some of them. Furthermore, it is explained how the framework cast as a lift-constrained optimisation is also of fundamental importance.

Sometimes, even if the gradient of a large design space is obtained efficiently, say using the DGM-based adjoint chain described in Chap. 5.3, this does not automatically means it can be effectively used. This is what happens when the free-nodes approach is used, in fact, gradient and shape smoothness are two important issues as described in Chap. 6.6. These are some of the reasons why parameterisation is necessary.

Speaking of parameterisation, the *l*-CST was specifically developed to bridge the gap between maximum explorability for which the free-nodes approach excels, and update smoothness for which any indirect parameterisation method is an appropriate choice. To investigate its properties, the *l*-CST is tested against the free-nodes approach by parameterising a shock-control bump with the *l*-CST in a localised area of the aerofoil. Furthermore, these two cases are also compared against the CST (based on the Bernstein polynomials) to show the consequence of having a poor local support.

The chapter concludes with a summary of the most important findings.

7.2 Optimisation Problem

The most general aerodynamic optimisation problem consists of determining the values of the DV vector such that the objective function, i.e. the drag coefficient ($C_{D,d}$ for a 3D and 2D case respectively), is minimised:

$$I = C_{D,d} \tag{7.1}$$

subject to the equality constraint:

$$C_{L,l} = C_{L,l}^{target} \tag{7.2}$$

where $C_{L,l}^{target}$ is the target lift constraint obtained by letting the AoA be controlled by the flow solver at each design iteration. Therefore, the objective function should be corrected consistently as [187]:

$$I^* = I - \left(\frac{\frac{dI}{d\alpha}}{\frac{dC_{L,l}}{d\alpha}} \right)_{L,l} \left(C_{L,l} - C_{L,l}^{target} \right) \tag{7.3}$$

where in a well-converged flow run the difference $\left(C_{L,l} - C_{L,l}^{target} \right)$ will tend to a null value. Each DV is allowed to move between a lower, D^l, and an upper limit, D^u:

$$D_i^l \leq D_i \leq D_i^u, \quad i = 1, N_{DV} \tag{7.4}$$

The scalar objective function serves as a measure of the fitness of the design process. Two important figures of interest, namely drag and lift coefficients are considered where the former is probably the most interesting for its profound effect on the airlines' budget.

In general, the optimisation problem can be constrained or non-constrained, being the former representative of this aerodynamic shape design framework. The flow and mesh constraints were analysed while deriving the main gradient equation in Chaps. 2.3.2 and 5.3 respectively. Furthermore, the coefficient of lift is maintained fixed. This is considered as an explicit constraint which means that the gradient w.r.t. the objective function must be corrected to take into account the different AoA [187]:

$$\left\{\frac{dI^*}{d\boldsymbol{D}}\right\} = \left\{\frac{dI}{d\boldsymbol{D}}\right\} - \left(\frac{\frac{dI}{d\alpha}}{\frac{dC_L}{d\alpha}}\right)\left\{\frac{dC_L}{d\boldsymbol{D}}\right\} \tag{7.5}$$

This correction prevents the optimiser from reducing the drag by simply reducing the lift which eventually leads the optimiser to go toward a flat-plate like shape.

This thesis focuses on the parametric DVs that control the shape which changes in a continuous manner. In all the subsequent simulations, the surface mesh nodes are updated in the y-direction (z-direction in 3D) only in order not to change the chord and more importantly to avoid mesh elements becoming entangled which is likely to happen if they are allowed to be moved along the x-direction (x and y in 3D) as well. Furthermore, there is also another reason, as noted by Amoiralis and Nikolos [162], the solver is much more sensitivity to a displacement in the y-direction than in the x-direction. This is acceptable if only deformations away from both leading and trailing edges are considered.

7.3 Search Direction

It was mentioned that gradients are computed via the discrete adjoint approach. In Chap. 3, this choice was justified by the type and the N_{DV} being optimised. Computing the gradient is just a step toward the DV updates, in fact, other steps such as Jacobian or Hessian computations followed by a choice of the step to advance each DV are also very important. This is generally referred to as *search direction* and this chapter aims at giving a more thorough discussion on this matter.

There are several search direction strategies available as briefly depicted in Fig. 7.2. These can be briefly classified based on the number of computed derivatives. Zero-order methods require only the evaluation of the function at a point in the design space. An examples of this is the Simplex method [188]. The second type is the so-called *first-order method*, because only the evaluation of the first derivatives is needed. Examples of such methods are the steepest descent method and its improved version called the conjugate gradient method [189]. The third type of strategy available is based on second derivatives and for this reason is called *second-order method*. Physically, the second derivative, i.e. Hessian matrix, represents the local objective

function level set curvature at a point in the design space. This can be evaluated directly (Newton algorithms) or by using an approximation (quasi-Newton algorithms).

The same classification can be made from another point of view: constrained versus non-constrained optimisation problem. First order methods such as the steepest descent are widely used for unconstrained problem. Its advantages are simplicity and the possibility to use partially converged flow and flow adjoint as long as the gradients are smoothed in the Sobolev space as suggested by Jameson and Kim [9]. It biggest drawback, unless the gradient is smoothed, is its slow convergence near the optimum and the fact that it is badly influenced if the DVs are not scaled properly. Examples can be found in Hicks and Henne [2] and Nadarajah and Jameson [28]. Newton methods can accelerate the convergence rate with the added advantage that the DVs do not need to be scaled, but at the cost of computing the Hessian matrix which may become impractical for large cases. On the down side, this requires a fully converged flow and flow-adjoint solution. Quasi-Newton methods can be used as an alternative to solve the memory issue, but they still require a fully converged flow and flow-adjoint solution which can be difficult to attain.

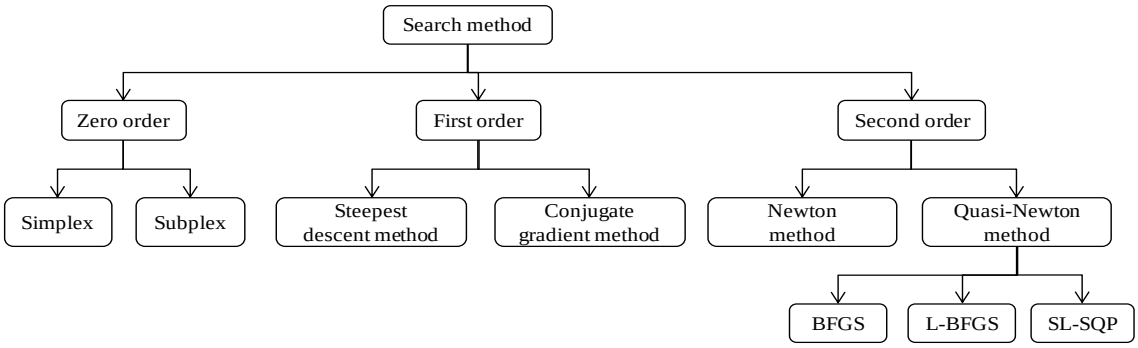


Figure 7.1 Classification of the search algorithms for a gradient-based optimisation.

Before giving an explanation as why the L-BFGS was chosen it is important to mention one important issue. One of the disadvantages of gradient-based search algorithms is that their cost of the function evaluation grows linearly [190] with the problem dimensionality. As the N_{DV} is increased, for instance by two fold, an increase in cost of the same magnitude is expected. The

issue needs to be investigated in the availability of the Hessian and the related memory and time required to build such a matrix.

Given the fact that, second order information is more accurate than first order information, the choice was reduced to two optimisers available in the NLopt library [191], namely SL-SQP (Sequential Least-Squares Quadratic Programming) [192] and L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno). Although, both SL-SQP and L-BFGS use the same update, the first is also capable of enforcing equality and inequality constraints. Considering that, the only constraint used in the present work was enforced explicitly on the gradient as reported in Eq. (7.5) the L-BFGS represented the best choice for the test cases studied in Chaps. 7.4 and 7.5.

7.3.2 The L-BFGS Algorithm

Let us start by constructing a quadratic approximation, P, of the objective function around the DV at iteration k:

$$P(D^{k+1}) = I(D^k) + \nabla I(D^k)^T (D^{k+1} - D^k) + \frac{1}{2}(D^{k+1} - D^k)^T \nabla^2 I(D^k)(D^{k+1} - D^k) \tag{7.6}$$

Let us assume that $\nabla^2 I(D^k)$ is positive definite, thus the minimum of this quadratic approximation can be found by setting the first derivative to zero, i.e. $\nabla P\left(D^{k+1}\right) = 0$, which yields:

$$D^{k+1} = D^k - [\nabla^2 I(D^k)]^{-1} \nabla I(D^k) \tag{7.7}$$

The general DV update for a gradient-based algorithm is given as:

$$D_i^{k+1} = D_i{}^k + \alpha^k p^k \tag{7.8}$$

where α^k is the step size and p^k is the descent direction. Now substituting Eq. (7.6) back to Eq. (7.7) and assuming $\alpha^k = 1$, the following holds:

$$p^k = -[\nabla^2 I(D^k)]^{-1} \nabla I(D^k) \tag{7.9}$$

These equations describe the standard line-search Newton method. The capability to compute and store the entire Hessian makes a clear distinction between Newton and quasi-Newton methods. The former compute the Hessian, whereas the latter approximates its entries.

The direction of improvement used in this work is computed using the L-BFGS optimiser [193]. To explain how the L-BFGS works, it is better to explain how its parent BFGS works. Since computing the Hessian can be expensive, the BFGS [194] approximates its value by considering only first order derivatives:

$$\nabla^2 I\left(D^{k+1}\right) \approx B^{k+1} \tag{7.10}$$

where B is an approximation of the Hessian. At this point what is needed is the inverse of the approximated matrix:

$$\left(B^{k+1}\ \right)^{-1} = H^{k+1} \tag{7.11}$$

This eventually leads to the standard BFGS formula for the inverse of the approximated Hessian matrix:

$$H^{k+1} = \left(V^k\right)^T H^k V^k + \rho^k s^k \left(s^k\right)^T \tag{7.12}$$

where

$$\begin{aligned} \rho^k &= \frac{1}{(y^k)^T s^k} \\ V^k &= I - \rho^k y^k \left(s^k\right)^T \end{aligned} \tag{7.13}$$

This update requires a storage cost bounded by $\mathcal{O}(N_{DV}^2)$. Since the inversion of the Hessian approximation is dense, as the N_{DV} is increased the cost of storing this matrix becomes expensive. The idea behind the L-BFGS is to avoid the storage of the entire approximated Hessian by considering only some vectors that implicitly represent the full Hessian matrix. In formula:

$$
\begin{aligned}
H^{k+1} &= \left(V^k \dots V^{k-\tilde{m}}\right)^T H^0 \left(V^{k-\tilde{m}} - V^k\right) \\
&+ \rho^{k-\tilde{m}} \left(V^k \dots V^{k-\tilde{m}+1}\right)^T s^{k-\tilde{m}} \left(s^{k-\tilde{m}}\right)^T \left(V^{k-\tilde{m}+1} - V^k\right)^T \\
&+ \rho^{k-\tilde{m}} \left(V^k \dots V^{k-\tilde{m}+2}\right)^T s^{k-\tilde{m}+1} \left(s^{k-\tilde{m}}\right)^T \left(V^{k-\tilde{m}+2} - V^k\right)^T \\
&\vdots \\
&+ \rho^k s^k \left(s^k\right)^T
\end{aligned}
\tag{7.14}
$$

where the vector length, $\tilde{m}$, is computed as $\tilde{m} = min(k, m-1)$ where m is provided by the user. An $m = 10$ was used in the present work. The cost of this update is then bounded by $\mathcal{O}(N_{DV})$. The larger m is, the larger the memory requirement but the faster the convergence rate. Examples of applications can be found in Elliot and Peraire [195] and Rumpfkeil and Zing [196].

7.4 3D Wing Optimisation

7.4.1 Consistent versus Non-consistent Approaches

A discrete adjoint aerodynamic shape optimisation involves the linearisation of the entire chain. By doing so, and assuming the shape is parameterised, three types of sensitivities are needed, namely flow, grid and shape sensitivities. In this framework, depending on the way the chain is constructed, there are four possible inconsistencies that may be highlighted.

The choice of the continuous adjoint over the discrete adjoint approach introduces the first inconsistency (see Fig. 7.2), because there is a discrepancy between the adjoint-gradient and the discretised objective function. However, as noted by Anderson and Venkatakrishnan [35] this error disappears as one refines the mesh. On the other hand, if the discrete adjoint approach is chosen, the discretised gradient is consistent with the discretised objective function by definition. In this last case, particular attention needs to be paid [35] to the way the grid sensitivity is computed, whereas for the continuous approach there exists a way (surface integral) to avoid the computation of this term as briefly depicted in Fig. 7.2.

Mesh sensitivity refers to both volumetric and surface mesh sensitivities as described in Chap. 2.4.1. Volumetric sensitivity is related either to the mesh movement or to the grid generator used depending on the strategy chosen to deform the mesh. If the second strategy is used a second inconsistency in the computation of the gradient is introduced. In fact, mesh

regeneration changes the point discretisation, especially for unstructured meshers, which in turn inevitably leads to different discretisation error [100, 101].

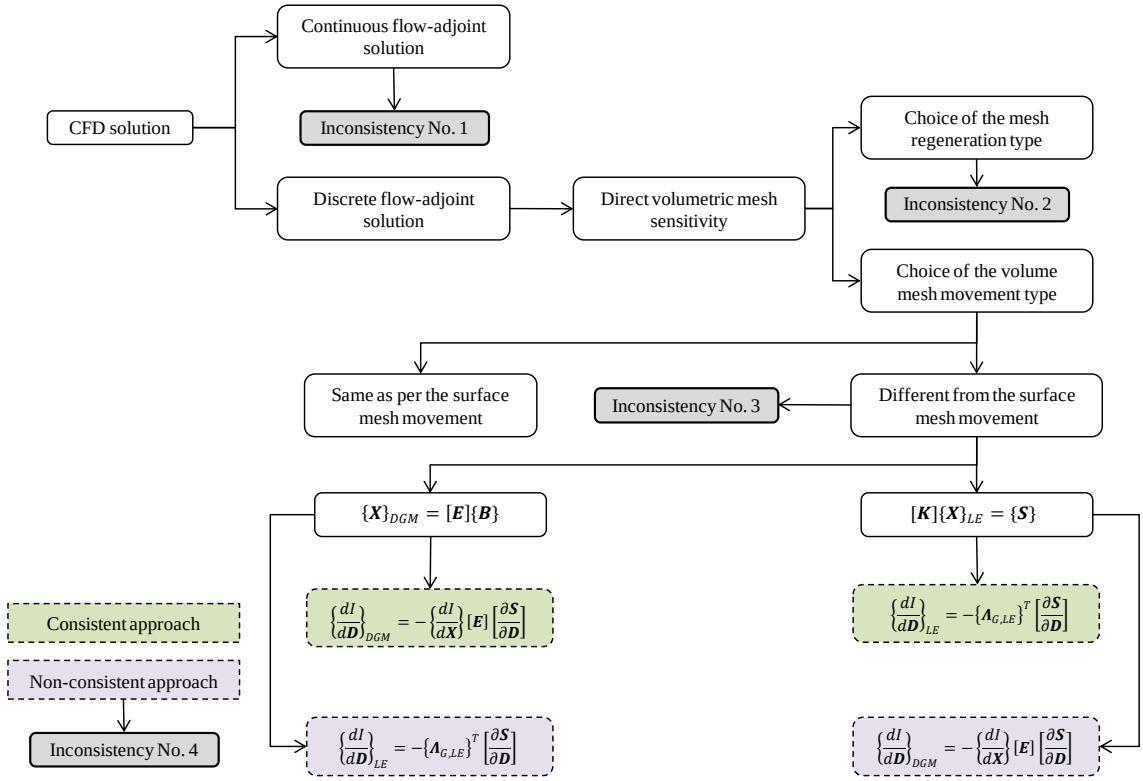


Figure 7.2 Four possible inconsistencies in the discrete adjoint chain.

The analytical availability of the sensitivity of the surface mesh w.r.t. the parametric DVs depends on the choice of the shape parameterisation which is generally easily differentiable. To this regards, there is another third subtle inconsistency (see Fig. 7.2). In fact, the first volume cell has some points on the surface which effectively define the shape and the other(s) in the volume domain which directly play an important role for the boundary layer resolution. For this cell, it is possible to use either a consistent strategy where both surface and volume meshes are updated by the same algorithm or not. There are two important applications published in the literature where an integrated fully-consistent surface-volume mesh deformation is used. The first one was proposed by Morris *et al.* [79] where they used the RBF method, whereas the other was proposed by Hicken and Zingg [38] where the authors used the B-splines method instead.

Historically, any mesh movement has been available before its linearisation. This has left open the question as to whether the non-consistent approach, thus introducing the fourth inconsistency, is indeed an option or it is something that must be avoided. This could lead to situations where one has to use either a different mesh movement or a different grid sensitivity strategies, for a series of reasons where robustness and/or memory issue are the most common. In this situation, the non consistent approach would offer more flexibility.

This study investigates numerically the effects of using different non-consistent (grid) approaches in an optimisation loop and compares the result against the consistent approaches. This is interesting because, should any large difference be highlighted, this would mean that one cannot choose the non-consistent approach without incurring a deviation (in the objective function) w.r.t. the fully consistent approach. On the other hand, in absence of an appreciable difference, one would be allowed to choose based on efficiency or robustness grounds.

Some of the most important applications found in the surveyed literature are reported in Tab. 7.1. This shows a tendency to prefer a consistent approach over a non-consistent approach. These approaches were investigated as standalone cases, but a thorough comparison of what happens when different combinations are considered (see Fig. 7.2) has not been published yet.

From Eq. (5.4), after solving the flow and if necessary also the mesh adjoint linear system of equations, the final sensitivity can be written in general form as:

$$\left\{\frac{dI}{d\boldsymbol{D}}\right\} = \{\boldsymbol{\Lambda}_G\}^T \left[\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{D}}\right] \tag{7.15}$$

From here it is understood that, in the discrete adjoint approach, the differentiation of the mesh movement residual is necessary. However, this can happen in a consistent or non-consistent manner w.r.t. the chosen mesh movement as depicted in Fig. 7.2. In order to provide some guidelines, the following research points are addressed:

- Does the non-consistent approach lead to different optima?
- Are the initial gradients and DV updates the same?
- At which iteration substantial/slight deviations are registered? Do they stay constant or diverge?

- Are these deviations (if present) due to different mesh sensitivities or different mesh movements?
- Which approach, between different mesh movements and grid sensitivities introduces the largest deviation?

To answer these questions, each single step of the chain is analysed as follows:

- Distribution over the computational domain of the gradient $\{dI/d\boldsymbol{X}\}$
- Effect of different mesh movements: LE and DGM
- Effect of different grid sensitivities: $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE,DGM}$
- Product between $\{dI/d\boldsymbol{X}\}$ and $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$

Each one of these points should not be considered as a standalone factor, but an integrated part of the optimisation chain.

Table 7.1 Mesh movement and sensitivity approaches used in the literature.

Authors	Mesh movements	Mesh sensitivity	Consistency
Zhu and Qin [197] Bobrowski *et al.* [198]	RBF	LE	Non consistent
Nemec and Zingg [69] Martins *et al.* [54]	Algebraic method	Algebraic method	Consistent
Jakobsson and Amoignon [199]	RBF Laplace smoother	RBF Laplace smoother	
Nielsen and Park [37] Nambu *et al.* [102]	LE	LE	
Mavriplis [200] Maute *et al.* [201]	Spring analogy	Spring analogy	
Burgreen and Baysal [202] Le Moigne and Qin [67]	Algebraic method	Algebraic method	
Truong *et al.* [100]	LE Algebraic method	LE Algebraic method	

7.4.2 Setup of the Optimisation Framework

The workflow used in all the following simulations is schematically described in Fig. 7.3. It consists of one flow solution, two adjoint solutions, one search direction and if necessary a mesh adjoint solution. This is repeated until convergence has been reached. There are two points in the framework depicted in Fig. 7.3 where the choice of maintaining consistency or not appears. The first one is when one has to choose which mesh movement between LE and DGM, whereas the second one is when one chooses which mesh movement to differentiate between DGM and LE.

For each case reported in Tab. 7.2, there are a total of 10 DVs. Their spatial positions are depicted in Fig. 7.4 and were chosen in order to maintain the planform fixed. These are the FFD CPs which control the upper surface only and are allowed to move within $\mp 25\%$ of its original value. This relative low N_{DV} was necessary in order to be able to concisely keep track of both DV updates and DV gradients.

To investigate the effect of grid extent differentiation and consistent versus non-consistent approaches, a series of test cases on the same mesh (ONERA M6) were devised as briefly listed in Tab. 7.2. For each case comments are made on the convergence history, gradients (corrected for the change in AoA), DV updates and where possible a justification is provided.

7.4.3. Grid Sensitivity Differentiation Extent

The goal of this section is to investigate the effect of different grid differentiation extents in an optimisation loop. These are tested by employing a series of different supporting box heights as shown in Fig. 7.5. Five cases are tested, four of them use the supporting box placed at roughly 30, 50, 100 and 150% of its original distance which roughly corresponds to the MAC (Mean Aerodynamic Chord), whereas one of them does not use the supporting box and the differentiation is performed over the entire volume domain.

As can be seen from Figs. 7.6 and 7.7, the introduction of the supporting box causes a negligible difference in the final value of the optima up to a point where it starts to deviate significantly as shown for the case where supporting box was placed at 30% of the original height. This happens because, such a low supporting box height is cutting the physical information carried by

gradient $\{dI/d\boldsymbol{X}\}$ as described in Scenario No. 1 depicted in Fig. 5.14 (top). However, these differences (not considering the $\Delta\boldsymbol{X}_{mDGM}\,[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM,30\%s.b.}$ case) are not necessarily an error as they fall below the CFD accuracy threshold generally estimated in not less than half a DC (Drag Count, 10^{-4}). This study proved that the driving quantity is the gradient $\{dI/d\boldsymbol{X}\}$ and that its non null values are effectively distributed very close to the wall. Furthermore, these results prove that a smaller subset of the entire Jacobian $[\partial\boldsymbol{X}/\partial\boldsymbol{S}]$ can be effectively used.

Table 7.2 List of test cases and addressed research questions.

Chapter	**Research question**	**Mesh movement type**	**Grid sensitivity**	N_{DV}	**Consistency**
7.4.3	Grid sensitivity differentiation extent	oDGM	oDGM	10	Consistent
		mDGM	mDGM		
7.4.4	Establish a reference value	mDGM	mDGM	10	
		LE	LE		
7.4.5	Quantify the error introduced by using the same mesh movement, but different grid sensitivities	LE	mDGM	10	Non consistent
			LE		Consistent
		mDGM	mDGM	10	Consistent
			LE		Non consistent
7.4.6	Quantify the error introduced by using different mesh movements, but the same grid sensitivities	LE	LE	10	Consistent
		mDGM			Non consistent
		LE	mDGM	10	Non consistent
		mDGM			Consistent

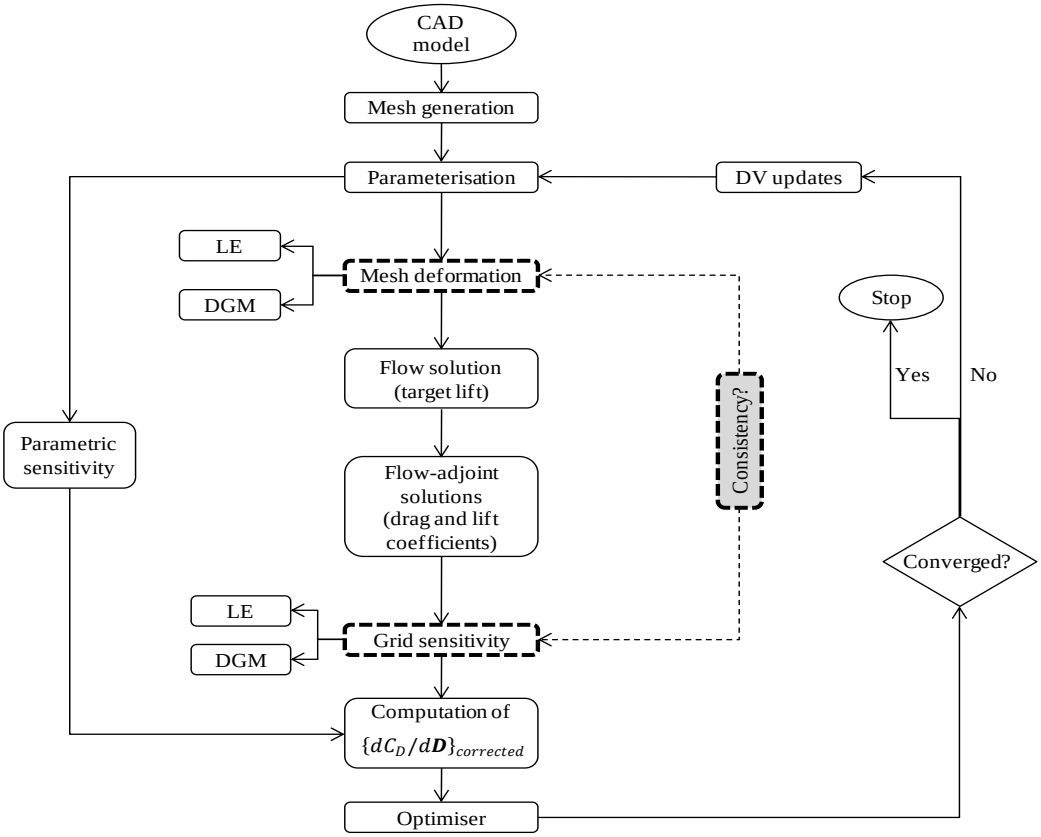


Figure 7.3 Workflow for the consistent versus non-consistent grid sensitivity test cases.

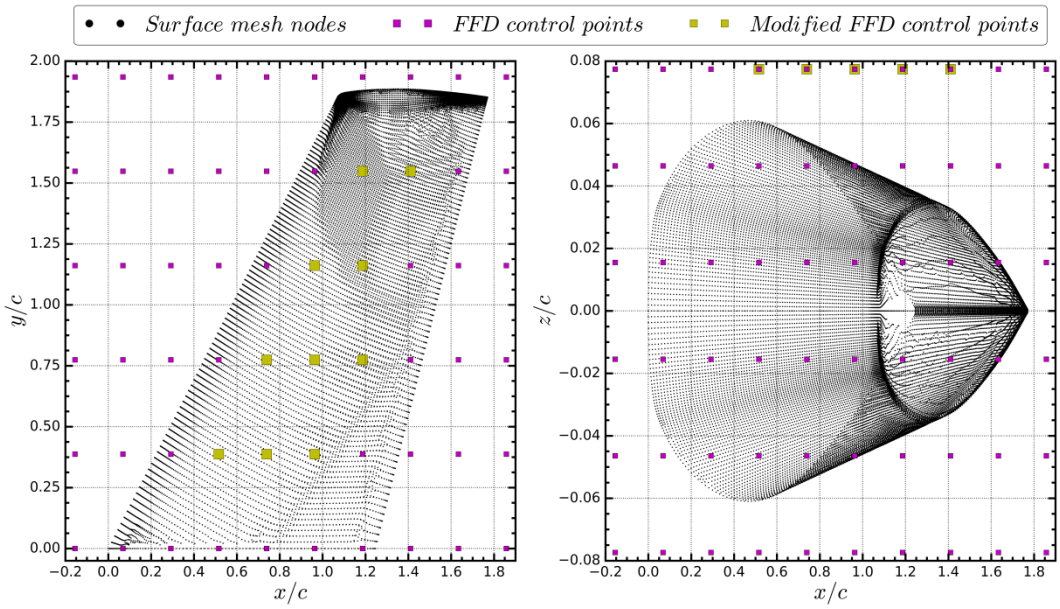


Figure 7.4 FFD box constructed around the ONERA M6 wing.

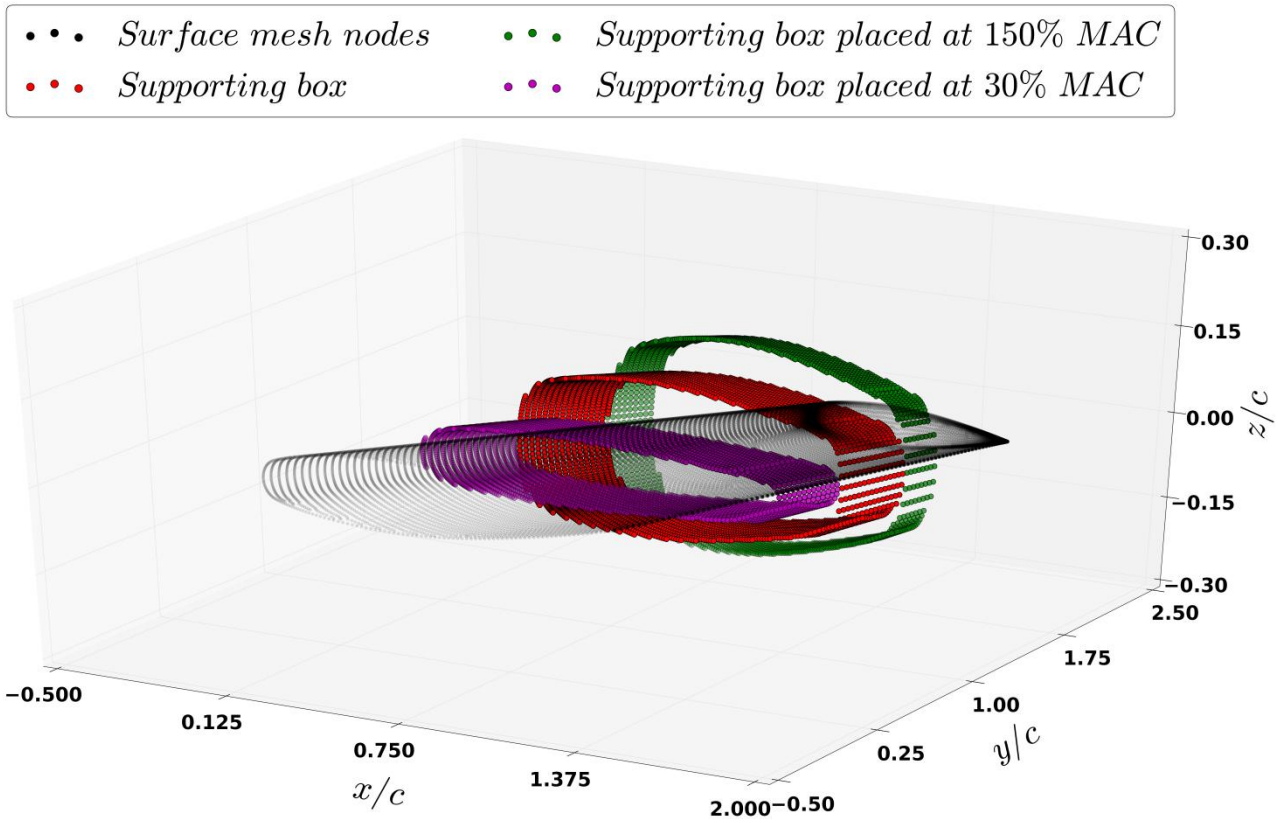


Figure 7.5 Isometric view of different supporting box spatial positions for the ONERA M6 wing.

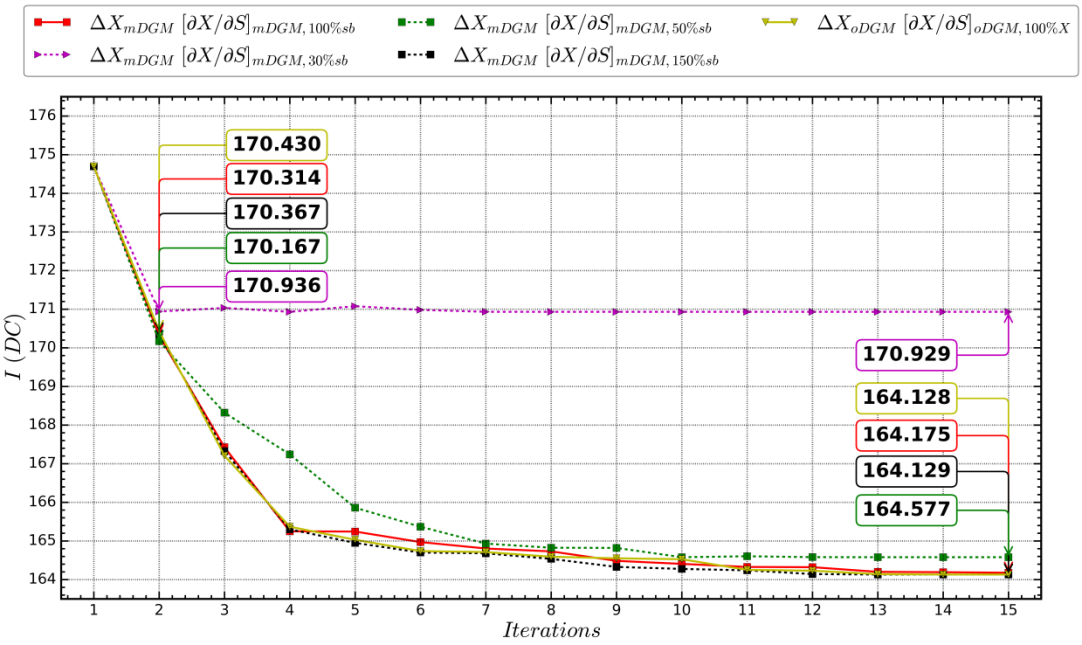


Figure 7.6 Convergence history comparison for different mesh size differentiations.

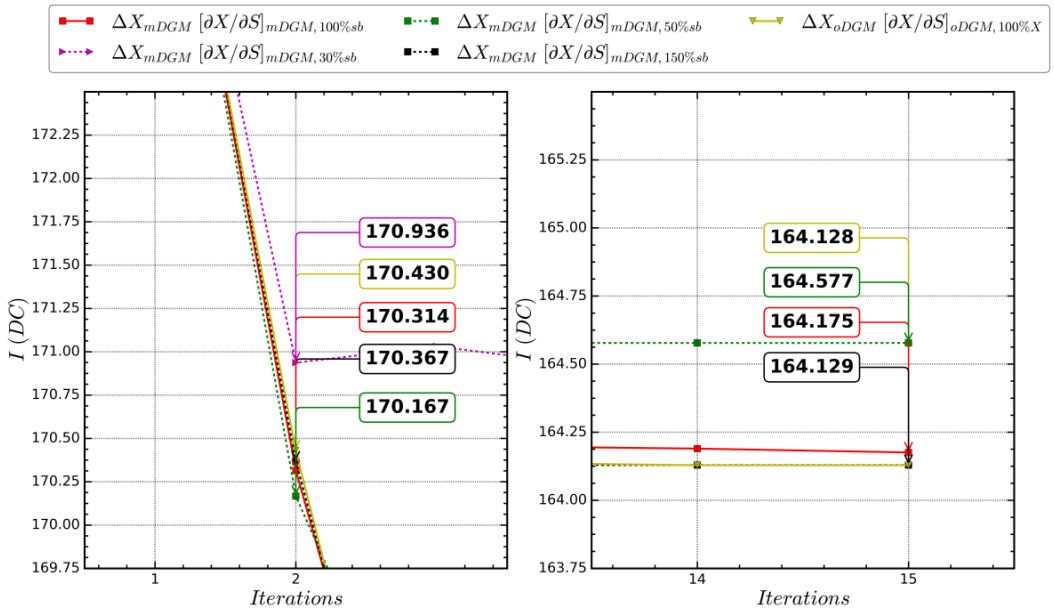


Figure 7.7 Convergence history close up view on the second and last iterations.

7.4.4 Fully Consistent Approaches

This section tries to establish two reference cases to compare the non-consistent approaches against. Referring to Figs. 7.8 and 7.9, the optimisations featuring a consistent approach (listed in Tab. 7.2) are started from the same initial mesh, and at iteration No. 2, a slight difference is registered which is then maintained till the end of the simulation. The comparisons of the original and optimised pressure coefficient along with a few sampled stations are depicted in Figs. 7.10 and 7.11 respectively.

Interestingly, when the DV gradients (see Fig. 7.12) are analysed, they do not have the same starting values and the same paths to convergence. However, this is not reflected in deviations in the initial DV values depicted in Fig. 7.13, although there is a deviation in the paths to convergence. In this case, the differences are solely due to the different mathematical formulations of both mesh updates and mesh sensitivities. However, although some small differences are present in both gradients and DV updates, in the context of a lift-constrained optimisation, these do not lead to different optima.

Jakobsson and Amoignon [199] compared two consistent approaches, namely RBF and Laplace smoothing mesh movement in a optimisation loop. They conclude that, the different minima are due to the different mesh movements, but no comments were made for either the gradients or on the DV updates.

Furthermore, there is also another reason why different deformed grids are not introducing large deviations. In Eq. (5.36), the pressure and viscous components do not evenly contribute to the gradient and this depends on the flow regime. In fact, Widhalm *et al.* [36] established that for subsonic flow, the gradient is mainly influenced by the viscous component, whereas for transonic flow the gradient is mainly influenced by the pressure component. Subsequently, Truong *et al*. [100] investigated further this matter in relation to different mesh movements and established that, at subsonic flow regime, a change in the near-wall mesh orthogonality affects the subsonic case most where the viscous contribution is the largest. On the other hand, for the transonic regime, the pressure part, due to the presence of shocks, contributes most to the gradient and is less sensitive to a change in the deform mesh orthogonality index.

7.4.5 Effect of Non-consistent Mesh Sensitivities

The goal of this section is to numerically investigate what happens when the same mesh movement is used in conjunction with different grid sensitivities. This is representative of situations where another linearised mesh update strategy, different from the one used to move the mesh, is either too memory or computationally expensive.

Referring to Figs. 7.8 and 7.9, the optimisations are started from the same initial non-optimised geometry and at iteration No. 2, a slight difference is registered in terms of computed drag coefficient which is maintained up to the end of the optimisation. On the other hand, referring to Fig. 7.12, it is clear that, all the DV gradients start from different values although not all have the same quantitative deviations. These offsets are not maintained constant and for some DV gradients, a substantial change larger than the initial deviation is registered. However, these deviations are not reflected in the DV updates (see Fig. 7.13), instead both cases have the same DV starting values. Furthermore, a close look of the DVs' paths reveals that only for DVs No. 5-7 there are noticeable deviations. It is therefore concluded that the small difference registered in the gradient are not large enough for the optimiser to compute different DV updates.

The initial deviations in the gradients are initiated by the different grid sensitivities. As per the case presented in Chap. 7.4.4, these do not lead to large differences in the optima. The gradients computed using two different grid sensitivity strategies are therefore called *direct factors*, as depicted in Fig. 7.14, because they are the first in temporal order to introduce a change in the chain. From the third iteration onward, the other factors (updated DVs and deformed mesh) also become *direct factors* because all together are indeed introducing a small change in the chain.

7.4.6 Effect of Non-consistent Mesh Movements

The goal of this section is to numerically investigate what happens when the same differentiated mesh movement is used in conjunction with different mesh movements. This is representative of situations where another mesh movement, different from the one differentiated, is deemed either more robust or less computationally intensive.

Referring to Figs. 7.8 and 7.9, as for the case described in Chap. 7.4.5, the optimisations are started from the same initial non optimised mesh, but at iteration No. 2, a very small difference

is registered in terms of computed drag coefficient which is maintained up to the end of the simulation. Referring to Figs. 7.12 and 7.13, it is clear that, in contrast to the cases reported in Chap. 7.4.5, all the DV gradients start from the same values, a consequence of using the same grid sensitivity strategy, and the DV updates show no deviations, which in turn is a consequence of having the same gradient update. Furthermore, all the DVs follow the same path to convergence with some minor deviations.

The small differences registered at iteration No. 2 are due to the different mesh movements used that introduce a deviation in the near-wall mesh orthogonality even if the surface mesh update is the same at iteration No. 1. Nevertheless, this is not enough for the flow solver to predict a large difference in the value of the objective function. This is why, the deformed mesh becomes the *direct factor* as it is the first, in the time-line history, to initiate the small negligible deviations, whereas the gradients and updated DVs are *indirect factor*. As per the case described in Chap. 7.4.5, from the third iteration onward they all become direct factors as depicted in Fig. 7.14

Although these small differences do not lead to a large deviation in the final optima, an investigation was performed to prove the hypothesis that different near wall orthogonality indices are causing a different residual error (which is the reflected in a small deviation on the objective functions). The flow adjoint is an useful tool in this case as it provides the sensitivity of the objective function w.r.t. the residual [32] (see Eq. (5.35)). Only the first element (corresponding to the density residual) of the adjoint vector for the drag coefficient is depicted in Fig. 7.15. However, the same analysis also applies to the other five state variables and for the lift coefficient as well. By taking the difference of these two flow-adjoint vectors one visualises the deviation in computing the objective function due to the different residuals which in turn can be traced back to the different deformed mesh orthogonality indices:

$$\{\boldsymbol{\Lambda}_F\}_{(\Delta \boldsymbol{X}_{LE},\Delta \boldsymbol{X}_{mDGM}\ )-[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}} = \left\{\frac{\partial C_D}{\partial \boldsymbol{R}}\right\}_{\Delta \boldsymbol{X}_{LE}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}} - \left\{\frac{\partial C_D}{\partial \boldsymbol{R}}\right\}_{\Delta \boldsymbol{X}_{mDGM}\ [\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}} \tag{7.16}$$

As can be seen in Fig. 7.15 (bottom), this hypothesis is confirmed by the non-null value of the adjoint vector plotted accordingly to Eq. (7.16). The difference is two orders of magnitude lower which is a small change, but nevertheless not null.

In the discussion presented above, it was concluded that small differences in the mesh can actually be detected and studied by plotting each variable in the flow adjoint vector. Now, by listing in Tab. 7.3, the two mesh metrics introduced in Chap. 4.7.4, it is shown that there is indeed a small change also in the quality of the deformed mesh. However, this time the evaluation is not computed on the second iteration but on the last one in order to investigate how much the different mesh movement strategies have affected the grid quality.

Table 7.3 Mesh quality metrics evaluated on the last iteration mesh.

Strategy	**Mesh movement**	**Mean mesh quality**			**Dihedral angle**	
		q_i	q_i^{min}	q_i^{max}	θ_i^{worst}	No. $\theta_i < 5°$
Baseline		0.797843	0.0158565	0.999173	2.07613	25
$\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	mDGM	0.80007	0.0218816	0.999173	2.07613	25
$\Delta \boldsymbol{X}_{LE}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$		0.797812	0.0158565	0.999173	0.52355	41
$\Delta \boldsymbol{X}_{LE}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$	LE	0.797853	0.0158597	0.999173	0.523653	41
$\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$		0.797853	0.0158596	0.999173	0.523652	41

Contrary to what was said regarding the mean quality mesh metric in Tab. 4.9 (Chap. 4.7.4), Tab. 7.3 shows a slight variation also in the mesh quality metrics. This can be explained with the fact that, the deformations being analysed here, contrary to those used in Tab. 4.9, concern the entire upper wing.

The mean quality mesh stays almost constant with small variations in the range of 10^{-5}. This is probably due to the small variation in the minimum value registered as the maximum mean quality mesh stays constant for all the cases reported in Tab. 7.3. Interesting, for the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ case, the mean quality mesh is slightly improved. This is a direct consequence of the higher value registered in the minimum mean quality mesh metric.

However, the change is not large and this value, as per the case reported in Tab. 4.9, does not highlight any interesting trend.

On the other hand, as it was previously concluded in Chap. 4.7.4, the dihedral angle is a much better metric to study small changes in the deformed mesh. For all the cases reported in Tab. 7.3, with the exception of the fully consistent $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ case, the worst dihedral angle dropped from ∼2.07 to ∼0.52. This is also reflected in an increase in the number of elements featuring a value lower than five which increased from 25 to 41. Furthermore, the fact that $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ approach features a better overall mesh quality indices can be explained by noticing that, for these small deformations, the adoption of the supporting box is more beneficial than the LE in maintaining the quality of the mesh to an acceptable level.

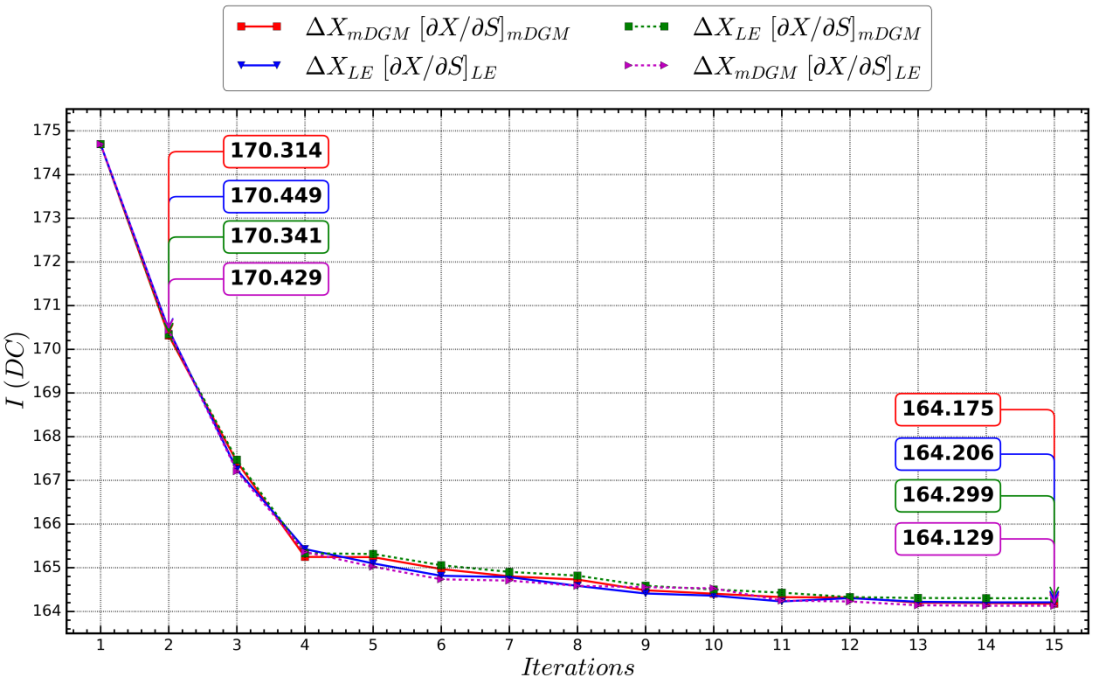


Figure 7.8 Convergence history comparison for the consistent and non-consistent cases.

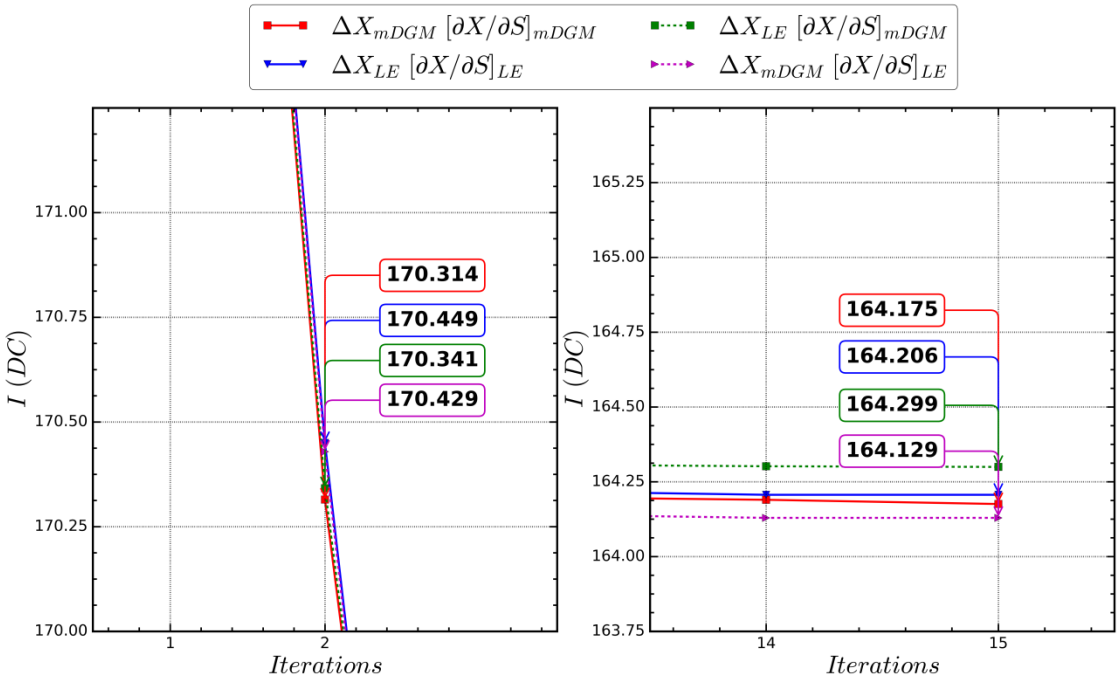


Figure 7.9 Convergence history close up view on the second and last iterations.

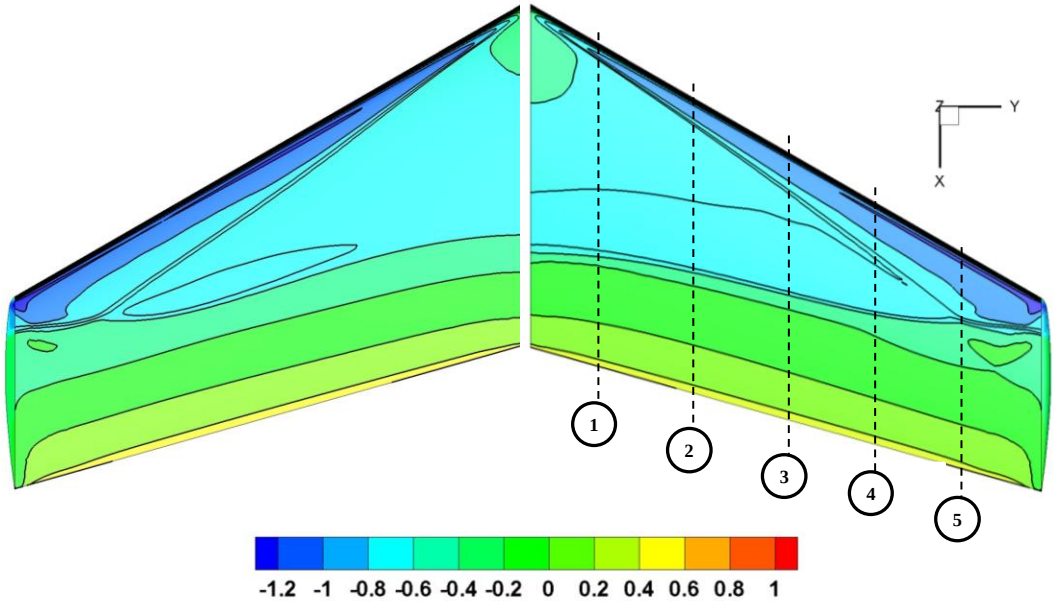


Figure 7.10 Pressure coefficient for the optimised (left) and original geometry (right).

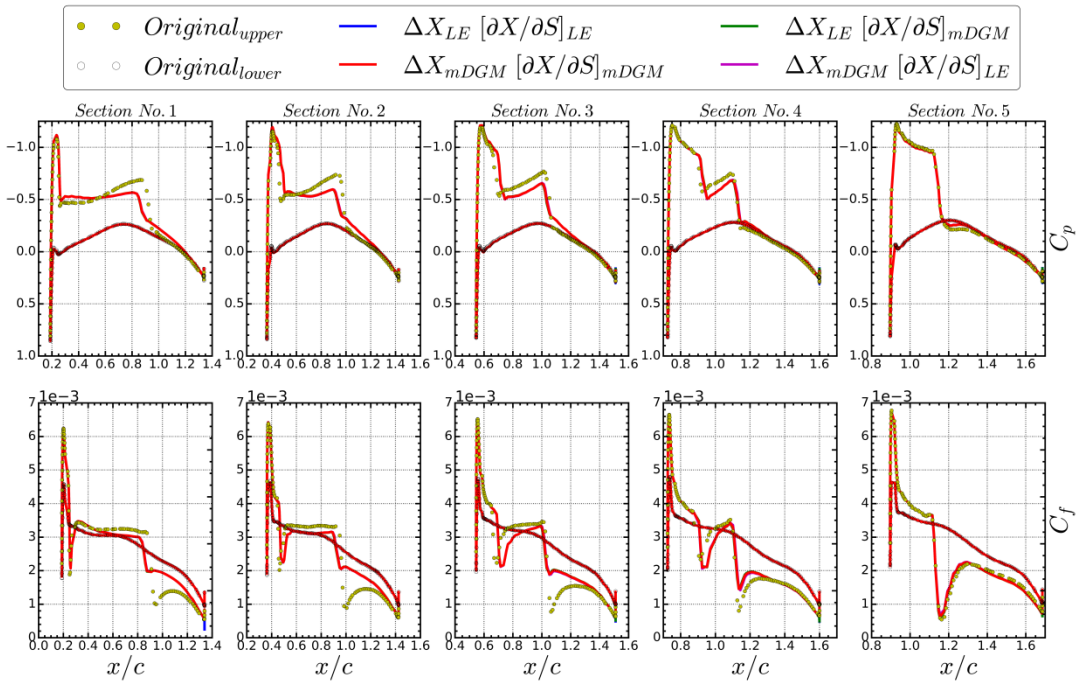


Figure 7.11 Pressure and skin friction coefficient sampled at different stations.

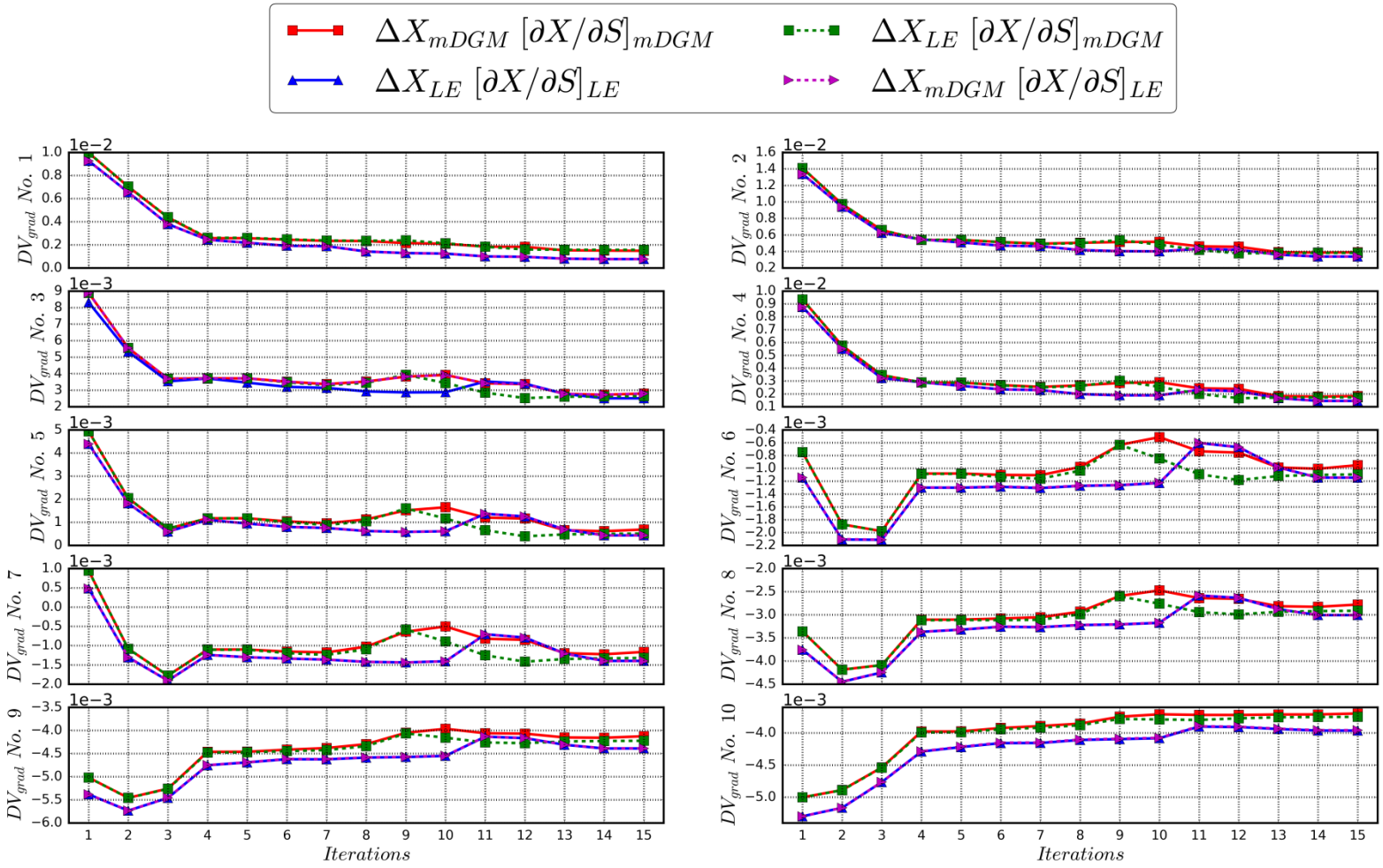


Figure 7.12 DV gradients (corrected for the change in the AoA) history comparison for the consistent and non-consistent cases.

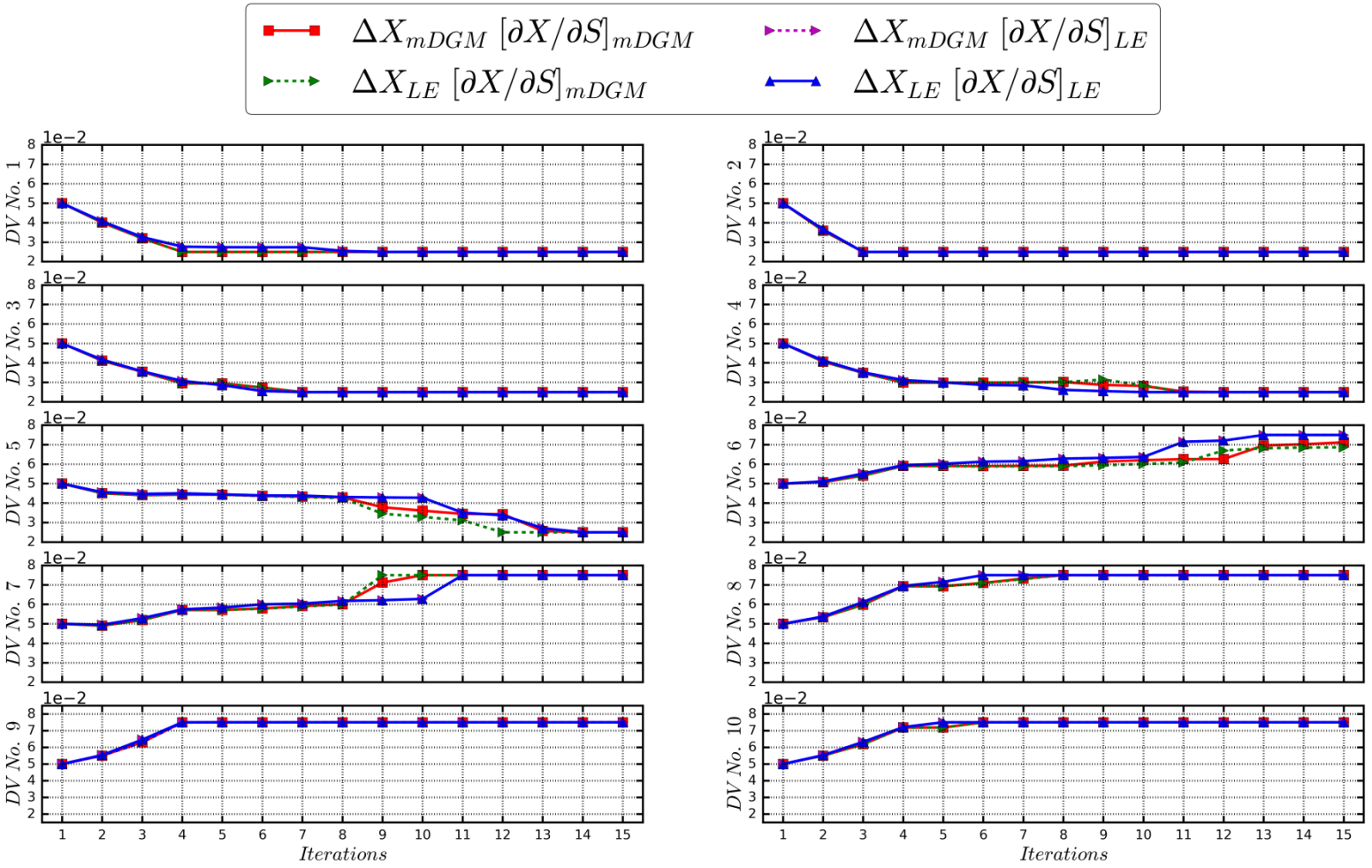


Figure 7.13 DV updates history comparison for the consistent and non-consistent cases.

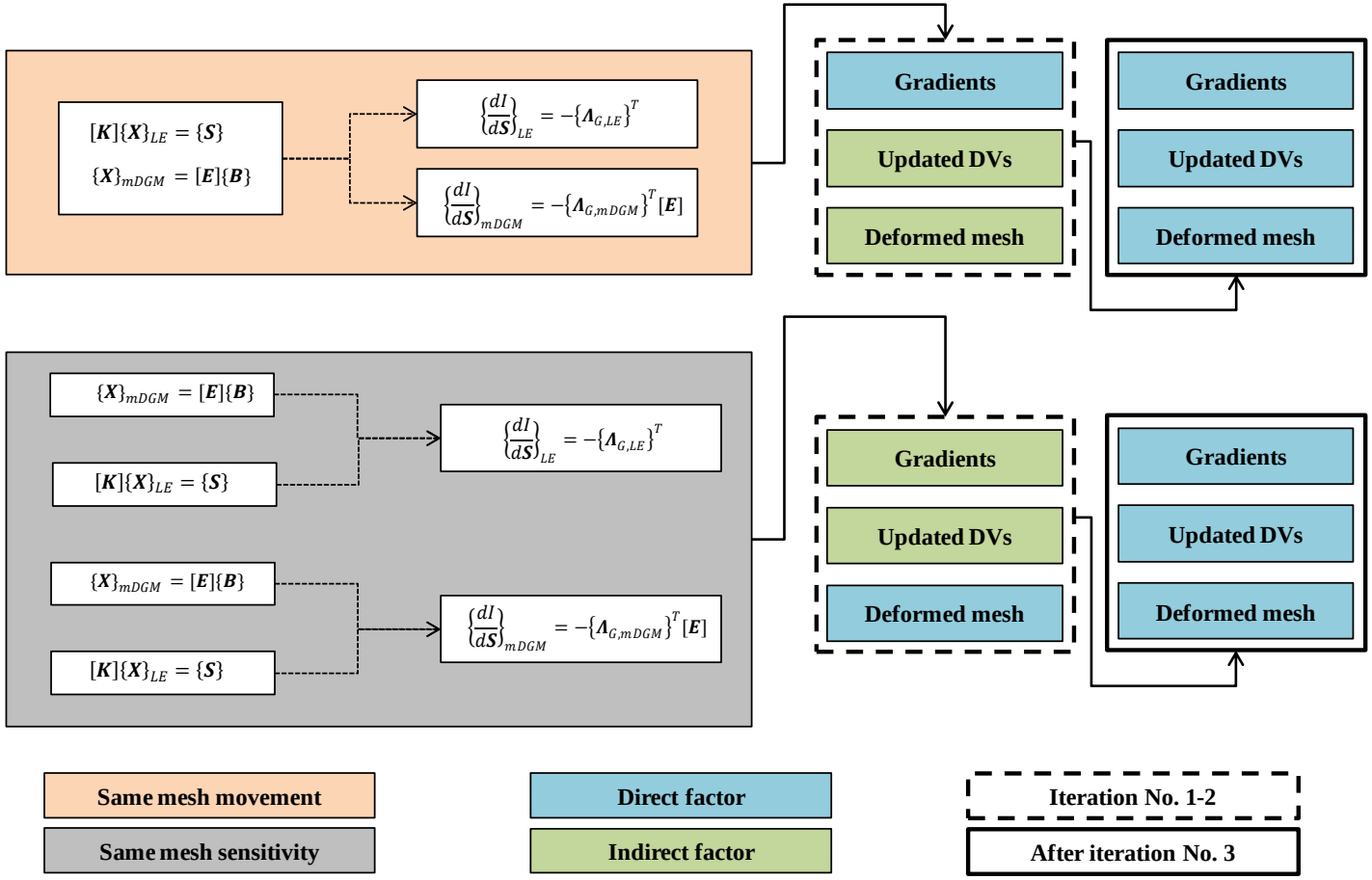
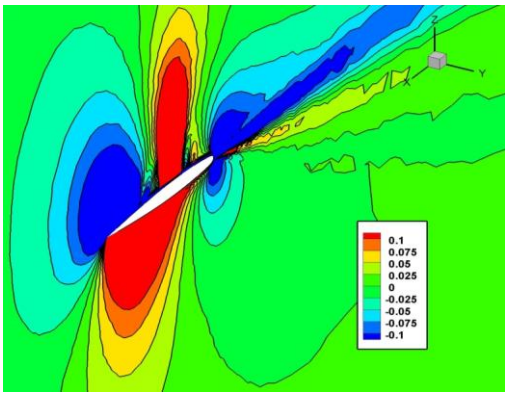
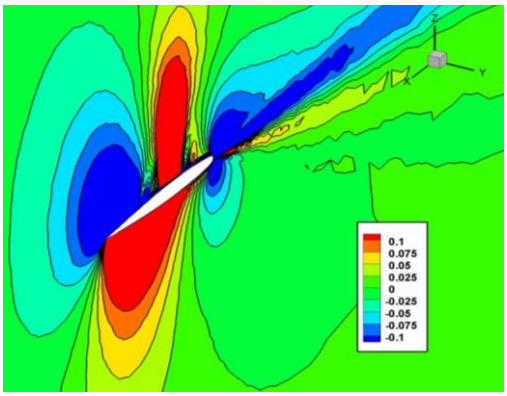


Figure 7.14 Scheme of the post-processing analysis for the consistent and non-consistent cases.

$$\Lambda_1\left(\Delta \boldsymbol{X}_{LE}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{mDGM}\right)$$



$$\Lambda_1\left(\Delta \boldsymbol{X}_{mDGM}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{mDGM}\right)$$



$$\Lambda_1\left(\Delta \boldsymbol{X}_{LE}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{mDGM}\right) - \Lambda_1\left(\Delta \boldsymbol{X}_{mDGM}\left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{S}}\right]_{mDGM}\right)$$

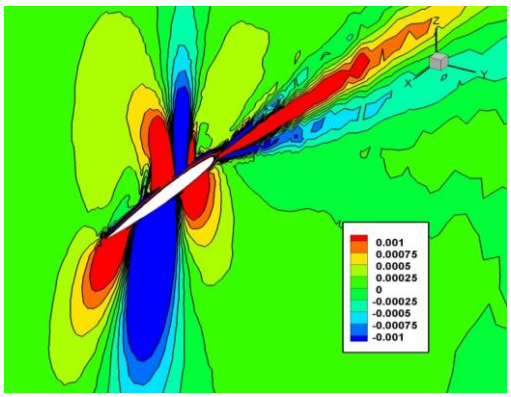


Figure 7.15 Representation of the first adjoint variable associated to the density for two approaches.

7.4.7 Total CPU Time Comparison

In all the optimisations discussed so far, different combination of consistent and non consistent approaches were tested, everything else being the same. The CPU time performance is therefore assessed in relation to only these changes.

As reported in Tab. 7.4, the CPU time cost of the first iteration is analysed separately from the others because this is where the mesh pre-processing is performed. As explained in Chap. 5.8.2, pre-processing refers to the computation of matrix $[\boldsymbol{E}]$ and $[\boldsymbol{K}]$ which is performed once only and not repeated afterwards (see Fig. 7.16).

At the first iteration, the LE method offers the fastest pre-processing CPU time as compared to mDGM. However, this advantage is quickly offset by the net saving in CPU time offered by the mDGM while deforming the mesh and computing the gradient. In fact, at the end of the first iteration, the $\Delta\boldsymbol{X}_{LE}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ approach requires a CPU time which is 3.63 times greater than the $\Delta\boldsymbol{X}_{mDGM}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach. Something similar is registered also for the non-consistent cases where, if the $\Delta\boldsymbol{X}_{LE}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ is used, this is translated in an increased of CPU time equal to 186.67%, whereas if the $\Delta\boldsymbol{X}_{mDGM}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ is used there is an increase of 148.01% in CPU time over the $\Delta\boldsymbol{X}_{mDGM}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach.

From Tab. 7.4, it is evident that, after the first iteration, the advantage of the mDGM over the LE becomes more and more evident corroborating the conclusions made in Chaps. 4.7.1 and 5.8.2. In fact, for each iteration after the first, there is an increase in CPU time for the $\Delta\boldsymbol{X}_{LE}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ approach of almost 8.65 times compared to the consistent $\Delta\boldsymbol{X}_{mDGM}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach. Something similar is registered also for the non consistent cases where if the $\Delta\boldsymbol{X}_{LE}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ is used, this is translated in an increase in CPU time equal to 430.63%, whereas if the $\Delta\boldsymbol{X}_{mDGM}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{LE}$ is used the increase registered is of 335.29% over the $\Delta\boldsymbol{X}_{mDGM}[\partial\boldsymbol{X}/\partial\boldsymbol{S}]_{mDGM}$ approach. This is entirely due to the advantage the mDGM offers while deforming the mesh or computing the grid sensitivity.

In order to estimate the cumulative CPU time difference due to both the mesh movement and sensitivity, the CPU times required to build matrices $[\boldsymbol{E}]$ and $[\boldsymbol{K}]$ are not considered after the first iteration (see Fig. 7.16) and the estimate is done using the remaining CPU times registered at the first iteration only, which also explains the linear trend shown in Fig. 7.17.

Table 7.4 Total CPU time comparisons for different approaches.

Iter.	Strategy	CPU time (s)			
		Pre-processing	Mesh movement	Gradient computation	Total
First iteration	$\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	598_{mDGM}	50_{mDGM}	358_{mDGM}	1,006
	$\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$	$598_{mDGM} + 121_{LE}$		$1{,}726_{LE}$	2,495
	$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$	121_{LE}	$1{,}807_{LE}$	$1{,}726_{LE}$	3,654
	$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	$598_{mDGM} + 121_{LE}$		358_{mDGM}	2,884
For each iteration after the first	$\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$	Not computed	50_{mDGM}	358_{mDGM}	408
	$\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$			$1{,}726_{LE}$	1,776
	$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$		$1{,}807_{LE}$	$1{,}726_{LE}$	3,533
	$\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$			358_{mDGM}	2,165

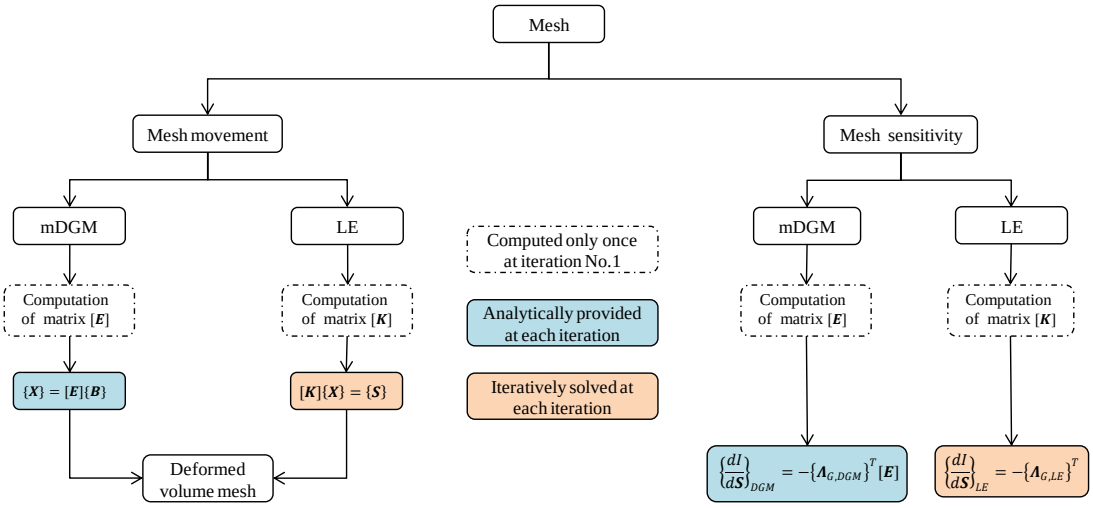


Figure 7.16 Step-by-step procedure for mesh movements and sensitivities.

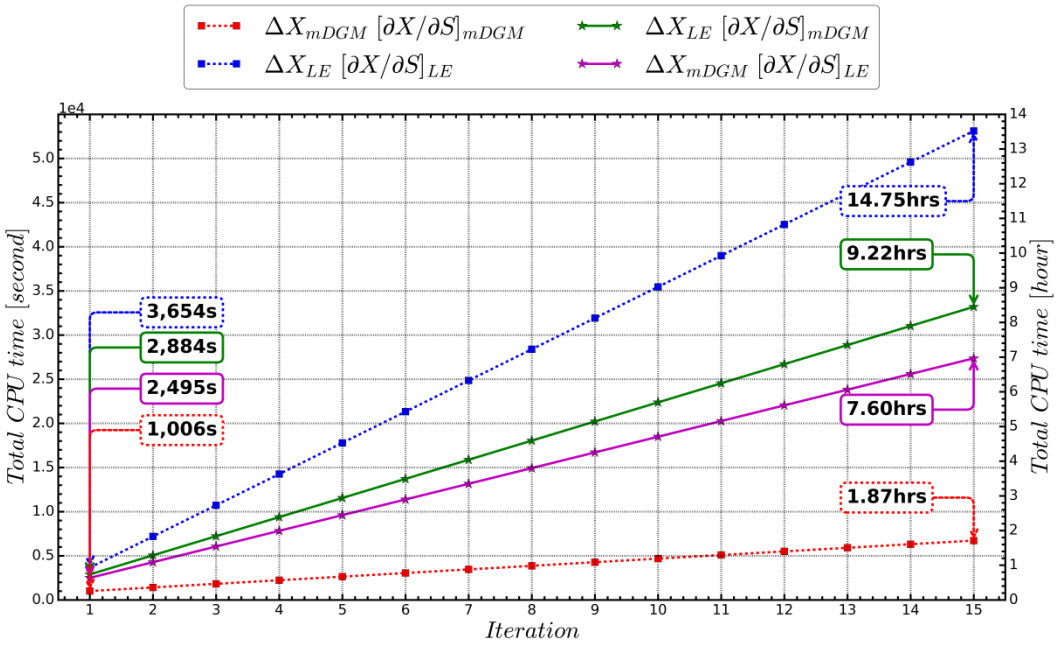


Figure 7.17 Cumulative CPU times comparison for different approaches.

It is evident that, if only the consistent approaches are compared, the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ approach offers a net advantage. For a simulation of 15 iterations (see Fig. 7.17), the initial disadvantage registered in the pre-processing of the mesh is quickly overcome with a final additional CPU time of almost 13hrs over the $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ approach. For what regards the non-consistent cases, it is obvious that there is an advantage over the $\Delta \boldsymbol{X}_{LE}[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{LE}$ consistent approach and a disadvantage over the $\Delta \boldsymbol{X}_{mDGM}\,[\partial \boldsymbol{X}/\partial \boldsymbol{S}]_{mDGM}$ consistent approach.

In order to further understand the impact of the saving offered by the mDGM method, the cost of each step is compared against the overall CPU time of a full optimisation iteration. The results are reported in Figs. 7.18 and 7.19 for the LE and mDGM fully consistent cases respectively. To simplify the analysis, only the second iteration (representative of the others iterations except the first) is considered. Of course, it is recognised that negligible changes within the iterations are present. Since the comparison needs to consider the relative time between the steps, the total cost (computed as the sum of one flow solution, two flow-adjoint solutions, one mesh movement and two grid sensitivities) is considered as a reference value. For the case under analysis, a single flow-adjoint takes roughly 80% of the flow solution, whereas both LE-based mesh movement and sensitivity steps take separately roughly 30% of the same reference time. These values are in line with what was also reported by Nielsen and Park [37].

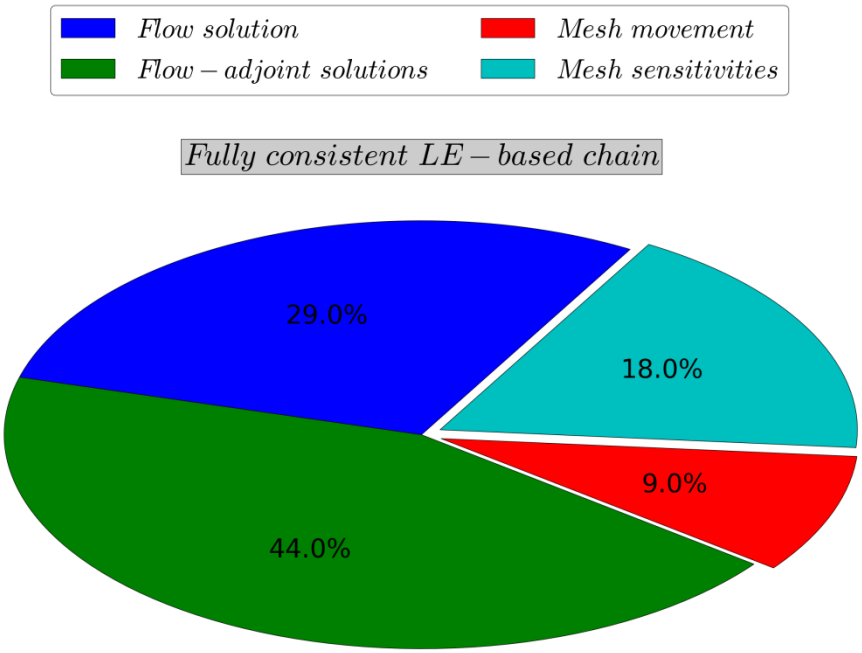


Figure 7.18 Cumulative CPU time breakdown for the fully consistent LE-based chain.

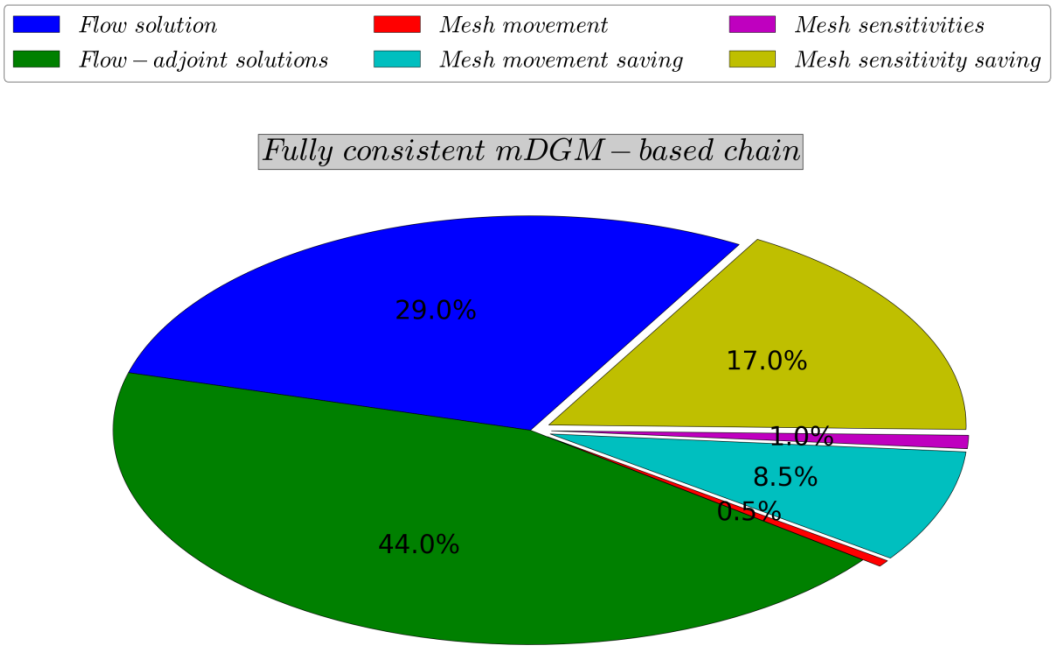


Figure 7.19 Cumulative CPU time breakdown for the fully consistent DGM-based chain.

In conclusion, it can be seen that both mesh movement and grid sensitivity steps represent together roughly the 27% of the total CPU time. The main goal of this work was to reduce this specific CPU expense by using the mDGM approach. In fact, Fig. 7.19 shows that the saving offered by the mDGM-based fully consistent chain is significant both in the mesh movement and sensitivity steps. The previous 18% of the total CPU time spent in computing the two grid sensitivities is greatly reduced down to just 1%, a saving of 17% w.r.t. the overall cost. On the other hand, the CPU time used for the mesh movement is reduced to a insignificant percentage, i.e. 0.5%, of the total CPU time.

7.5 2D Aerofoil Optimisation

7.5.1 *local*-CST versus Free-nodes Approach

The purpose of this chapter is to investigate the potential benefits and feasibility of the *l*-CST within an optimisation exercise. To do so, it is shown how, using an adjoint-based optimisation framework, it is possible to map all the design space, parameterise the entire upper profile using the *l*-CST, but modify only a specific area.

Based on the optimisation problem described in Chap. 7.2, the RAE 5243 aerofoil (see also Chap. 5.4.1) is chosen to minimise the coefficient of drag at constant lift coefficient of 0.82 and Mach number of 0.68. To demonstrate the *l*-CST (local) control capability, the entire upper surface is parameterised, but only a specific area of the aerofoil is given local control in order to obtain a shock bump. Since the goal is to enforce a strong local control with C^2 continuity, the 11^{th} degree *l*-CST is compared against the free-nodes approach used in conjunction with the Sobolev smoothing applied to either the gradient or the shape.

Furthermore, in order to corroborate the advantage of having a strong local support, the three approaches discussed above are also compared against the 11^{th} degree CST approach.

7.5.2 Setup of the Optimisation Study

A shock control bump works creating a ramp facing the incoming flow which creates a series of isentropic compression waves. Its effectiveness was proven by Fulker *et al.* [203] and the bump's height is one of the most important design parameters as noted by Wong *et al.* [204]. A control bump requires roughly one fifth of the chord, and the CST would need a higher than *l*-CST degree curve to impose such a local deformation. As explained at length in Chap. 6.5.3, although the curve degree stays the same, two additional parameters are used to model the bump, i.e. u_{CP_6} and l_6. However, their value is established a-priori and only coefficient A_6 is allowed to change. Needless to say, parameters u_{CP_6} and l_6 could in theory become free parameters if deemed necessary.

Of course, it is recognised that, it is possible to only locally parameterise the profile or to use the H-H [2] bump function, but this would mean the loss of the CST advantages. The bump can be obtained using the *l*-CST by choosing the appropriate CP and its local exponent affecting the bump location and crest respectively as shown in Fig. 6.5. For the aerofoil under analysis this happens to be the 6th CP ($N_{CP} = 12$) at $u_{CP_6} = 0.55$ with $l_6 = 400$. On the other hand, the free-nodes approach can possibility grow a bump within the same selected surface mesh nodes. This is a good example to show how the map provided by gradients $\{dC_{d,l}/d\boldsymbol{S}_{upper}\}$, as depicted in Fig. 5.4, can help to identify specific sensitive areas without worrying about the accuracy and cost of getting the gradient.

The gradient based on the DGM and discussed at length in Chap. 5.3 is used in conjunction with the second-ordered quasi-Newton L-BFGS [193] optimiser. The coupling between the Sobolev and a quasi-Newton optimiser was also used by Özkaya and Gauger [205]. Based on Eq. (5.11), the gradient for the *l*-CST was computed as (for both objective functions $I = C_{d,l}$):

$$\left\{\frac{dI}{d\boldsymbol{D}_{l-CST}}\right\} = -\left\{\frac{dI}{d\boldsymbol{X}}\right\}[\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}_{parameterised}}{\partial \boldsymbol{D}}\right] \tag{7.17}$$

where $\{\boldsymbol{D}_{l-CST}\}$ is the DVs vector containing the 36 coefficients (12 A_i coefficient, 12 l_i coefficients and 12 u_{CP_i} coefficients), but the only free DV allowed to change is only coefficient A_6. The values of u_{CP_6} and l_6 are fixed a-priori. On the other hand, the gradient for the CST was computed as:

$$\left\{\frac{dI}{d\boldsymbol{D}_{CST}}\right\} = -\left\{\frac{dI}{d\boldsymbol{X}}\right\}[\boldsymbol{E}]\left[\frac{\partial \boldsymbol{S}_{parameterised}}{\partial \boldsymbol{D}}\right] \tag{7.18}$$

where $\{\boldsymbol{D}_{CST}\}$ is the DVs vector containing the 12 A_i coefficients, but only coefficient A_6 is allowed to change.

At each iteration an additional flow-adjoint solution w.r.t. the lift coefficient, C_l, was computed in order to accurately compute $(dC_{d,l}/d\alpha)$, as explained in Chap. 7.2. The Jacobian $[\partial \boldsymbol{S}_{parameterised} \ / \partial \boldsymbol{D}]$ was computed accordingly to Eq. (6.17).

Whenever a parameterisation technique is used, a smooth-like bump (small or big) is expected, whereas the free-nodes approach is likely to generate a non-smooth deformation, unless some precautions are taken. The smooth Sobolev gradient, i.e. subscript S, for the free-nodes approach was computed as:

$$[\boldsymbol{S}]\left\{\frac{dI}{d\boldsymbol{D}_{free\,-nodes}}\right\}_S = -\left\{\frac{dI}{d\boldsymbol{X}}\right\}[\boldsymbol{E}] \tag{7.19}$$

where $[\boldsymbol{S}]$ is the smoothing matrix defined as in Eq. (6.26) and a value of 10 for the smoothing factor ε. Furthermore, $\{\boldsymbol{D}_{free\,-nodes}\}$ is the DVs vector containing all the 180 upper surface mesh nodes, although only roughly 30 of them are allowed to move accordingly to the area highlighted by the red and yellow dots in Fig. 7.21. The gradient expressed in Eq. (7.19) is then corrected for a change in the AoA as shown in Eq. (7.5). Similarly, this second order elliptic smoothing can also be applied to the shape as suggested by Mohammadi [10] and its formulation is expressed in Eq. (6.28).

In order to understand the fundamental difference of these two approaches, it is beneficial to look at the two optimisation loops as depicted in Fig. 7.20. When the Sobolev operator is first applied to the gradient, this is then subsequently fed to the optimiser. This last one receives a modified gradient (see Eq. (7.21)) and provides the DV updates based on this. On the other hand, these two steps are also performed when the shape is smoothed directly, but are applied in a different temporal order. In fact, as depicted in Fig. 7.20 (right), the raw gradient is first computed and then fed directly to the optimiser which in turn provides the DV updates. At this point the shape is updated and subsequently smoothed using the Sobolev operator.

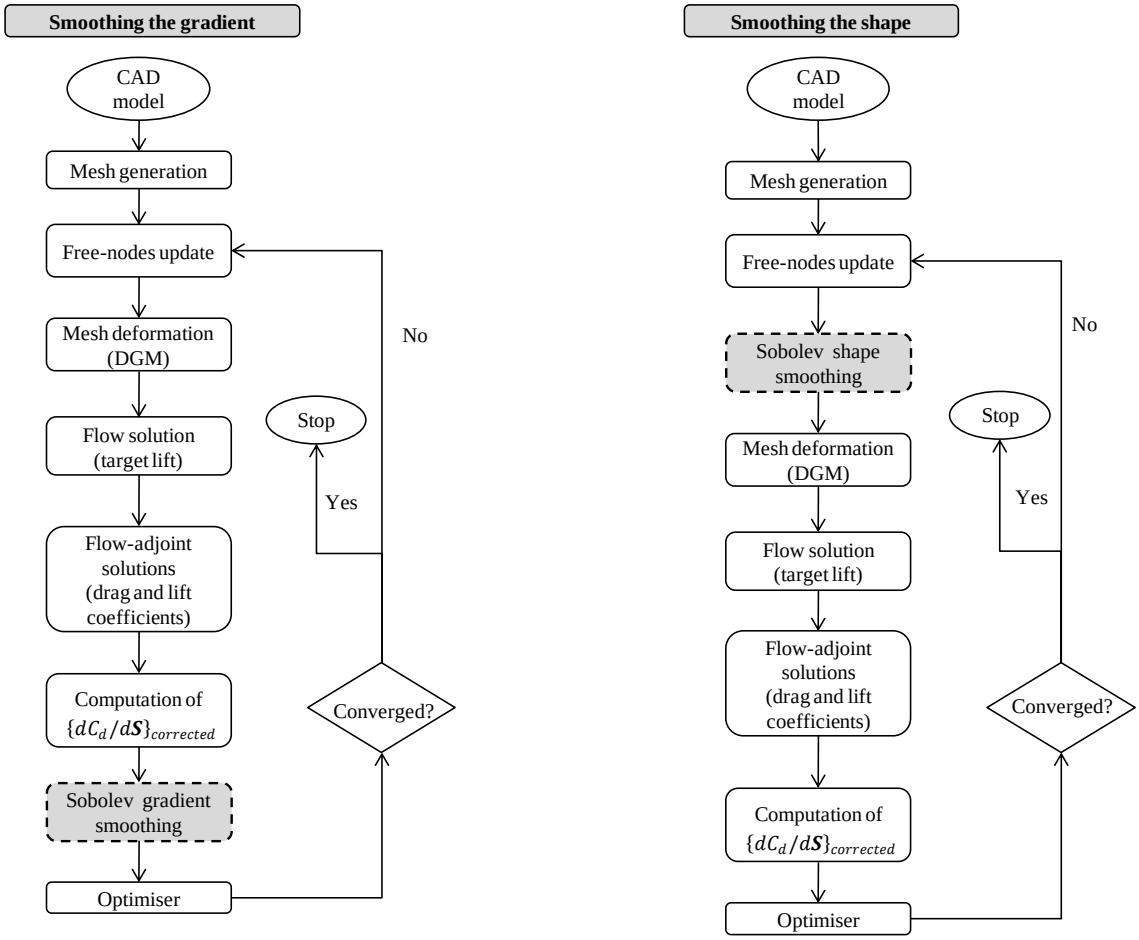


Figure 7.20 Workflow comparison between gradient and shape smoothing.

7.5.3 Results

The *l*-CST and the free-nodes approaches compared in this section modify the same specific part of the geometry shown in Fig. 7.21 within the red and yellow dots. The initial and final shapes are depicted in Fig. 7.21 (top left), whereas the resulting pressure distributions are depicted in Fig. 7.21 (top right). The bumps obtained using the free-nodes technique seem to be quite similar, but significantly different from the one obtained using the *l*-CST. These differences are also reflected in the resulting pressure coefficient distributions where the slight changes in the coefficient of pressure outside the bump boundaries can be explained by the fact that the AoA is different from the initial value (a consequence of the lift-constrained solution).

In order to further investigate these differences, the slope and curvature of the upper profile are plotted in Fig. 7.21 (bottom left and right respectively). These two metrics show some fundamental differences between the optimised shapes obtained by the three methods. In fact, the *l*-CST provides a C^2 continuous curve which is reflected in a much smoother bump than the cases where the free-nodes approach were used. The Sobolev smoothing has certainly improved both the slope and curvature profiles, but with some limitations. In fact, smoothing directly the shape seems to suffer more from high oscillation components than the strategy where the gradient was directly smoothed instead. The possibility to get a poor smoothness level while using the Sobolev operator was highlighted also by both Kim *et al.* [185] and Wu *et al.* [206]. Furthermore, another fundamental difference can be seen in the two frameworks compared in Fig. 7.20. In particular, whenever the gradient is smoothed directly, the Sobolev operator is applied just before the optimiser is called. On the other hand, when the shape is smoothed directly, the Sobolev operator is applied directly to the shape after the DVs were updated using the raw gradient.

The three strategies managed to achieve a large drag reduction as shown in Fig. 7.22. Both free-nodes approaches show a fairly similar convergence rate, however the case where the shape was smoothed directly shows some severe oscillations just before convergence was reached. On the other hand, the case where the gradient was smoothed and therefore the shape was smoothed indirectly, obtained an additional reduction of 4.51 DC. As noted by Wu *et al.* [186], whenever the Sobolev operator is used, the direction of improvement is effectively modified and there is no guarantee that the smoothed gradient or shape would lead to the same variation of the target function, although there is a gain in regularity in the shape update. On the other hand, the *l*-CST managed a reduction of 23.64 DC with far less iterations. This is an improvement of just 0.848 DC compared to the best performing free-nodes approach, but with the advantage of a smooth C^2 optimised profile which is of paramount importance for transonic aerodynamics.

Fundamental differences are also registered in terms of convergence rate. The free-nodes approaches show a rather slow convergence rate, whereas the *l*-CST convergence history shows a faster but not regular trend, meaning non monotonically decreasing. This can be explained with the fact that the targeted highly sensitive area induced the optimiser to overshoot at iterations No. 3, and 5, rapidly change direction and eventually settle down to a more regular path.

Overall the results can be explained using the concept of exploitation, i.e. convergence speed, and exploration, i.e. search of the design space. The *l*-CST surfs the design space using a N_{DV} which is far smaller than the DVs ($= N_{SM}$) used by the free-nodes approach. This is translated in a faster exploitation, but with an improved (thanks to the LF) exploration capability. On the contrary, the free-nodes approach is characterised by excellent space exploration ($N_{SM} = N_{DV}$), but with very poor exploitation, i.e. slow convergence.

In conclusion, the fact that the free-nodes approach did not outperform the *l*-CST although possesses higher explorablity, can be justified by the fact that these extra degrees of freedom increase the possibility of getting stuck in a local minimum.

To corroborate these results, the 11th degree CST was also tested and compared with the other three strategies to see what is the effect of using only a global support parameterisation to model a shock control bump. A fundamental difference is that the CST 6th CP affects an area greater than that indentified by the yellow and the red dots depicted in Fig. 7.21. This is because the CST based on the BPs is unable to provide local control.

One of the first noticeable similarity with the *l*-CST is the CST irregular convergence history. This is very different from the almost monotone decreasing convergence history featured by the two free-nodes approaches tested. In fact, the CST shares the multiple overshooting of the objective function and the fewer than the free-nodes approaches iterations to convergence. The fact that the CST did not manage to perform as well as the other three methods tested, i.e. *l*-CST and free-nodes, can be explained with the fact that, for this test case, the highly sensitive shock area requires either a more global or a localised change in the geometry.

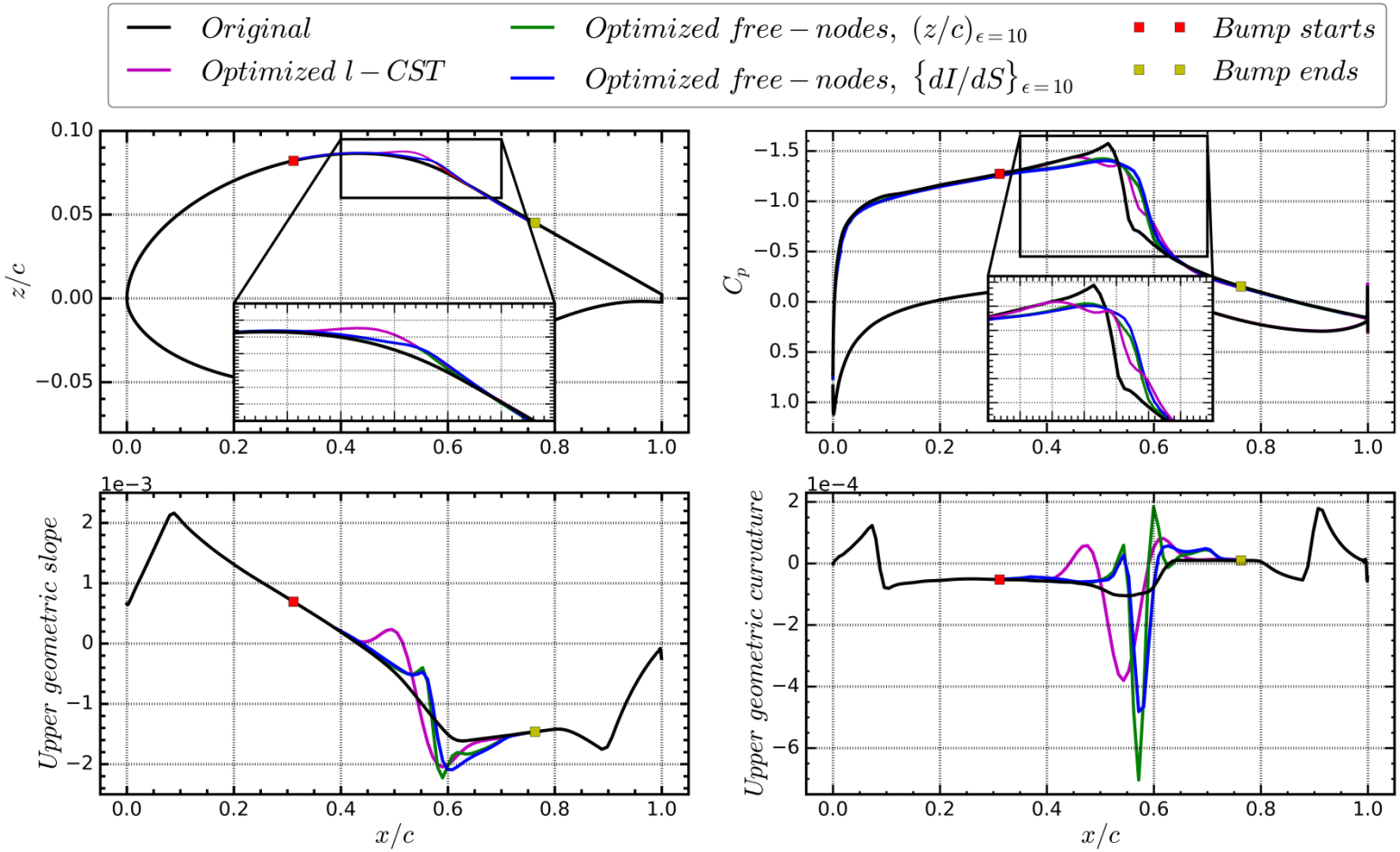


Figure 7.21 Initial and optimised shape, pressure distribution, slope and curvature.

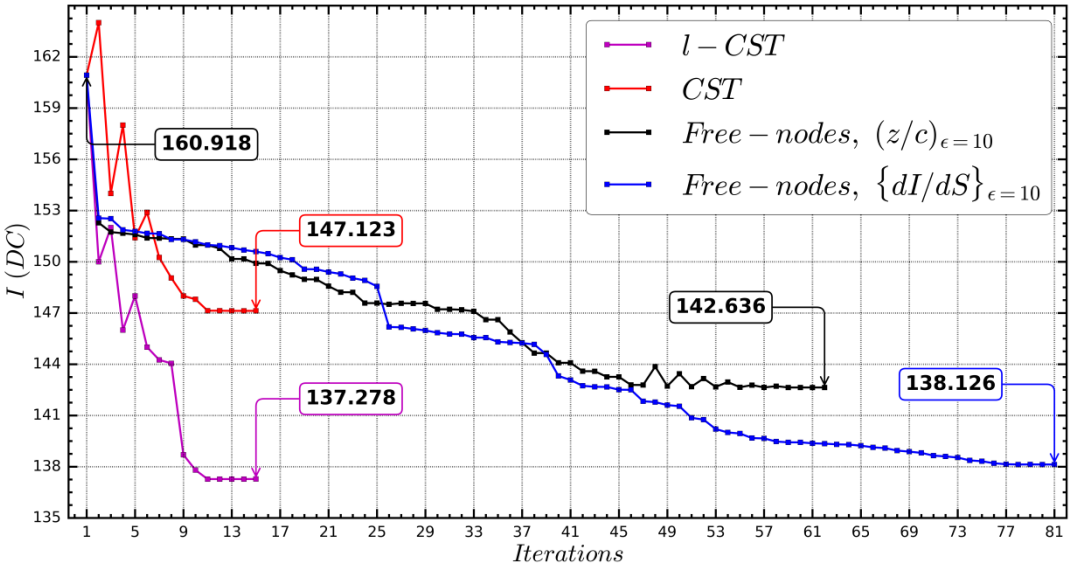


Figure 7.22 Convergence histories comparison.

7.6 Summary

The consistent versus non-consistent approaches and the *l*-CST parameterisation method both based on the DGM adjoint chain were tested in an optimisation loop. The results were analysed and a discussion provided to explain the differences.

It was shown how there is no need to differentiate the entire computational domain while evaluating the grid sensitivity. This has then laid down the foundations to study the non-consistent mesh movement and differentiation approaches. It was shown that small differences in the gradients are initiated by employing different grid sensitivities. On the other hand, small differences in the deformed grids are initiated by different mesh movements. Furthermore, the small differences in the gradients were not large enough to trigger the optimiser to predict different DV updates and the differences in the deformed grids were not big enough to induce the flow solver to predict a different value of the objective function. It was therefore concluded that the non-consistent approaches can be indeed used.

A highly flexible aerofoil parameterisation, termed *l*-CST, was tested in an optimisation loop against the free-nodes approach. The method was shown to provide enough local control to model a localised bump while maintaining a C^2 continuous curve. The *l*-CST was able to

provide a better minimum as compared to both the free-nodes approaches tested because it provides a better mix between exploitation and exploration feature. Furthermore, it was shown that, within the test cases considered, smoothing the gradient seems to be better than smoothing the shape.

Chapter 8 Conclusions and Future Studies

The main focus of this thesis was to investigate a new method, i.e. the differentiated version of the DGM, capable of reducing the memory and CPU time requirements of the grid sensitivity computation in the discrete adjoint framework. At the same time, a new formulation of the CST, termed *l*-CST, was proposed to bridge the gap between the free-nodes approach and the CST parameterisation technique. The link between these two techniques lays on the need to efficiently map a large design space while being able to target specific areas without increasing the order of the parameterised curve.

The novel approaches introduced in this work were also studied in a series of optimisation exercises. First, the differentiation of the DGM was investigated within the consistent and non-consistent discrete adjoint frameworks. Secondly, the *l*-CST was also investigated in an optimisation loop, but against the free-nodes approach.

In this chapter what was achieved is summarised and what still needs to be done is outlined.

8.1 Conclusions

There are a series of conclusions that were reached in this thesis. These have implications distributed in four areas, namely mesh movement, mesh sensitivity, parameterisation and optimisation framework. These are summarised as follows:

- It was highlighted that using a fast algebraic mesh movement method, namely the DGM, offers large advantages in terms of low memory and low CPU time consumption. This can be achieved thanks to the explicit availability of the surface-to-volume map. However, this needs to be balanced with the robustness issue related to the way the Delaunay map is constructed. To this regard a solution was discussed where it was shown that the introduction of the supporting box allows to obtain an improved Delaunay elements distribution.

- An analysis of the advantages and possible issues that may arise while using the DGM-based gradient chain in the discrete adjoint framework was proposed. The main conclusion is that, while using an explicit mesh movement, the mesh adjoint system

does not need to be solved because the volumetric grid sensitivity, i.e. Jacobian $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$, is available explicitly, thus bringing a significant saving in the CPU time and memory as compared to any implicit mesh movement such as LE. The comparison against the LE-based gradient highlighted the close similarity in trend between the sensitivity maps. The reason why the general gradient trend of the map $\{dI/d\boldsymbol{S}\}$ is not so different even thought two different mesh movements are used needs to be investigated in the close to the wall distribution of the physical term, i.e. $\{dI/d\boldsymbol{X}\}$. Furthermore, it was established that the volumetric mesh sensitivity, i.e. $[\partial \boldsymbol{X}/\partial \boldsymbol{S}]$, must have a distribution in the computational domain such that it does not cut off the physical information carried by $\{dI/d\boldsymbol{X}\}$.

- A method based on the corrected BPs, termed *l*-CST, was used to confer the CST parameterisation method a strong on-demand local control. The method is based on a simple local trigonometric function which works as a cut-off filter on the BPs. The strong local control was obtained while maintaining the same curve degree. Effectively, this was achieved independently from the N_{CP}. The method was tested on an inverse geometric fitting exercise for a series of 2D aerofoils where it was demonstrated how a lower BP order could be used to guarantee the same and sometimes better fitting accuracy.

- The DGM-based gradient chain was studied in an optimisation loop by considering two frameworks. The first employed a 3D geometry, the ONERA M6 wing, whereas the second employed a 2D aerofoil, the RAE 5243. The conclusions are summarised as follows:

    - In the first framework, the DGM-based gradient was studied in a series of 3D lift-constrained drag reduction optimisations by addressing the question of consistent versus non-consistent mesh movement and mesh sensitivity approaches. The study raised the question as to what happens when different non-consistent mesh approaches are used in the discrete adjoint optimisation framework. The different minima found in this work can be considered essentially the same. In particular, the initial small changes in the gradients (caused by non-consistent grid sensitivities) were not large enough to trigger

different DV updates (computed by the optimiser), whereas small changes in the first (deformed) grid updates (caused by non-consistent grid movements) were not enough to trigger a large difference in the objective function (computed by the flow solver). Based on the numerical results presented in this work and within the framework tested, non-consistent approaches can be used without incurring in a large deviation w.r.t. the consistent approach.

- In the second framework, the *l*-CST was tested on a 2D lift-constrained drag reduction optimisation against the free-nodes method with the Sobolev smoothing operator applied to either the shape or the gradient. The *l*-CST was able to deliver a better minima w.r.t. the two free-nodes approaches in much less iterations to convergence. This is because the *l*-CST, like any other parameterisation method, has excellent exploitation feature with an improved exploration capability delivered by the LF.

8.2 Suggestions for Future Work

This chapter outlines some suggested future studies which are the result of a continuous refinement of the original research questions and are indeed provided in form of *improved questions*. The recommendations are given for mesh deformation, grid sensitivity, parameterisation and optimisation framework. These are summarised as follows:

- This thesis describes how explicit and implicit mesh movements work. The former offers low memory and computation time requirements with relative poor robustness features, whereas for the latter the situation is reversed. From the experience gained in this work, two recommendations can be made. Firstly, while choosing a mesh movement, this needs to be done considering also the requirements imposed by its differentiated formulation. Secondly, to have a robust mesh deformation which features both low memory and computation time requirements, one needs to find an explicit formulation based on the area/volume of each single grid element. This is important because negative area/volume is the requirement that ultimately defines the robustness of any mesh movement method.

- In order to further corroborate the memory and time saving while moving the mesh and computing the gradient, a comparison against a parallel implementation can be done. In this case, although not addressed in this thesis, the DGM method can be easily parallelised as each step is independent from the other operations. On the other hand, the parallelisation of the LE method is less straightforward due to the presence of the preconditioner.

- It was shown how, by using the supporting box, the quality of the $\{dI/d\boldsymbol{S}\}$ map was improved. However, this was done by maintaining the supporting box nodes fixed. Another strategy would be to let these nodes move in order to allow large deformations. This option requires to establish the link between the wall and the supporting box points which in turn makes also necessary to revisit the linearisation of the DGM method.

- It was shown how a weak link can be made between gradient $\{dI/d\boldsymbol{S}\}$ and both C_p and C_f plots. However, this simplified visual comparison does not apply to sensitive areas where neither pressure nor skin friction plot show a rapid change. It was established by Hinchliffe and Qin [46] that these areas are not a numerical artefact. However, a further investigation is needed to better understand the underlying physics.

- The importance of having an on-demand local control for a 2D test case was demonstrated. Its natural extension would be a 3D application.

- Following the optimisation studies, the following points require further investigation:

    - In a lift-constrained optimisation, the gradient w.r.t. the AoA could be computed using either finite differences or complex step by rotating the shape so as to properly account for the effect of different grid movements.

    - A non-constrained lift optimisation would bring further insight to understand the effect of the target lift constraint.

    - The gradient computed as $\{dI/d\boldsymbol{S}\}$ and not as $\{dI/d\boldsymbol{D}\}$, and therefore with no parameterisation, would exclude any possible influence or gradient regularisation/smoothing imposed by the parameterisation technique.

- During the analysis of vector $\{dI/d\boldsymbol{X}\}$, the need to analyse different flow regimes has emerged, namely subsonic and transonic, because this changes the pressure and viscous contribution of $\{dI/d\boldsymbol{X}\}$ on $\{dI/d\boldsymbol{D}\}$. This is expected to play an important role at subsonic regime when different mesh movements are used. In fact, at subsonic speed the near wall mesh orthogonality plays a major role when compared to the transonic case.

- The *l*-CST highlighted the differences resulting using the Sobolev operator applied directly to the gradient or the shape separately. A further investigation using the 3D extension of the Sobolev gradient for unstructured grids (i.e. the Laplace-Beltrami operator [176] would bring further insight into the difference between these two different approaches.

References

[1] Jameson, A., "Computational Aerodynamics for Aircraft Design," *Science*, Vol. 245, No. 4916, 1989, pp. 361-371.

[2] Hicks, R. M., and Henne, P., “Wing Design by Numerical Optimization,” *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407-412.

[3] Drela, M., "Pros & Cons of Airfoil Optimization, "*Frontier of Computational Fluid Dynamics*, edited by D. A. Caughey and M. M. Hafez, World Scientific, Singapore, 1998, pp. 363–381.

[4] Sobieszczanski-Sobieski, J., "The Case for Aerodynamic Sensitivity Analysis," *NASA CP-2457*, September 1986.

[5] Pironneau, O., "On Optimum Design in Fluid Mechanics, "*Journal of Fluid Mechanics*, Vol. 64, No. 1, 1974, pp. 97–110.

[6] Jameson, A., "Aerodynamic Design via Control Theory, "*Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233-260.

[7] Reuther, J., Jameson, A., Farmer, J., Martinelli, L., and Saunders, D., "Aerodynamic Shape Optimization of Complex Aircraft Configurations Via an Adjoint Formulation," *34th Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 96-0094, January 1996.

[8] Liu, X., Qin, N., and Xia, H., "Fast Dynamic Grid Deformation Based on Delaunay Graph Mapping," *Journal of Computational Physics*, Vol. 211, No. 2, 2006, pp. 405-423.

[9] Jameson, A., and Kim, S., “Reduction of the Adjoint Gradient Formula in the Continuous Limit,” *41st AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2003-0040, January 2003.

[10] Mohammadi, B., “Optimal Shape Design, Reverse Mode of Automatic Differentiation and Turbulence,” *35th Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 97-0099, January 1997.

[11] Kulfan, B. M., “Universal Parametric Geometry Representation Method,” *Journal of Aircraft*, Vol. 45, No. 1, 2008, pp. 142–158.

[12] Sederberg, T. W., and Parry S. R., “Free-form Deformation of Solid Geometric Models,” *ACM SIGGRAPH Computer Graphics*, Vol. 20, No. 4, 1986, pp. 151-160.

[13] Nelder, J. A., and Mead, R., "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308-313.

[14] Simpson, T. W., Mauery, T. M., Korte, J. J., and Mistree, F., "Kriging Models for Global Approximation in Simulation-based Multidisciplinary Design Optimization," *AIAA Journal*, Vol. 39, No. 12, 2001, pp. 2233-2241.

[15] Vicini, A., and Quagliarella, D., "Airfoil and Wing Design Through Hybrid Optimization Strategies," *AIAA Journal*, Vol. 37, No. 5, 1999, pp. 634-641.

[16] Lehner, S., and Crossley, W. A., "Hybrid Optimization for a Combinatorial Aircraft Design Problem," *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Hilton Head, USA, AIAA 2009-7116, September 2009.

[17] Martins, J. R. R. A., and Hwang, J. T., "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models," *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2582-2599.

[18] Pulliam, T. H., Nemec, M., Holst, T., Zingg, D. W., and Kwak, D., "Comparison of Evolutionary (Genetic) Algorithm and Adjoint Methods for Multi-objective Viscous Airfoil Optimizations," *41st AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2003-0298, January 2003.

[19] Zingg, D. W., Nemec, M., and Pulliam, T. H., “A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization,” *European Journal of Computational Mechanics*, Vol. 17, No. 1–2, 2008, pp. 103–126.

[20] Giles, M. B., and Pierce, N. A., "An Introduction to the Adjoint Approach to Design," *Flow, Turbulence and Combustion*, Vol. 65, No. 3-4, 2000, pp. 393-415.

[21] Narayanan, P., and Chattopadhyay, A., "A Discrete Semianalytical Procedure for Aerodynamic Sensitivity Analysis Including Grid Sensitivity," *Computers & Mathematics with Applications*, Vol. 32, No. 3, 1996, pp. 61-71.

[22] Lions, J. L., "Optimal Control of Systems Governed by Partial Differential Equations," Springer-Verlag, Translated by Mitter, S. K., Berlin, 1971.

[23] Jameson, A., and Reuther, J., "Control Theory Based Airfoil Design Using the Euler Equations," *5th Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, USA, AIAA 94-4272, September 1994.

[24] Jameson, A., Martinelli, L., and Pierce, N. A., "Optimum Aerodynamic Design Using the Navier–Stokes Equations," *Theoretical and Computational Fluid Dynamics*, Vol. 10, No. 1-4, 1998, pp. 213-237.

[25] Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and Der Weide, E. V., "ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers," *AIAA Journal*, Vol. 46, No. 4, 2008, pp. 863-873.

[26] Dwight, R. P., and Brezillon, J., "Effect of Approximations of the Discrete Adjoint on Gradient-based Optimization," *AIAA Journal*, Vol. 44, No. 12, 2006, pp. 3022-3031.

[27] Giles, M. B., Duta, M. C., Muller, J. D., and Pierce, N. A., "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, Vol. 41, No. 2, 2003, pp. 198-205.

[28] Nadarajah, S., and Jameson A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization," *38th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2000-0667, January 2000.

[29] Zymaris, A. S., Papadimitriou, D. I., Giannakoglou, K. C., and Othmer, C., "Continuous Adjoint Approach to the Spalart–Allmaras Turbulence Model for Incompressible Flows," *Computers & Fluids*, Vol. 38, No. 8, 2009, pp. 1528-1538.

[30] Lozano, C., "Discrete Surprises in the Computation of Sensitivities from Boundary Integrals in the Continuous Adjoint Approach to Inviscid Aerodynamic Shape Optimization," *Computers & Fluids*, Vol. 56, 2012, pp. 118-127.

[31] Giles, M. B., and Pierce, N. A., "On the Properties of Solutions of the Adjoint Euler Equations," *6th ICFD Conference on Numerical Methods for Fluid Dynamics*, Oxford, UK, March 1998.

[32] Jacques, P., Nguyen-Dinh, M., and Trontin, P., "Goal Oriented Mesh Adaptation Using Total Derivative of Aerodynamic Functions with Respect to Mesh Coordinates with Applications to Euler Flows," *Computers & Fluids*, Vol. 66, 2012, pp. 194-214.

[33] Ta'asan, S., "Trends in Aerodynamic Design and Optimization: a Mathematical View Point," *12th AIAA Computational Fluid Dynamics Conference*, San Diego, USA, AIAA 95-1731, June 1995.

[34] Venditti, D., and Darmofal, D., "A Multilevel Error Estimation and Grid Adaptive Strategy for Improving the Accuracy of Integral Outputs," *14th AIAA Computational Fluid Dynamics Conference*, Norfolk, USA, AIAA 99-3292, November 1999.

[35] Anderson, W. K., and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers & Fluids*, Vol. 28, No. 4, 1999, pp. 443-480.

[36] Widhalm, M., Brezillon, J., Ilic, C., and Leicht, T., "Investigation on Adjoint Based Gradient Computations for Realistic 3D Aero-Optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Forth Worth, USA, AIAA 2010-9129, September 2010.

[37] Nielsen, E. J., and Park, M. A., "Using an Adjoint Approach to Eliminate Mesh Sensitivities in Computational Design," *AIAA Journal*, Vol. 44, No. 5, 2006, pp. 948-953.

[38] Hicken, J. E., and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA Journal*, Vol. 48, No. 2, 2010, pp. 400-413.

[39] Mavriplis, D. J., "Formulation and Multigrid Solution of the Discrete Adjoint Problem on Unstructured Meshes," in Computational Dynamics 2004: *Proceedings of the Third International Conference on Computational Fluid Dynamics*, ICCFD3, Toronto, 12-16 July, pp. 663-668, Springer, Berlin, Germany, 2004.

[40] Dwight, R., Brezillon, J., and Vollmer, D., "Efficient Algorithms for Solution of the Adjoint Compressible Navier–Stokes Equations with Applications," *In Proceedings of the ONERA-DLR Aerospace Symposium (ODAS)*, Toulouse, 2006.

[41] Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856-869.

[42] Kim, S., Alonso, J. J., and Jameson, A., "A Gradient Accuracy Study for the Adjoint-based Navier-Stokes Design Method," *37th Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 1999-299, January 1999.

[43] Soemarwoto, B. I., "The Variational Method for Aerodynamic Optimization Using the Navier–Stokes Equations," NASA CR-97-206277, December 1997.

[44] Kim, C. S., Kim, C., and Rho, O. H., "Sensitivity Analysis for the Navier-Stokes Equations with Two-equation Turbulence Models," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 838-845.

[45] Palacios, F., Alonso, J. J., Colonno, M., Hicken, J., and Lukaczyk, T., "Adjoint-based Method for Supersonic Aircraft Design Using Equivalent Area Distributions," *50th AIAA Aerospace Sciences Meeting*, Nashville, USA, AIAA 2012-0269, January 2012.

[46] Hinchliffe, B. L., and Qin, N., "Using Surface Mesh Sensitivity From Mesh Adjoint Solution for Transonic Wing Drag Reduction," *54th AIAA Aerospace Sciences Meeting and Exhibit*, San Diego, USA, AIAA 2016-0560, January 2016.

[47] Weinerfelt, P., and Enoksson, O., "Numerical Methods for Aerodynamic Optimization," *8th International Symposium on Computational Fluid Dynamics*, Bremen, Germany, September 1999.

[48] Nielsen, E. J., and Anderson, W. K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 37, No. 11, 1999, pp. 1411-1419.

[49] Mohammadi, B., "Flow Control and Shape Optimization in Aeroelastic Configurations," *37th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 99-0182, January 1999.

[50] Kim, H. J., Obayashi, S., and Nakahashi, K., "Flap-deflection Optimization for Transonic Cruise Performance Improvement of Supersonic Transport Wing," *Journal of Aircraft*, Vol. 38, No. 4, 2001, pp. 709-717.

[51] Wang, G., Xu, L., Li, C., and Ye, Z-Y., "Influence of Mesh Sensitivity on Computational-Fluid-Dynamics-Based Derivative Calculation," *AIAA Journal*, Vol. 54, No. 12, 2016, pp. 3717-3726.

[52] Kim, H. J., Sasaki, D., Obayashi, S., and Nakahashi, K., "Aerodynamic Optimization of Supersonic Transport Wing Using Unstructured Adjoint Method," *AIAA Journal*, Vol. 39, No. 6, 2001, pp. 1011–1020.

[53] Anderson, W. K., and Bonhaus, D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers & Fluids*, Vol. 23, No. 1, 1994, pp. 1-21.

[54] Martins, J. R. R. A., Alonso J. J., and Reuther J. J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," *Optimization and Engineering*, Vol. 6, No. 1, 2005, pp. 33–62.

[55] Nemec, M., and Zingg, D. W., "Newton-Krylov Algorithm for Aerodynamic Design Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 856–859.

[56] Hicks, R. M., Murman, E. M., and Vanderplaats, G. N., "An Assessment of Airfoil Design by Numerical Optimization," NASA TM-X-3092, July 1974.

[57] Hicks, R. M., and Vanderplaats, G. N., "Application of Numerical Optimization to the Design of Low Speed Airfoils," NASA TM-X-3213, March 1975.

[58] Lyness, J. N., and Moler, C. B., "Numerical Differentiation of Analytic Functions," *SIAM Journal on Numerical Analysis*, Vol. 4, No. 2, 1967, pp. 202–210.

[59] Squire, W., and Trapp, G., "Using Complex Variables to Estimate Derivatives of Real Functions," *SIAM Review*, Vol. 40, No. 1, 1998, pp. 110–112.

[60] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, 2003, pp. 245–262.

[61] Nielsen, E. J., and Kleb, W. L., "Efficient Construction of Discrete Adjoint Operators on Unstructured Grids by Using Complex Variables," *AIAA Journal*, Vol. 44, No. 4, 2006, pp. 827-836.

[62] Sturdza, P., "An Aerodynamic Design Method for Supersonic Natural Laminar Flow Aircraft," Ph.D. thesis, Stanford University, December 2003.

[63] Newman, J. C., Whitfield, D. L., and Anderson, W. K., "Step-size Independent Approach for Multidisciplinary Sensitivity Analysis," *Journal of Aircraft*, Vol. 40, No. 3, 2003, pp. 566-573.

[64] Anderson, W. K., Newman, J. C., Whitfield, D. L., and Nielsen, E. J., "Sensitivity Analysis for the Navier-Stokes Equations on Unstructured Meshes Using Complex Variables," *AIAA Journal*, Vol. 39, No. 1, 2001, pp. 56-63.

[65] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "High-fidelity Aerostructural Design Optimization of a Supersonic Business Jet," *Journal of Aircraft*, Vol. 41, No. 3, 2004, pp. 523–530.

[66] Burgreen, G. W., Baysal, O., and Eleshaky, M. E., "Improving the Efficiency of Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 32, No. 1, 1994, pp. 69–76.

[67] Le Moigne, A., and Qin, N., "Variable-Fidelity Aerodynamic Optimization for Turbulent Flows Using a Discrete Adjoint Formulation," *AIAA Journal*, Vol. 42, No. 7, 2004, pp. 1281–1292.

[68] Fillola, G., Le Pape, M., and Montagnac, M., "Numerical Simulations Around Wing Control Surfaces," *24th International Congress of the Aeronautical Sciences*, Yokohama, Japan, August 2004.

[69] Mavriplis, D. J., "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes," *AIAA Journal*, Vol. 44, No. 1, 2006, pp. 42-50.

[70] Nielsen, E. J., and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[71] Bischof, C., Carle, A., Khademi, P., and Mauer, A., "ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs," *IEEE Computational Science and Engineering*, Vol. 3, No. 3, 1996, pp. 18–32.

[72] Carle, A., Fagan, M., and Green, L. L., "Preliminary Results from the Application of Automated Adjoint Code Generation to CFL3D," *7th AIAA/USAF/NASA/ISSMO Symposium*, St. Louis, USA, AIAA 98-4807, September 1998.

[73] Newman III, J. C., Taylor III, A. C., Barnwell, R. W., Newman, P. A., and Hou, G. J. W., "Overview of Sensitivity Analysis and Shape Optimization for Complex Aerodynamic Configurations," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 87-96.

[74] Newman, J. C., and Taylor, A. C., "Three-dimensional Shape Sensitivity Analysis and Design Optimization Using the Euler Equations on Unstrucutred Grids," *14th Applied Aerodynamics Conference*, New Orleans, USA, AIAA 96-2464, June 1996.

[75] Bischof, C. H., Mauer, A., Jones, W. T., and Samareh, J., "Experiences with Automatic Differentiation Applied to a Volume Grid Generation Code," *Journal of Aircraft*, Vol. 35, No. 4, 1998, pp. 569–573.

[76] Sadrehaghighi, I., Robert E. Smith, and Tiwari, S. N., "Grid Sensitivity and Aerodynamic Optimization of Generic Airfoils," *Journal of Aircraft*, Vol. 32, No. 6, 1995, pp. 1234-1239.

[77] Korivi, V. M., Newman, P. A., and Taylor, III, A. C., "Aerodynamic Optimization Using Sensitivity Derivatives From a Three-dimensional Supersonic Euler Code," *Journal of Aircraft*, Vol. 35, No. 3, 1998, pp. 405–411.

[78] Pagaldipti, N., and Chattopadhyay, A., "A Discrete Semianalytical Procedure for Aerodynamic Sensitivity Analysis Including Grid Sensitivity," *Computers & Mathematics with Applications*, Vol. 32, No. 3, 1996, pp. 61-71.

[79] Morris, A. M., Allen, C. B., and Rendall, T. C. S., "Domain-Element Method for Aerodynamic Shape Optimization Applied to a Modern Transport Wing," *AIAA Journal*, Vol. 47, No. 7, 2009, pp. 1647– 1659.

[80] Tennekes, H., and Lumley, J., "A First Course in Turbulence," MIT Press, Cambridge, 1972.

[81] Parviz, M., and Kim, J., "Tackling Turbulence with Supercomputers," *Scientific American*, Vol. 276, No. 1, 1997, pp. 62-68.

[82] Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *Recherche Aerospatiale*, No. 1, 1994, pp. 5-21.

[83] Chien, K.-Y., "Predictions of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model," *AIAA Journal*, Vol. 20, No. 1, 1982, pp. 33-38.

[84] Wilcox, D. C., "Reassessment of the Scale-Determining Equation for Advanced Turbulence Models," *AIAA Journal*, Vol. 26, No. 11, 1988, pp. 1299-1310.

[85] Menter, F. R., "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598-1605.

[86] Edwards, J. R., and Chandra, S., "Comparison of Eddy Viscosity-transport Turbulence Models for Three-dimensional, Shock-separated Flowfields," *AIAA Journal*, Vol. 34, No. 4, 1996, pp. 756–763.

[87] Rung, T., Bunge, U., Schatz, M., and Thiele, F., "Restatement of the Spalart-Allmaras Eddy-Viscosity Model in a Strain-Adaptive Formulation," *AIAA Journal*, Vol. 41, No. 7, 2003, pp. 1396-1399.

[88] Spalart, P. R., and Rumsey, C. L., "Modification and Clarification for the Implementation of the Spalart-Allmaras Turbulence Model," *7th International Conference on Computational Fluid Dynamics*, ICCFD7-1902, Big Island, USA, July 2012.

[89] Mavriplis, D. J., Vassberg, J. C., Tinoco, E. N., Mani, M., Brodersen, O. P., Eisfeld, B., Wahls, R. A., Morrison, J. H., Zickuhr, T., Levy, D., and Murayama, M., "Grid Quality and Resolution Issues from the Drag Prediction Workshop Series," *Journal of Aircraft*, Vol. 46, No. 3, 2009, pp. 935-950.

[90] Gerhold, T., Galle, M., Friedrichs, O., and Evans, J., "Calculation of Complex Three-Dimensional Configurations Employing the DLR TAU-Code," *35th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 1997-0167, January 1997.

[91] Jameson, A., and Yoon, S., "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA Journal*, Vol. 25, No. 7, 1987, pp. 929-935.

[92] Turkel, E., "Preconditioning Techniques in Computational Fluid Dynamics," *Annual Review of Fluid Mechanics*, Vol. 31, No. 1, 1999, pp. 385-416.

[93] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-stepping Schemes," *14th AIAA Fluid and Plasma Dynamics Conference*, Palo Alto, USA, AIAA 1981-1259, June 1981.

[94] Jameson, A., "Analysis and Design of Numerical Schemes for Gas Dynamics: Artificial Diffusion, Upwind Biasing, Limiters and their Effect on Accuracy and Multigrid Convergence," *International Journal of Computational Fluid Dynamics*, Vol. 4, No. 3-4, 1995, pp. 171-218.

[95] Steger, J. L., "Implicit Finite-Difference Simulation of Flow About Arbitrary Two-Dimensional Geometries," *AIAA Journal*, Vol. 17, No. 7, 1978, pp. 679-686.

[96] Venkatakrishnan, V., "Perspective on Unstructured Grid Flow Solvers," *AIAA Journal*, Vol. 34, No. 3, 1996, pp. 533-547.

[97] Yoon, S., and Jameson, A., "Lower-Upper Symmetric-Gauss-Seidel Method for the Euler and Navier-Stokes equations," *AIAA Journal*, Vol. 26, No. 9, 1988, pp. 1025-1026.

[98] Newman III, J. C., Taylor III, A. C., Barnwell, R. W., Newman, P. A., and Hou, G. J. W., "Overview of Sensitivty Analysis and Shape Optimization for Complex Aerodynamic Configuration," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 87-89.

[99] Laflin, K. R., Klausmeyer, S. M., Zickuhr, T., Vassberg, J. C., Wahls, R. A., Morrison, J. H., Brodersen, O. P., Rakowitz, M. E., Tinoco, E. T., and Godard, J-L., "Data Summary From Second AIAA Computational Fluid Dynamics Drag Prediction Workshop," *Journal of Aircraft*, Vol. 42, No. 5, 2005, pp. 1165-1178.

[100] Truong, A. H., Oldfield, C. A., and Zingg, D. W., "Mesh Movement for a Discrete-adjoint Newton-Krylov Algorithm for Aerodynamic Optimization," *AIAA Journal*, Vol. 46, No. 7, 2008, pp. 1695-1704.

[101] Samareh, J. A., “Application of Quaternions for Mesh Deformation,” NASA TM-2002-211646, April 2002.

[102] Nambu, T., Mavriplis, D. J., and Mani, K., "Adjoint-based Shape Optimization of High-lift Airfoils Using the NSU2D Unstructured Mesh Solver," *52nd AIAA Aerospace Sciences Meeting*, National Harbor, USA, AIAA 2014-0554, January 2014.

[103] Yang, Z., and Mavriplis, D. J., "Mesh Deformation Strategy Optimized by the Adjoint Method on Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 12, 2007, pp. 2885-2896.

[104] Batina, J. T., "Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex-Aircraft Aerodynamic Analysis," *AIAA Journal*, Vol. 29, No. 3, 1991, pp. 327–333.

[105] Stein, K., Tezduyar, T., and Benney, R., “Mesh Moving Techniques for Fluid-Structure Interactions with Large Displacements,” *Journal of Applied Mechanics*, Vol. 70, No. 1, 2003, pp. 58–63.

[106] Helenbrook, B. T., "Mesh Deformation Using the Biharmonic Operator," *International Journal for Numerical Methods in Engineering*, Vol. 56, No. 7, 2003, pp. 1007-1021.

[107] Beckert, A., and Wendland, H., "Multivariate Interpolation for Fluid-structure-interaction Problems Using Radial Basis Functions," *Aerospace Science and Technology*, Vol. 5, No. 2, 2001, pp. 125-134.

[108] Witteveen, J. A. S, and Hestre, B., "Explicit Mesh Deformation Using Inverse Distance Weighting Interpolation," *19th AIAA Computational Fluid Dynamics*, San Antonio, USA, AIAA 2009-3996, June 2009.

[109] Mavriplis, D. J., and Yang, Z., "Unstructured Dynamic Meshes with Higher-order Time Integration Schemes for the Unsteady Navier-Stokes Equations," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, USA, Nevada, AIAA 2005-1222, January 2005.

[110] Loehner, R., and Yang, C., "Improved ALE Mesh Velocities for Moving Bodies," *Communications in Numerical Methods in Engineering*, Vol. 12, No. 10, 1996, pp. 599-608.

[111] Blom, F. J., "Considerations on the Spring Analogy," *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, 2000, pp. 647–668.

[112] Byun, C., and Guruswamy, G. P., "A Parallel Multi-block Moving Grid Method for Aeroelastic Applications on Full Aircraft," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Luis, USA, AIAA 98-4782, September 1998.

[113] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional Springs for Two-dimensional Dynamic Unstructured Fluid Meshes," C*omputer Methods in Applied Mechanics and Engineering*, Vol. 163, No. 1, 1998, pp. 231-245.

[114] Murayama, M., Nakahashi, K., and Matsushima, K., "Unstructured Dynamic Mesh for Large Movement and Deformation," *40th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2002-0122, January 2002.

[115] Luke, E., Collins, E., and Lades, E., "A Fast Mesh Deformation Method Using Explicit Interpolation," *Journal of Computational Physics*, Vol. 231, No. 2, 2012, pp. 586-601.

[116] Lefrançois, E., "A Simple Mesh Deformation Technique for Fluid Structure Interaction Based on a Submesh Approach," *International Journal for Numerical Methods in Engineering*, Vol. 75, No.9, 2008, pp. 1085–1101.

[117] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-free Approach to High-Fidelity Aerostructural Optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, USA, AIAA 2010-9231, September 2010.

[118] Wang, Y., Qin, N., and Zhao, N., "Delaunay Graph and Radial Basis Function for Fast Quality Mesh Deformation," *Journal of Computational Physics*, Vol. 294, 2015, pp. 149-172.

[119] Žalik, B., and Lamot, M., "Contribution to Triangulation Algorithms for Simple Polygons," *Journal of Computing and Information Technology*, Vol. 8, No. 4, 2000, pp. 319-331.

[120] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H., "The Quickhull Algorithm for Convex Hull," *ACM Transactions on Mathematical Software*, Vol. 22, No. 4, 1996, pp. 469-483.

[121] Mavriplis, D. J., "Unstructured Grid Techniques," *Annual Review of Fluid Mechanics*, Vol. 29, No. 1, 1997, pp. 473-514.

[122] Xiao, T., Ang, H., and Tong. C., "A New Dynamic Mesh Generation Method for Large Movements of Flapping Wings with Complex Geometries," *Acta Aeronautica et Astronautica Sinica*, Vol. 29, No. 1, 2008, pp. 41-48.

[123] McDaniel, D. R., and Morton, S. A., "Efficient Mesh Deformation for Computational Stability and Control Analyses on Unstructured Viscous Meshes," 4*7th AIAA Aerospace Sciences Meeting*, Orlando, USA, AIAA 2009-1363, January 2009.

[124] Durrani, N., Chaudhry, S. R., and Qin, N., "Delaunay Graph Mapping Based Mesh Deformation for Simulation of a Spanwise Rigid and Flexible Flapping NACA0012 Wing Using DES with Parallel Implementation," *49th AIAA Aerospace Sciences Meeting*, Orlando, USA, AIAA 2011-1266, January 2011.

[125] Sun, S., Lv, S., Yuan, Y., and Yuan, M., "Mesh Deformation Method Based on Mean Value Coordinates Interpolation," *Acta Mechanica Solida Sinica,* Vol. 29, No. 1, 2016, pp. 1-12.

[126] Li, J., Gao, Z., Huang, J., and Zhao, K., "Aerodynamic Design Optimization of Nacelle/Pylon Position on an Aircraft," *Chinese Journal of Aeronautics*, Vol. 26, No. 4, 2013, pp. 850-857.

[127] Yang, Z., and Mavriplis, D. J., "Mesh Deformation Strategy Optimized by the Adjoint Method on Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 12, 2007, pp. 2885-2896.

[128] Dwight, R., “Robust Mesh Deformation Using the Linear Elasticity Equations,” in Computational Fluid Dynamics 2006 (H. Deconinck and E. Dick, eds.), pp. 401–406, Springer, 2006.

[129] Biedron, R. T., and Thomas, J. L., "Recent Enhancements To The FUN3D Flow Solver For Moving-Mesh Applications," *47th AIAA Aerospace Sciences Meeting*, Orlando, USA, AIAA 2009-1360, January 2009.

[130] Barrala, N., Lukeb, E., and Alauzet, F., "Two Mesh Deformation Methods Coupled with a Changing-connectivity Moving Mesh Method for CFD Applications," *23rd International Meshing Roundtable*, Vol. 82, 2014, pp. 213-227.

[131] Vassberg, J. C., Sclafani, J. A., and DeHaan, M. A., "A Wing-body Fairing Design for the DLR-F6 model: a DPW-III Case Study," *23rd AIAA Applied Aerodynamics Conference*, Toronto, Canada, AIAA 2005-4730, June 2005.

[132] Fulker, J. L., and Simmons, M. J., "An Experimental Study of Shock Control Methods," DRA/AS/HWA/TR94007/1, 1994.

[133] Schmitt, V., and Charpin, F., "Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers," AGARD AR-138, May 1979.

[134] Hou, G. J. W., Maroju, V., Taylor, A. C., and Korivi, V. M., "Transonic Turbulent Airfoil Design Optimization with Automatic Differentiation in Incremental Iterative Forms," *12th Computational Fluid Dynamics Conference*, San Diego, USA, AIAA 95-1692, 1995.

[135] Anderson, W. K., and Bonhaus, D. L., "Airfoil Design on Unstructured Grids for Turbulent Flows," *AIAA Journal*, Vol. 37, No. 2, 1999, pp. 185-191.

[136] Leatham, M., Stokes, S., Shaw, J. A., Cooper, J., Appa, J., and Blaylock, T. A., "Automatic Mesh Generation for Rapid-Response Navier-Stokes Calculations," *Fluids 2000 Conference and Exhibit*, Denver, USA, AIAA 2000-2247, June 2000.

[137] Kim, C., Kim, C., and Rho, O., "Effects of Constant Eddy Viscosity Assumption on Gradient-based Design Optimization," *40th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2002-0262, January 2002.

[138] Khayatzadeh, P., and Nadarajah, S. K., "Aerodynamic Shape Optimization via Discrete Viscous Adjoint Equations for the k− ωSST Turbulence and γ− Reθ Transition Models," *49th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, USA, AIAA 2011–1247, January 2011.

[139] Giles M. B., and Suli, E., "Adjoint Methods for PDEs: a Posteriori Error Analysis and Postprocessing by Duality," *Acta Numerica*, Vol. 11, 2002, pp. 145–236.

[140] Venditti, D. A., and Darmofal, D. L., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-dimensional Viscous Flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22 – 46.

[141] Giles, M. B., and Pierce, N. A., "Improved Lift and Drag Estimates Using Adjoint Euler Equations," *14th AIAA Computational Fluid Dynamics Conference,* Norfolk, USA, AIAA 99-3293, November 1999.

[142] Harris, M. J., and Qin, N., "Geometric Representation of Flow Features Using the Medial Axis for Mesh Generation," *AIAA Journal,* Vol. 53, No. 1, 2015, pp. 246-259.

[143] Gatlin, G. M., Rivers, M. B., Goodliff, S. L., Rudnik, R., and Sitzmann, M., "Experimental Investigation of the DLR-F6 Transport Configuration in the National Transonic Facility," *26th AIAA Applied Aerodynamics Conference,* Honolulu, USA, AIAA 2008-6917, August 2008.

[144] Samareh, J. A., "Survey of Shape Parametrization Techniques for High-Fidelity Multidisciplinary Shape Optimization," *AIAA Journal,* Vol. 39, No. 5, 2001, pp. 877–884.

[145] Yamazaki, W., Mouton, S., and Carrier, G., "Efficient Design Optimization by Physics-based Direct Manipulation Free-Form Deformation," *AIAA Journal,* Vol. 48, No. 8, 2010, pp. 1817-1832.

[146] Hsu, W. M., Hughes, J. F., and Kaufman, H., "Direct Manipulation of Free-Form Deformations," *Computer Graphics,* Vol. 26, No. 2, 1992, pp. 177-184.

[147] Campbell, R. L., "An Approach to Constrained Aerodynamic Design with Application to Airfoils," NASA TP-3260, November 1992.

[148] Khurana, M. S., Winarto, H., and Sinha, A., "Airfoil Geometry Parameterization Through Shape Optimizer and Computational Fluid Dynamics," *46th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, USA, AIAA 2008-295, January 2008.

[149] Eyi, S., and Lee, K. D., "Inverse Airfoil Design Using the Navier Stokes Equations," *Engineering Optimization,* Vol. 28, No. 4, 1997, pp. 245-262.

[150] Cosentino, G. B., and Holst, T. L., "Numerical Optimization Design of Advanced Transonic Wing Configurations," *Journal of Aircraft,* Vol. 23, No. 3, 1986, pp. 192-199.

[151] Farin, G., "Curves and Surfaces for Computer-aided Geometric Design," Academic Press, San Diego, 1988.

[152] Sasaki, D., and Obayashi, S., "Low-boom Design Optimization for SST Canard-Wing-Fuselage Configuration," *16th AIAA Computational Fluid Dynamics Conference*, Orlando, USA, AIAA 2003-3432, June 2003.

[153] Lepine, J., Guibault, F., Trepanier, J. Y., and Pepin, F., "Optimized Nonuniform Rational B-spline Geometrical Representation for Aerodynamic Design of Wings," *AIAA Journal*, Vol. 39, No. 11, pp. 2033-2041.

[154] Bentamy, A., Guibault, F., and Trepanier, J., "Aerodynamic Optimization of a Realistic Aircraft Wing," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2005-332, January 2005.

[155] Aftosmis, M. J., Delanaye, M., and Haimes, R., "Automatic Generation of CFD-ready Surface Triangulations from CAD Geometry*," 37th Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 1999-776, January 1999.

[156] Nemec, M., Aftosmis, M. J., and Pulliam, T. H., "CAD-based Aerodynamic Design of Complex Configurations Using a Cartesian Method," *42nd AIAA Aerospace Sciences Conference*, Reno, USA, AIAA 2004-0113, January 2004.

[157] Sobieczky, H., "Parametric Airfoils and Wings," Notes on Numerical Fluid Mechanics, edited by Fujii, K., and Dulikravich, G. S., Vol. 68, Vieweg, Brunswick, Germany, 1998, pp. 71–88.

[158] Fuhrmann, H., "Design Optimisation of a Class of Low Reynolds, High Mach Number Airfoils for Use in the Martian Atmosphere," *23rd AIAA Applied Aerodynamics Conference*, Toronto, Canada, AIAA 2005-4603, June 2005.

[159] Khurana, M. S., Winarto, H., and Sinha, A. K., "Airfoil Optimization by Swarm Algorithm with Mutation and Artificial Neural Networks," *47th AIAA Aerospace Sciences Meeting*, Orlando, USA, AIAA 2009-1278, August 2009.

[160] Coquillart, S., "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling," *ACM SIGGRAPH Computer Graphics*, Vol. 24, No. 4, 1990, pp. 187–193.

[161] Hsu, W. M., Hughes, J. F., and Kaufman, H., "Direct Manipulation of Free-Form Deformations," *ACM SIGGRAPH Computer Graphics*, Vol. 26, No. 2, 1992, pp. 177–184.

[162] Amoiralis, E., I., and Nikolos, I. K., "Freeform Deformation Versus B-spline Representation in Inverse Airfoil Design," *Journal of Computing and Information Science in Engineering,* Vol. 8, No. 2, 2008.

[163] Lamousin, H. J., and Waggenspack, W. N., "NURBS-Based Free-Form Deformations," *IEEE Computer Graphics and Applications*, Vol. 14, No. 6, 1994, pp. 59–65.

[164] Samareh, J. A., "Geometry and Grid/Mesh Generation Issues for CFD and CSM Shape Optimization," *Optimization and Engineering,* Vol. 6, No. 1, 2005, pp. 21-32.

[165] Bloor, M. I., and Wilson, M. J., "Using Partial Differential Equations to Generate Free-Form Surfaces," *Computer-Aided Design,* Vol. 22, No. 4, 1990, pp. 202-212.

[166] Zhu, F., and Qin, N., "Intuitive Class/Shape Function Parameterization for Airfoils," *AIAA Journal,* Vol. 52, No. 1, 2014, pp. 17-25.

[167] Ceze, M., Hayashi, M., and Volpe, E., "A Study of the CST Parameterization Characteristics," *27th AIAA Applied Aerodynamics Conference,* San Antonio, USA, AIAA 2009-3767, June 2009.

[168] Marshall, D. D., "Creating Exact Bezier Representations of CST Shapes," *21st AIAA Computational Fluid Dynamics Conference,* San Diego, USA, AIAA 2013-3077, June 2013.

[169] Wu., L., and Padula, S., "Using High Resolution Design Spaces for Aerodynamic Shape Optimization Under Uncertainty," NASA TP-2004-213004, March 2004.

[170] Sóbester, A., and Barrett, T., "The Quest for a Truly Parsimonious Airfoil Parameterisation Scheme," *8th AIAA 2008 ATIO Conference,* Anchorage, USA, AIAA 2008-8879, September 2008.

[171] Sóbester, A., "Concise Airfoil Representation via Case-Based Knowledge Capture," *AIAA Journal,* Vol. 47, No. 5, 2009, pp. 1209-1218.

[172] Straathof, M. H., and Van Tooren, M. J. L., "Extension to The Class-Shape-Transformation Method Based on B-Splines," *AIAA Journal,* Vol. 49, No. 4, 2011, pp. 780-790.

[173] Punj, A., Govil, R., and Balasundaram, S., "A New Approach in Designing of Local Controlled Curves and Surfaces," *Applied Mathematics Letters*, Vol. 10, No. 1, 1997, pp. 89-94.

[174] Painchaud-Ouellet, S., Tribes, C., Trepanier, J.-Y., and Pelletier, D., “Airfoil Shape Optimization Using a Nonuniform Rational B-Splines Parameterization Under Thickness Constraint," *AIAA Journal*, Vol. 44, No. 10, 2006, pp. 2170-2178.

[175] Giunta, A. A., and Watson, L. T., "A Comparison of Approximation Modeling Techniques: Polynomial Versus Interpolating Models," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, USA, AIAA 98-4758, September 1998.

[176] Schmidt, S., Ilic, C., Schulz, V., and Gauger, N. R., "Three-dimensional Large-scale Aerodynamic Shape Optimization Based on Shape Calculus," *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2615-2627.

[177] Reuther, J., Jameson, A., Alonso, J., J., Rimlinger, M., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–74.

[178] Wu, L., and Padula, S., "Using High Resolution Design Spaces for Aerodynamic Shape Optimization under Uncertainty," NASA TP-2004-213004, March 2004.

[179] Protas, B., Bewley, T. R., and Hagen, G., "A Computational Framework for the Regularization of Adjoint Analysis in Multiscale PDE Systems," *Journal of Computational Physics*, Vol. 195, No. 1, 2004, pp. 49-89.

[180] Eppler, K., Schmidt, S., Schulz, V., and Ilic, C., "Preconditioning the Pressure Tracking in Fluid Dynamics by Shape Hessian Information," *Journal of Optimization Theory and Applications*, Vol. 141, No. 3, 2009, pp. 513-531.

[181] Barger, R. L., "Streamline Curvature Design Procedure for Subsonic and Transonic Ducts," NASA TN D-7368, December 1973.

[182] Stück, A., and Rung, T., "Filtered Gradients for Adjoint-based Shape Optimisation," *20th AIAA Computational Fluid Dynamics Conference*, Honolulu, USA, AIAA 2011-3072, June 2011.

[183] Neuberger, J. W., "Steepest Descent and Differential Equations," *Journal of the Mathematical Society of Japan*, Vol. 37, No. 2, 1985, pp. 187-195.

[184] Kim, S., Hosseini, K., Leoviriyakit, K., and Jameson, A., "Enhancement of Class of Adjoint Design Methods via Optimization of Parameters," *AIAA Journal*, Vol. 48, No. 6, 2010, pp. 1072-1076.

[185] Kim, H. J., Koc, S., and Nakahashi, K., "Surface Modification Method for Aerodynamic Design Optimization," *AIAA Journal*, Vol. 43, No. 4, 2005, pp. 727-740.

[186] Wu, L., Krist, S., and Campbell, R., "Transonic Airfoil Shape Optimization in Preliminary Design Environment," *Journal of Aircraft*, Vol. 43, No. 3, 2006, pp. 639-651.

[187] Reuther, J., Alonso, J., Rimlinger M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers," *Journal of Aircraft*, Vol. 36, No. 2, 1999, pp. 61-74.

[188] McKinnon, K. I. M, "Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point," *SIAM Journal on Optimization*, Vol. 9, No. 1, 1998, pp. 148-158.

[189] Hestenes, M. R., and Stiegel. E., "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, Vol. 49, No. 6, 1952, pp. 409-436.

[190] Zhoujie, L., Xu, Z., and Martins, J. R. R. A., "Benchmarking optimization algorithms for wing aerodynamic design optimization," *Proceedings of the 8th International Conference on Computational Fluid Dynamics*, Chengdu, China, 2014.

[191] Johnson, S. G., "The NLopt Nonlinear-optimization Package," http://ab-initio.mit.edu/nlopt (accessed on January 2017).

[192] Kraft, D., "Algorithm 733: TOMP–Fortran Modules for Optimal Control Calculations," *ACM Transactions on Mathematical Software*, Vol. 20, No. 3, 1994, pp. 262-281.

[193] Liu, D. C., and Nocedal, J., "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, Vol. 45, No. 1-3, 1989, pp. 503-528.

[194] Fletcher, R., "A New Approach to Variable Metric Algorithms," *The Computer Journal*, Vol. 13, No. 3, 1970, pp. 317-322.

[195] Elliott, J., and Peraire, J., "Aerodynamic Optimization on Unstructured Meshes with Viscous Effects," *13th AIAA Computational Fluid Dynamics Conference*, Snowmass Village, USA, AIAA 97-1849, June 1997.

[196] Rumpfkeil, M., and Zingg, D. W., "Unsteady Optimization Using a Discrete Adjoint Approach Applied to Aeroacoustic Shape Design," *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, USA, AIAA 2008-18, January 2008.

[197] Zhu, F., and Qin, N., "Using Mesh Adjoint for Shock Bump Deployment and Optimisation on Transonic Wings," *53rd AIAA Aerospace Sciences Meeting and Exhibit*, Kissimmee, USA, AIAA 2015-1488, January 2015.

[198] Bobrowski, K., Ferrer, E., Valero, E., and Holger, B., "CAD-based Aerodynamic Shape Optimization Using Geometry Surrogate Model and Adjoint Methods," *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization*, Washington, USA, AIAA 2016-3831, June 2016.

[199] Jakobsson, S., and Amoignon, O., "Mesh Deformation Using Radial Basis Functions for Gradient-Based Aerodynamic Shape Optimization," *Computers and Fluids*, Vol. 36, No. 6, 2007, pp. 1119–1136.

[200] Mavriplis, D. J., "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes," *AIAA Journal*, Vol. 44, No. 1, 2006, pp. 42-50.

[201] Maute, K., Nikbay, M., and Farhat, C., "Sensitivity Analysis and Design Optimization of Three-Dimensional Non-Linear Aeroelastic Systems by the Adjoint Method," *International Journal for Numerical Methods in Engineering*, Vol. 56, No. 6, 2003, pp. 911-933.

[202] Burgreen, G. W., and Baysal, O., "Three-Dimensional Aerodynamic Shape Optimization Using Discrete Sensitivity Analysis," *AIAA Journal*, Vol. 34, No. 9, 1996, pp. 1761–1770.

[203] Fulker, J. L., Ashill, P. R., and Simmons, M. J., "Study of Simulated Active Control of Shock Waves on an Aerofoil," Tech. Rep. 93025, DERA, May 1993.

[204] Wong, W. S., Le Moigne, A., and Qin, N., "Parallel Adjoint-based Optimisation of a Blended Wing Body Aircraft with Shock Control Bumps," *Aeronautical Journal*, Vol. 111, No. 7, 2007, pp. 165–174.

[205] Özkaya, E., and Gauger, N., "Automatic Transition from Simulation to One-Shot Shape Optimization with Navier–Stokes Equations," *GAMMMitteilungen,* Vol. 33, No. 2, 2010, pp. 133–147.

[206] Wu, H. Y., Yang, S., Liu, F., and Tsai, H. M., "Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization," *16th AIAA Computational Fluid Dynamics Conference*, Orlando, USA, AIAA 2003-4095, June 2003