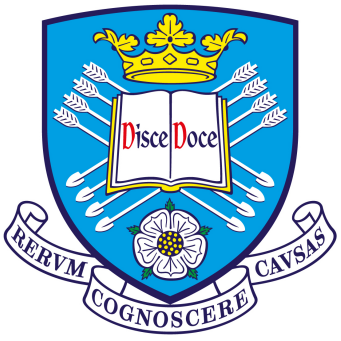
University of Sheffield



Anisotropic & Edgewise Adjoint Error Estimation & Grid Adaptation with Applications to Turbomachinery Flows

by

Guglielmo Vivarelli

A thesis submitted in partial fulfillment for the degree of
Doctor of Philosophy

in the
Faculty of Engineering
Department of Mechanical Engineering

March 2019

University of Sheffield

Abstract

Faculty of Engineering
Department of Mechanical Engineering

Doctor of Philosophy

by Guglielmo Vivarelli

Nowadays, *Computational Fluid Dynamics* is employed on a daily basis, in various industrial settings, to approximate understand and anticipate the behaviour of varied and alternate geometries. Despite the advancements in computational power and flow solver developments, accuracy is still an issue. To overcome this, mesh adaptation could be a solution, as it would, if correctly employed, allow to automatically and efficiently reduce sources of error. However, this technology has found very limited use outside academia, as to date, it still lacks the necessary robustness.

The most promising grid adaptation technique is based on adjont-error estimation, and has consistently shown to outperform feature-related approaches. However, so far, it has not been employed to reduce inaccuracies for turbomachinery cases. The objective of this research work, is to develop new adjoint-based processes to improve the performance quantity estimates with respect to much finer grids and apply them to jet engine components. Moreover, to reduce the cost of these procedures, novel and efficient methods will be devised. The first, will use feature-based mesh movement to align the grid edges, and then cell-based adjoint refinement. It will be shown that the first helps improve dual-error accuracy while being inexpensive. The second approach will employ mesh-adjoint-based mesh movement to cluster nodes towards the functional's regions of high sensitivity. It will be shown that this improves the quantity's estimate while aligning the grid with flow and dual complexities. Moreover, it will be proven that it reduces the standard computation of the adjoint-error. Once the quantity of interest has converged, mesh refinement will be used. Two different edgewise formulations of the adjoint-error will be employed to adapt the grid. As will be seen, in all cases the performance quantities improve significantly.

Acknowledgements

I would like to start by expressing my gratitude towards Professor *Ning Qin* and Professor *Shahrokh Shahpar*. I greatly appreciate the opportunity they gave me, as the project has proven to be very interesting and challenging. Moreover, due to their professional ability and great experience, their supervision throughout the work has been invaluable.
I also ought to mention my appreciation for Professor *Shahpar*'s effort in gaining extra funding, thereby allowing me to carry out the remainder of the work.
A necessary separate acknowledgement is required for *David Radford*. His patience in answering the many questions I had about *Hydra*'s solvers, has proven invaluable for my understanding. Moreover, his and Dr *Indi Tristanto*'s effort in reviewing parts of this thesis is greatly appreciated.
I would also like to thank (in alphabetical order) *A. Moreno, Dr D. De Grazia, D. Sgro', G. Occhioni & M. Ugolotti*, as they proved to be great company and for their moral support during my stay at *Rolls-Royce plc*.
A special mention is also necessary for the entire *CFD Methods* team at *Rolls-Royce plc, Derby*.
Last but not least, I ought to thank my (many) fellow researchers at the *University of Sheffield* for making my journey a much more pleasant experience.
Finally, the project has been fully funded by *Rolls-Royce plc* which I gratefully acknowledge.

List of Publications

Conference Papers

- G. Vivarelli, N. Qin & S. Shahpar, "*Combined Hessian and Adjoint Error-Based Anisotropic Mesh Adaptation for Turbomachinery Flows*", 55^{th} AIAA Aerospace Sciences Meeting, p. 1946, 2017
- G. Vivarelli, N. Qin, S. Shahpar & D. Radford, "*Efficient Adjoint-Based Mesh Adaptation Applied to Turbo-Machinery Flows*", ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition, American Society of Mechanical Engineers, 2018

Journal Articles

- G. Vivarelli, N. Qin, S. Shahpar & D. Radford, "*Advances in Adjoint-Based Mesh Adaptation I: Sequential Hessian & Adjoint Error-Based Anisotropic Mesh Adaptation*" (in preparation)
- G. Vivarelli, N. Qin, S. Shahpar & D. Radford, "*Advances in Adjoint-Based Mesh Adaptation II: Efficient, Edgewise & Anisotropic Adjoint Error-Based Mesh Adaptation*" (in preparation)

Contents

List of Figures

List of Tables

Names & Abbreviations

AD	**A**utomatic **D**ifferentiation
BC	**B**oundary **C**onditions
C	Programming language
AIAA	**A**merican **I**nstitute of **A**eronautics & **A**stronautics
CAD	**C**omputer **A**ided **D**esign
CFD	**C**omputational **F**luid **D**ynamics
CFL	**C**ourant-**F**riedrichs-**L**ewy number
CGNS	**C**FD **G**eneral **N**otation **S**ystem
CPU	**C**entral **P**rocessing **U**nit
DG	**D**iscontinuous **G**alerkin
DLR	**D**eutsches Zentrum für **L**uft- und **R**aumfahrt
FAS	**F**ull **A**pproximation **S**torage
FDM	**F**inite **D**ifference **M**ethod
FFD	**F**ree **F**orm **D**eformation
FEM	**F**inite **E**lement **M**ethod
FVM	**F**inite **V**olume **M**ethod
Fortran 77	**For**mula **Tran**slation version 77 programming language
GG	**G**reen **G**auss gradient operator
GMRES	**G**eneralised **M**inimum **RES**idual solver
HDF5	**H**ierarchical **D**ata **F**ormat 5
HH	**H**igh **H**igh frequencies
HL	**H**igh **L**ow frequencies
INRIA	**I**nstitut **N**ational de **R**echerche en **I**nformatique et en **A**utomatique
JST	**J**ameson **S**chmidt **T**urkel flux scheme
LE	**L**eading **E**dge

LH	**L**ow **H**igh frequencies
LHS	**L**eft **H**and **S**ide
LL	**L**ow **L**ow frequencies
LSQ	**L**east **SQ**uares gradient operator
MATLAB®	**Mat**rix **Lab**oratory programming software
MeshPost	**Mesh Post**-processing software
MP	**MeshPost**
MUSCL	**M**onotone **U**pwind **S**calar **C**onservation **L**aw
NACA	**N**ational **A**dvisory **C**ommittee for **A**eronautics
NASA	**N**ational **A**eronautics & **S**pace **A**dministration
NS	**N**avier-**S**tokes equations
ONERA	**O**ffice **N**ational d'**E**tudes et de **R**echerches **A**érospatiales
OPLUS	**O**xford **P**arallel **L**ibrary for **U**nstructured **S**olvers
PADRAM	**P**arametric **D**esign and **RA**pid **M**eshing
QR	Factorisation algorithm used in *Least-Squares derivative* calculation
RAE	**R**oyal **A**ircraft **E**stablishment
RANS	**R**eynolds **A**veraged **N**avier **S**tokes
RHS	**R**ight **H**and **S**ide
RISC	**R**educed **I**nstruction **S**et **C**omputer
RK	**R**unge-**K**utta
RR	**R**olls-**R**oyce
SA	**S**palart-**A**llmaras turbulence model
SIAM	**S**ociety for **I**ndustrial and **A**pplied **M**athematics
TAPENADE	*Automatic Differentiation* software
TAU	**DLR** flow solver
TE	**T**railing **E**dge
TVD	**T**otal-**V**ariation **D**iminishing

de	**D**iscretisation **E**rror
h-adaptation	Mesh refinement or coarsening
lp	**L**inerity **P**reserving
m-adaptation	Mesh regeneration
p-adaptation	High-order polynomial local refinement

p.d.e.	**P**artial **D**ifferential **E**quation
r-adaptation	Mesh movement
te	**T**runcation **E**rror
w.r.t.	**W**ith-**R**espect-**T**o

Symbols

Greek Alphabet:

Λ	*Lagrangian* multiplier for *error equidistribution* analysis
Υ	High order prolongation operator
Φ	*Jameson-Schmidt-Turkel* switch
Ψ	Adjoint vector
Ψ_f	Flow adjoint vector
Ψ_g	Mesh adjoint vector
Ω	Domain of integration
$\partial\Omega$	Domain of integration boundary
α	*Runge-Kutta* convective coefficient
$\widehat{\alpha}_i$	Design variables
β	*Runge-Kutta* dissipative coefficient
γ	Specific heat ratio
ϵ	*Non-computable correction*
$\epsilon^{(2)}$	*Jameson-Schmidt-Turkel* coefficient
ϵ_e	Edgewise *non-computable correction*
ζ	Cell-wise error constant for *equidistribution principle*
η	Mass averaged adiabatic efficiency between inlet and outlet
η_{BP}	Mass averaged adiabatic efficiency between inlet and bypass outlet
η_{ES}	Mass averaged adiabatic efficiency between inlet and engine outlet
θ	Angle
κ	*MUSCL* scheme coefficient
$\widetilde{\kappa}$	Spring stiffness coefficient
$\widehat{\kappa}$	*Spalart-Allmaras* turbulence model constant

λ	*Hessian* matrix eigenvalues
$\widetilde{\lambda}$	*Jacobian* matrix eigenvalues
$\widehat{\lambda}$	Bulk viscosity
μ	Dynamic viscosity
$\overline{\mu}$	Constant dynamic viscosity
ν	Molecular kinematic viscosity
$\widetilde{\nu}$	*Eddy* viscosity
ν_t	Turbulent viscosity
ξ	*Hermitian* interpolation coefficient
π	180°
ρ	Density
σ	*Spalart-Allmaras* turbulence model constant
τ	Shear stress
υ	Linear prolongation operator
χ	Ratio of *eddy* to molecular viscosity
ω	Movement relaxation factor
$\widehat{\omega}$	Adjoint *Runge-Kutta* working variable

Latin Alphabet:

$\|A\|$	*Jacobian* matrix of inviscid fluxes
A^L	Linearised *Jacobian* matrix of inviscid fluxes
$\mathbf{B}$	Combination of dissipative terms in the standard *Runge-Kutta* scheme
$\mathbb{B}$	Set of boundary surfaces
$\widehat{B}$	Convective/dissipative perturbation coefficient matrix
$\widetilde{B}$	*Dirichlet* boundary condition matrix
C	Convective operator
$\mathbf{C}$	Convective terms in the standard *Runge-Kutta* scheme
C_D	Drag coefficient
C_L	Lift coefficient
$\mathbf{D}$	Dissipative terms in the standard *Runge-Kutta* scheme
$\widehat{D}$	Dissipative operator
E	Total internal energy

$\mathbb{E}$	Set of edges connected to a node
E_r	Cell-wise error
$\mathbf{F}$	Discrete flux vector
$\mathscr{F}$	Continuous flux
$\overrightarrow{F}$	Spring force
$\mathbf{F}^I$	Inviscid flux
$\mathbf{F}^V$	Viscous flux
G	Gradient and *pseudo-Laplacian* operator
H	Coarse mesh characteristic spacing
$\mathbf{H}$	*Hessian* matrix
$\overline{\mathbf{H}}$	Positive semi-definite *Hessian* matrix
I	Identity matrix
I_h^H	Restriction operator
I_H^h	Prolongation operator
K	Set of nodes
$\mathbf{K}$	Mesh spring stiffness matrix
L	*Laplacian* operator
$\mathscr{L}$	*Lagrangian* formulation for *flow adjoint* determination
$\widehat{L}$	Linearised *partial differential equation*
$\widehat{\mathscr{L}}$	*Lagrangian* formulation for *mesh adjoint* determination
L^{lp}	*Linearity preserving Laplacian* operator
L^{bnd}	Boundary node *Laplacian* operator
M	*Mach* number
$\mathbf{M}$	Set of mesh metrics
$\widehat{M}$	Number of *Runge-Kutta* stages $(3, 5, 7)$
N_C	Number of coarse mesh nodes
N_D	Number of design variables
N_F	Number of fine mesh nodes
N_N	Number of mesh nodes
N_f	Number of integral quantities
N_t	Final time step
P	Static pressure
$\mathscr{P}$	Spring potential energy

$\widehat{P}$	Preconditioning matrix
$\widehat{P}^I$	Inviscid preconditioning matrix
$\widehat{P}^V$	Viscous preconditioning matrix
Pr	Mass averaged absolute total pressure ratio between inlet and outlet
$\mathbf{Q}$	Vector of conservative variables
$\mathscr{Q}$	Continuous conservative variables
$\mathbf{Q}_{QR}$	Matrix for QR factorisation
R	Ideal gas constant
$\mathbf{R}$	Discretised residual operator (flow and adjoint solvers)
R	Periodic boundary rotation matrix
$\mathscr{R}$	Continuous *Navier-Stokes* eq. (residuals)
$\widehat{R}$	*Runge-Kutta* with preconditioner operator
$\mathbf{R}^e$	Edgewise residual operator (flow and adjoint solvers)
$\mathbf{R}_{QR}$	Matrix for QR factorisation
Re	*Re*ynolds Number
S	Continuous control surface
$\mathbf{S}$	Discrete *Navier-Stokes* source terms
$\mathscr{S}$	Continuous *Navier-Stokes* source terms
$\widetilde{S}$	*Spalart-Allmaras* turbulence model variable
$\widehat{S}$	Vorticity
$\widehat{\mathscr{S}}$	Turbulence model source term
S_H	Multigrid coarse grid solution
S_t	Vorticity at turbulence trip location
T	Static temperature
Tr	Mass averaged absolute total temperature ratio
U	Primitive variables
V	Volume
$\widehat{V}$	Viscous flux functionality
$\mathbf{W}$	Distance weighting matrix in *Least-Squares* derivative calculation
$\mathbf{X}$	Set of grid coordinates and mesh metrics
$\mathbf{X}_c$	Volume mesh coordinates
$\mathbf{X}_{surf}$	Surface mesh coordinates
Z	Viscous *Jacobian* matrix

$\mathbf{Z}$	*Hessian*-matrix eigenvector
Z^L	Linearised viscous *Jacobian* matrix
a	Generic variable
$\widehat{a}$	Convection velocity in $1D$ *linear convection equation*
b	Generic variable
$c_{b1,2}$	*Spalart-Allmaras* turbulence model constants
c_h	Multigrid fine grid correction
$c_{t1,2,3,4}$	*Spalart-Allmaras* turbulence model constants
c_{v1}	*Spalart-Allmaras* turbulence model constant
$c_{w1,2,3}$	*Spalart-Allmaras* turbulence model constants
$\overrightarrow{c}_i$	i^{th} cell centroid coordinate
d	Original edge length
$\widehat{d}$	Dissipative term contribution
$\widetilde{d}$	Perturbed dissipative term
d^*	Anisotropic edge length
d_t	Distance to turbulence trip location
d_w	Wall distance
f_j	j^{th} integral quantity of interest
$f_{t1,2}$	*Spalart-Allmaras* turbulence model variables
$f_{v1,2}$	*Spalart-Allmaras* turbulence model variables
f_w	*Spalart-Allmaras* turbulence model variable
g	Functional sensitivity to state variables
$\widehat{g}$	*Spalart-Allmaras* turbulence model variable
g_t	*Spalart-Allmaras* turbulence model variable
h	Fine mesh characteristic spacing
$\mathbf{h}$	*Riemannian* axis stretching in physical space
i	Index
j	Index
k	Index
$k^{(2,4)}$	*Jameson-Schmidt-Turkel* scheme coefficients
$\widehat{k}$	Thermal conductivity
n	Index

n°	Number of
$\overrightarrow{n}$	Surface normal vector
$\overrightarrow{n}_e$	Normalised edge vector
m	Index
$\dot{m}$	Area averaged outlet mass flow
p	Flow solver order of accuracy
$\widehat{p}$	*Partial differential equation* source term
q	Flow variable
r	Linearised equation residuals
$\widehat{r}$	*Spalart-Allmaras* turbulence model variable
$\overrightarrow{r}$	Surface curvature vector
Δs	Discrete control surface vector
t	Time
$\widehat{t}_i$	Set of code instructions
u	x-velocity component
$\overline{u}$	Constant velocity
$\widehat{u}$	Solution to linearised equations
$\widetilde{u}$	Perturbed convective term
v	y-velocity component
$\mathbf{v}$	*Hessian* matrix eigenvectors
$\widehat{v}$	Adjoint vector
w	z-velocity component
$\overrightarrow{x}_i$	i^{th} node coordinate
Δx	Edge length
Δx_t	Coordinate distance to trip
$z_{1,2}$	Example functions for *automatic differentiation*

Special Mathematical:

$\perp$	Elements orthogonal to the matrix *NULL* space
$\parallel$	Elements in the matrix *NULL* space
$\nabla\cdot$	Divergence
$\Re$	Real set of numbers

$\Im$	Imaginary set of numbers
δ_{ij}	*Kronecker* delta
$\oint$	Closed integral
$\mathcal{O}(.)$	Of the order of
$\forall$	For all

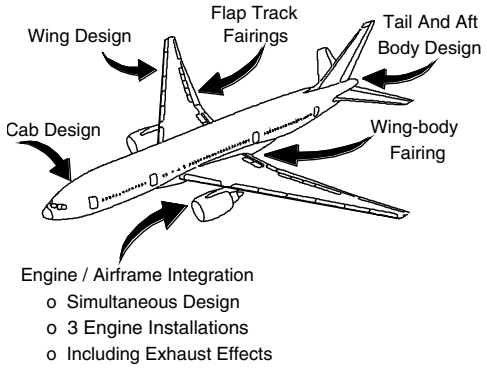
To my parents and my sister

Chapter 1

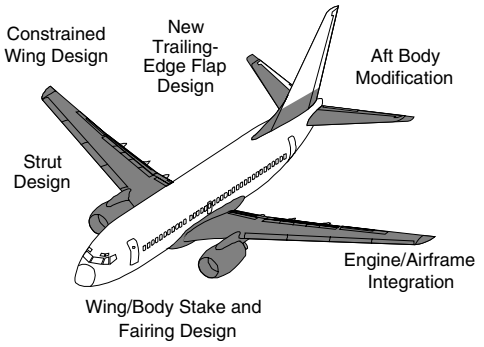
Introduction

1.1 Background

Over the last few decades, the use of *Computational Fluid Dynamics* (*CFD*) has steadily grown as improvements in computer hardware, numerical accuracy and general reliability have been accomplished. As a consequence, it is now employed on a daily basis in industrial settings, such as *Airbus*, *Boeing* and *Rolls-Royce* (*RR*), in the design process of a wide variety of components.



(A) *CFD* usage for *Boeing* 777 design.



(B) Predicted *CFD* use for future *Boeing* 737 design.

FIGURE 1.1: Examples of *CFD* usage at *Boeing* [1][1].

As highlighted by [2], the simulation of steady, transonic/supersonic flows may be considered a solved problem, reason for which *CFD* is consistently utilised for preliminary aerodynamic design. For example, [1, 2] report aircraft development cases where computer simulations have or are predicted to aid the design process (see Figs. 1.1 and 1.2). In particular, the *Boeing* 777 case (fig. 1.1a), *CFD* use was mainly related to the wing

[1]Reproduced with kind permission of *Elsevier*.

at high-speed, fuselage and engine/wing interfaces. Interestingly, the cab design was entirely carried out utilising computer simulations, and no further modifications were required after wind-tunnel experiments. This meant that experimental testing would be unnecessary for future cab design. Additionally, the authors state that the steady increase in *CFD* simulations has allowed a consistent reduction in wind-tunnel testing in an ever decreasing number of wing designs. These developments have therefore allowed significant costs and time savings.

(A) *Airbus A*380.

(B) *Boeing* 787.

FIGURE 1.2: Usage of *CFD* in commercial aircraft design [2][2].

Concerning cases more relevant to this work, [3] describe the current status of the usage of *CFD* on various components of a modern turbofan jet engine (figure 1.3). In particular, the authors report a 50% reduction in experimental testing of high-pressure compressors at *Pratt & Whitney* as a result of *CFD* simulations. Similarly, [4] shows how a *Reynolds Averaged Navier Stokes* (*RANS*) study favoured the improvement of

[2] 23rd *American Institute of Aeronautics and Astronautics (AIAA) Computational Fluid Dynamics Conference by AIAA*. Reproduced with kind permission of *AIAA* in the format republish in a thesis/dissertation.

fan and low pressure compressor efficiency by 0.6% and 1%, respectively, with a saving of $20M\$$ in rig testing.

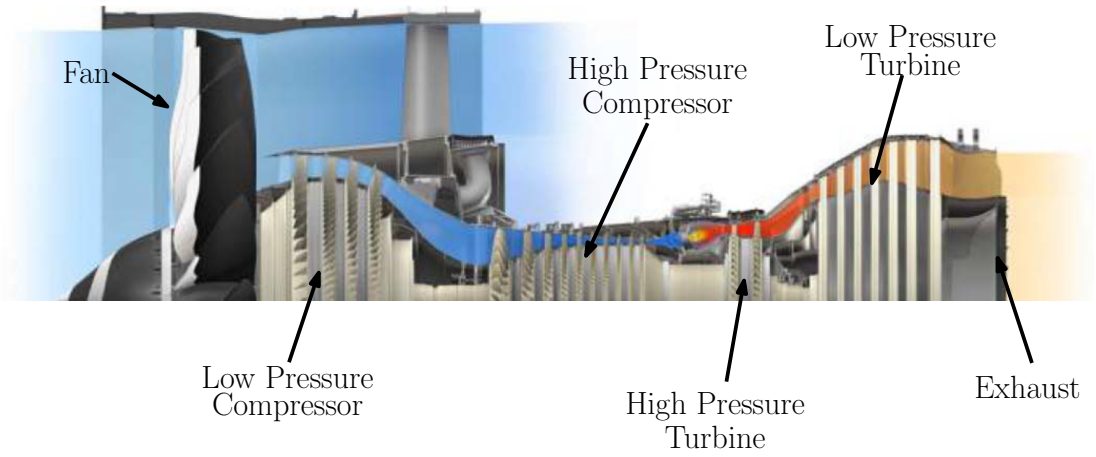


FIGURE 1.3: Components in a modern turbofan engine [3][3].

However, despite the impressive enhancements, *CFD* technology does still present many problems that require solving. According to [2], the main issues relates to the deficiency of computer simulations to deal with conditions different from steady attached flow. This is in agreement with [5]: in this case the authors also provide a detailed survey of the reasons behind this important limit to *CFD*:

1. Reduced funding for methodology development.
2. Computational hardware progress requiring novel software coding approaches to maximise benefits.
3. Lack of robustness outside the *RANS* realm.
4. Mesh generation, solution error estimation and grid adaptation.
5. Limited development of new algorithms to achieve higher-order accuracy results.
6. Simulation post-processing and data management.
7. Improved *CFD* solver reliability and automation for multi-disciplinary applications.

The 4^{th} item, i.e. error-estimation and mesh adaptation, concerns an unsolved problem even for the case of steady attached flows. In fact, many are the published examples of the successful application of a wide variety of approaches, but no evidence has been provided of consistent automatic improvement of the solution on complex geometries,

[3]Reproduced with kind permission of *AIAA*.

such as those used in industry [6].

Before considering grid adaptation techniques, it is necessary to produce evidence as to why they are needed. Such an example is provided by [7] and shown in fig. 1.4. Here the authors gathered the 3^{rd} *Drag-Prediction Workshop* drag coefficient (C_D) estimates from various codes using different turbulence models and meshes. As it can be seen, there is a significant scatter of the estimates: a $2.5 \cdot 10^{-3}$ variation in the predicted C_D can represent up to plus or minus 100 passengers on a civil aircraft [7]. Additionally, they highlight how the same *CFD* code employing the same turbulence model gave dissimilar results in mesh independence studies starting from different grids.

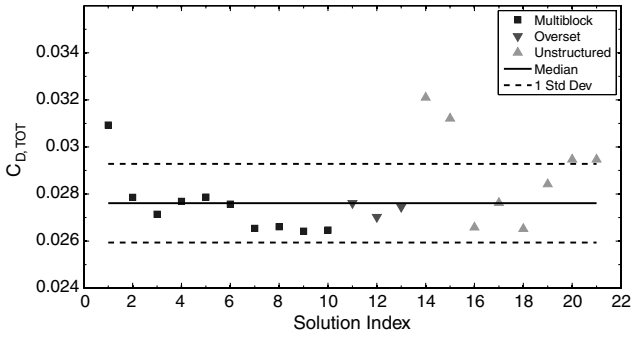


FIGURE 1.4: Scatter of the predicted drag coefficient (C_D) of the *DLR F*6 wing (*Mach* number 0.75, lift coefficient 0.5 and *Reynolds* number 5 million) with solution index representing different codes, turbulence models and grids using data from the 3rd *Drag-Prediction Workshop* [7][4].

More recently, as highlighted by [6], it has been shown that the ability to replicate all the physical features in a *CFD* solution has not yet been achieved. The authors give a twofold explanation as to why this is the case. Firstly, the grid generation, having always been an issue, has become even more intricate due to the greater complexity of the geometries being modelled today. Additionally to the increased mesh generation difficulty, this will also augment the burden on the flow solver, as many more smaller flow complexities will start to appear. The second reason for *CFD* shortcomings is related to the nonlinear nature of fluid flow and the model *Navier-Stokes* (*NS*) relations. For these reasons the authors argue that multiple mesh independence studies are required to achieve a reliable solution for a single case. In terms of time consumption, it should be noted that a complete refinement of a grid will increase the simulation time sixteen-fold. Therefore it can be concluded that the use of an automatic approach to modify a grid to reliably compute the flow solution would solve the issue. Moreover, as pointed out by [8], a successful mesh adaptation and error estimation technique would allow to reach asymptotic convergence of the flow solution, and thus reliability, on coarser grids. This, in turn, would have a positive outcome on a variety of aspects of a generic aerodynamic design process, such as reducing user/expert interaction, improved final design due to

[4]Reproduced with kind permission of *K. Fidkowski & D. Darmofal.*

more accurate solutions, decreased simulation time, increased widespread employment of *CFD* and so on.

1.2 Mesh Adaptation Process Overview

There are two main branches concerning mesh adaptation. These are divided according to the flow solution availability: *a-priori* techniques do not utilise any data other than the grid itself, while *a-posteriori* approaches will need an initial evaluation of the flow solution. The former techniques concern mainly the geometric characteristics of the mesh, such as cell aspect-ratio, skewness and size transition. Due to the fact that important regions of the flow are unknown, it is not possible to add, remove or move nodes strategically, reason for which these techniques have found limited use. On the other hand, *a-posteriori* procedures have shown consistent solution improvement with limited node addition, as they target the flow regions where most complex features appear. For this reason, a wide variety of these methodologies have been devised to improve the calculations and form the subject-core of this thesis[5].
The overall mesh adaptation process can be considered as a classical closed-loop feedback system as shown in fig. 1.5. In fact, once the flow solution is evaluated, an error estimation procedure follows. This is where the main diversity of the mesh adaptation technology resides, with a large number of possible strategies being currently available from published work. Once the error has been determined, it is employed to add, remove or move nodes in the domain, and the procedure is repeated until the conclusive calculation has converged to a fixed value.

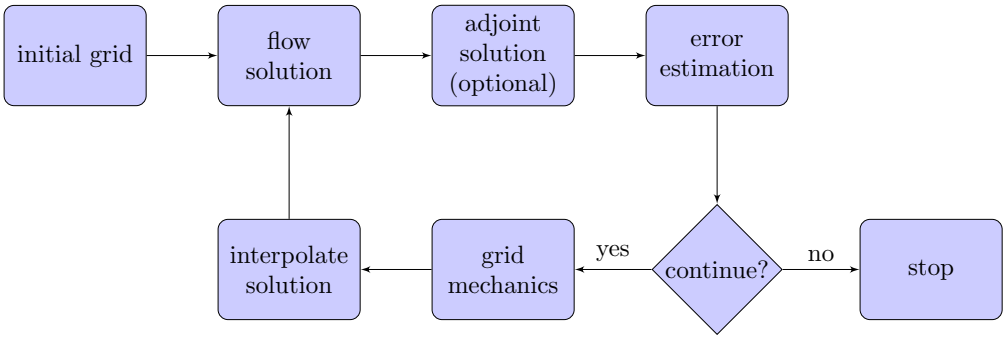


FIGURE 1.5: Generic grid adaptation process [8][6].

In general mesh adaptation can be seen as an automated control system to drive a car. The fact that it has to remain within the carriage delimiters and avoid hitting the opposing traffic on one side or the pedestrian sidewalk on the other, represent the stability

[5]From this point onwards whenever speaking about mesh adaptation it will be assumed it is relative to *a-posteriori* techniques unless specified.
[6]Reproduced with kind permission of *AIAA*.

limits of the flow solver. Repeated readings from the vehicle status, allow the control algorithm to limit the car movement within the boundaries equidistant from each side, with slight over- and under-shoots being corrected. Similarly, the iterative application of the grid modification technique should achieve flow accuracy convergence, with the solution oscillating within the limits of flow solver stability. Unfortunately, this last point represents the intricacy encountered when employing mesh adaptation techniques. In fact, often the initial flow solution is not in *asymptotic range*, i.e. it does not capture all the physical features present in the flow (regardless of their sharpness). Grid modification in one region may cause new, previously undetected complexities to appear. These may fall in portions of the mesh that are unsuitable for solver stability, therefore impeding any further analysis. Unfortunately, this is not the only issue, as more often than not, the error estimation procedures do not have a bound. Therefore, they may well continue to target the same regions of the flow despite these having achieved satisfactory resolution, causing over-refinement or even increasing the error. Additionally, the grid modification strategies, such as refinement or coarsening, can cause very aggressive oscillations that can result in poor convergence or even divergence.

Another important, generic aspect of *CFD* grid adaptation relates to the two most popular branches of error estimation techniques published. The oldest of these two methods is the *feature-based* approach. In this case, the data of interest relates to physical values of the nonlinear flow, such as density or static pressure, to capture shocks present in the solution. Ideally, the best parameter to apply is one that allows to identify all the flow intricacies, thus allowing node clustering to be located where they appear. On the other hand, the chosen physical variable should also be able to identify where less nodes are needed as the solution has a linear behaviour and thus be easily replicated using a coarser grid.

The other main mesh error estimation approach concerns *adjoint* technology. Originally, this technique was devised to optimise a geometry with an attenuated computational effort *with respect to* (*w.r.t.*) other approaches, such as the *Finite-Difference Method* (*FDM*). Once the nonlinear flow solution has been achieved, an adjoint solver is employed to determine the sensitivity of an integral quantity of interest (e.g. lift) *w.r.t.* multiple design parameters. As the adjoint relations solution for each functional is only carried out once, unlike other approaches, the reader can easily understand the advantage. However, an intermediate step in the process, requires the evaluation of sensitivity of the value of interest *w.r.t.* the flow residuals (this term is generally called *flow adjoint*). This quantity readily provides an indicator as to where solution imbalances are most influent on the functional. If one were to multiply by the actual flow residuals, a measure of the error of the integral value and the regions of the mesh causing it can be identified.

At this point, it should be highlighted, that feature-based approaches will attempt to

improve the overall flow solution, while adjoint techniques will only better the quantity they were evaluated for. Naively, it is easy to conclude that the former approaches should be pursed, as they are designed to modify the mesh to predict the overall flow solution more accurately, not just a single value. Significant research has shown, however, that adjoint approaches are more reliable and end up generating coarser grids (see for example [9, 10]). A more in-depth analysis of the overall technology will be provided in Chapter 2. A final mention should be made regarding the difference between *isotropic* and *anisotropic* approaches. Both words are derived the Greek language with the prefix *iso-* meaning identical and *ani-* referring to different. The suffix *-tropos* translates to direction. Therefore, in a mesh adaptation setting, the use of these words refers to the directionality of the error estimation and the grid modification technique. In fact, *isotropic* mesh adaptation will have the same error in all the three directions, while an *anisotropic* technique will be able to differentiate whether one direction has a greater error/requires more clustering *w.r.t.* others. Again, it is easy to understand that the latter will be more efficient, only refining/clustering nodes in the main direction, not in all.

1.3 Scope of the Work

The aim of this research is to devise *anisotropic* mesh adaptation methodologies related to adjoint techniques and apply them to steady turbomachinery cases. As previously mentioned, developing such approaches would allow a more efficient grid modification, resulting in a coarser grid, while still maintaining the same level of accuracy. Moreover, as *CFD* has reached a certain level of maturity, as discussed in previous sections, the application of *anisotropic*-adjoint technology and comparison with feature-based approaches may be applied to steady turbomachinery cases containing intricate geometry and flow features modelled with *RANS* equations. This represents a step forward in the implementation and development of mesh modification techniques as successful employment of such concepts would greatly benefit the application of everyday aerodynamic computer simulations and the reduction of industrial production design costs.

1.4 Outline of the Thesis

The document is divided into the following chapters:

- Chapter 2: **Mesh Adaptation Literature Review**
  In this part, a detailed analysis of the main adaptation techniques available in

present literature will be carried out. This will include feature and adjoint approaches, along with grid modification strategies that may be employed.

- Chapter 3: **Mesh Generation & Flow Solver Pre-Processing**
  An overview of the mesh generation process will be given, along with an insight into the steps required before running the flow solver. In particular, emphasis will be placed on the multigrid coarse grid generation.

- Chapter 4: **Flow Solver**
  An in-depth analysis of *Hydra* will carried out with detailed description of the flux reconstruction procedure, turbulence model, multigrid and time-stepping scheme.

- Chapter 5: **Adjoint Solver**
  As the central theme of the mesh adaptation technology employed is related to the adjoint, this chapter will provide the reader with the necessary information and proofs to be able to understand its main characteristics.

- Chapter 6: **Mesh Adaptation Tools**
  In this chapter the mesh adaptation software capabilities will be analysed. Moreover, detailed explanations of the various techniques implemented by the author along with the novel error estimation procedures developed will be provided.

- Chapter 7: **Combination of Feature and Adjoint Grid Adaptation**
  The first results chapter will focus on the integration of feature and adjoint-related approaches to be able to combine their strengths and provide a valid approach for grid modification.

- Chapter 8: **Fully Anisotropic & Edgewise Adjoint Grid Adaptation**
  The second set of results concerns the use of varied efficient adjoint strategies to be able to contain the cost of the approach but still maintain the relative advantages.

- Chapter 9: **Conclusions and Future Work**
  A summary of the research findings along with a discussion of the improvements required will be provided in the last chapter.

- Appendix A: **Error Equidistribution Principle**
  Detailed mathematical proof of the principle defining the optimal mesh.

- Appendix B: **Automatic Differentiation Example**
  Provides a simple but complete example of *Automatic Differentiation* related to the discussion in Chapter 5.

- Appendix C: **Feature-Based Mesh Adaptation Process applied to NASA Rotor** 37

This appendix contains the results of the feature-based mesh adaptation applied to the compressor case and used as a comparison.

- Appendix D: **Feature-Based Mesh Movement Applied to a Fan Blade**
  The final part of the thesis gives an overview of the application of feature-based mesh adaptation to the fan case of chapter 8.

Chapter 2

Mesh Adaptation Literature Review

2.1 Introduction

The birth of mesh adaptation as an individual research thematic dates back to the early *Eighties*, with the first conferences dedicated to this subject taking place in 1982 [11]. It consists of the automatic modification of a grid based on some sensor, to reduce the *discretisation error* (*de*), without the need to refine the mesh uniformly. In fact, as stated by [12, 13] this type of error is the most difficult to estimate and the main source of inaccuracy in a *CFD* solution, with others being *round-off*, *iterative* and *statistical sampling* errors. In a more general framework, [14] also includes *modelling inaccuracies*, *input uncertainties* and *post-processing errors* as causes of solution inexactness.
The *de* of a *partial differential equation* (*p.d.e.*) is defined as the difference between the continuous analytical solution and that of the discretised system [12]. The very same author is able to show how this quantity acts as a local source of error that for the *NS* relations, is then transported by the convection terms throughout the domain. To this end, two different test-cases to show this behaviour are illustrated: in subsonic flow, the *de* is propagated along the streamlines, while for hypersonic cases it is convected along the *Mach* (M) lines.
Apart from the automated accuracy improvement with minimal node count, optimal mesh adaptation can be beneficial in other aspects. As stated by [15], without mesh adaptation, the grid generation process is far too complex. In fact, if this is not employed in the overall *CFD* procedure, a significant amount of time has to be spent by the engineer to generate an appropriate mesh. Moreover, unless the user has a basic knowledge of the test-case aerodynamics and the flow solver's numerics performance,

the overall process can require iteratively modifying the grid and running the solver to achieve an appreciable level of convergence of the parameters of interest and resolution of the main flow features.
Despite the potential that mesh adaptation can produce in the overall computational aerodynamic design, both in terms of efficiency and accuracy, it presents a significant challenge, as despite 30 years of research, so far it has found limited use. It is not uncommon, for these procedures to cause more problems without improving (or even reducing) the solution accuracy [15]. As an example, the author reports how an oscillatory solution behaviour can trigger incorrect adaptation and thus increase error or cause divergence. Therefore, to date, despite the significant improvements of the techniques, they are still more of a research subject, rather than a means to automate and reduce time in computational aerodynamic design processes.
The overall mesh adaptation process can be split into three components [11]:

1. optimal mesh determination
2. error estimation
3. adaptation mechanics

Each of these will be separately analysed in the following sections.

2.2 Optimality Criteria

To appropriately modify a mesh, it is necessary to be able to determine when it reaches its optimal state. This means that the solution is sufficiently accurate no further change is required. In fact, the user will generally not know when this point has been reached, as the location and intensity of flow characteristics may only be guessed in either case, requiring a significant amount of experience.
In general, all techniques do employ the so-called *error-equidistribution* principle. This states that the optimal mesh is reached once the error sensor employed is equal over all elements in the mesh [16]. Here, the author also reviews the proof of this concept, which is outlined in more detail in appendix A. From a more intuitive point of view, [15] expresses it as a mapping from the physical space, where cell sizes may have any value, to a computational domain, where the grid elements have uniform size. During the mesh adaptation process, a threshold is then determined to *equidistribute* the inaccuracies in the grid, and once the statistical average error in the mesh has a relatively constant behaviour it is possible to terminate the procedure (see for e.g. [17] for a feature-based approach and [18] for an adjoint-related case). The statistical analysis is not always

employed, though. In fact, a final inaccuracy level may be determined starting with the application of the *equidistribution* principle ([19]), while in other cases a limit on the number of adaptation steps or final node count may be preferred.

2.3 Error Estimation

As discussed in the previous chapter, there are two main branches of mesh adaptation: *a-priori* and *a-posteriori*. In the first case, the flow solution is not used and therefore it is more complex to reduce inaccuracies in quantities of interest. Consequently, the main focus of this work will concern the latter techniques and their state-of-the-art will be discussed in detail.
A-posteriori error estimation methods, employ the current flow solution to determine the grid error and it is therefore assumed that the current solution is relatively close to the exact one. Effectively, [11], explains how this is only valid for *elliptic* or *parabolic* sets of relationships, as in *hyperbolic* cases, such as the *NS* equations, this assumption could cause incorrect refinement. Nevertheless, it is reasonable to consider the flow solution as sufficiently accurate to also determine an error for *hyperbolic* relations. The various techniques to determine the sources of inaccuracies in a grid, may be grouped into three categories [12]:

1. feature-based
2. adjont-based
3. truncation-error (te) based

While the feature and adjoint techniques will be discussed in detail as they were extensively employed in this work, truncation-error based procedures will only be mentioned in brief.

2.3.1 Feature Approaches

Possibly the most popular and simplest to implement, feature-related techniques attempt to highlight as sources of inaccuracies solution complexities. Therefore flow quantities, such as density or *Mach* number, are generally utilised as sensors. According to [11] common feature-based approaches are differences of variables, *Hessian* matrix of flow quantities or comparison of derivatives.
Effectively, though, the main issue regarding these techniques is the incapability of handling complexities of different magnitude [11, 15, 20]. In fact, there is no single flow

quantity that has a significant variation over all possible features appearing. For example, in a turbulent transonic flow the *Mach* number is able to capture the shock, wake and boundary layer, but is unable to detect any strong gradient at the wall. On the other hand, static pressure will highlight shocks and their propagation to the wall, but will not vary across any wake, separation and boundary layer present in the solution. To this end [11, 15] suggest using multiple flow parameters with non-dimensionalisation of each indicator. In particular [20], merges multiple flow quantities into a single mesh metric with a local rather than global approach, while [21] utilises the static pressure and *Mach* number to capture all flow quantities. A similar approach is taken by [22, 23], where shocks and wakes are extracted separately by employing a combination of velocity, density, *Mach* number and static pressure.

One of the main characteristics of feature-based approaches, relates to the amount of research that has been carried out to determine *anisotropic* error sensors. In fact, most flow features display directionality, in the sense that variations of quantities across them is far greater than in the tangential direction. Moreover, the ability to align the grid with solution complexities such as shocks, will allow to satisfy the *Rankine-Hugoniot* relationships, thus improving accuracy [21]. Being able to determine a directional error estimator will allow to achieve a higher density of nodes where they are actually needed, hence also improve the solution. On the other hand, *isotropic* approaches do not place any relative importance on the error directionality. This is not a disadvantage in terms of achieving an improved accuracy, but concerns the larger amount of nodes required to achieve the same solution *w.r.t. anisotropic* approaches. It is therefore mainly a question of efficiency of the resulting adapted grid, that has pushed the research community towards these approaches. As discussed by [24], the concept of *anisotropic* mesh adaptation appeared towards the end of the *Eighties* in [25]. In this case, grid generation was modified to be able to produce elements with different stretching in 2D. Three-dimensional versions were developed a few years later, in [26, 27], but according to [24], the visible *anisotropy* was minimal. A far more powerful approach to capture the flow features directions was developed by [28]. In this case, the authors argued that the absolute value of the *Hessian* matrix of a scalar quantity of the flow could be adopted to map the edge length to a *Riemannian* metric. Obviously, the *Hessian* matrix represents the error in the linear interpolation over an element [29]. Since this ground-breaking article, this technique has been the source of significant research, and is nowadays considered the most promising feature-based adaptation technique. This process was later on employed by [30] in a combined movement and refinement development. The fact that this error estimator can be deployed to move the grid nodes and generate an *anisotropic* grid, makes it suitable in a structured data setting, as no connectivity between the elements is changed. Therefore, [29] made use of it to determine a new edge length that could be fed into a spring-stiffness based mesh movement procedure, thus allowing *equidistribution.*

One of the 2D cases analysed is shown in figure 2.1. As it can be seen from the grid on the lower-left-hand-side (*LHS*) image, the starting mesh has been significantly stretched and clustered towards the main flow features shown on the right-hand-side (*RHS*).

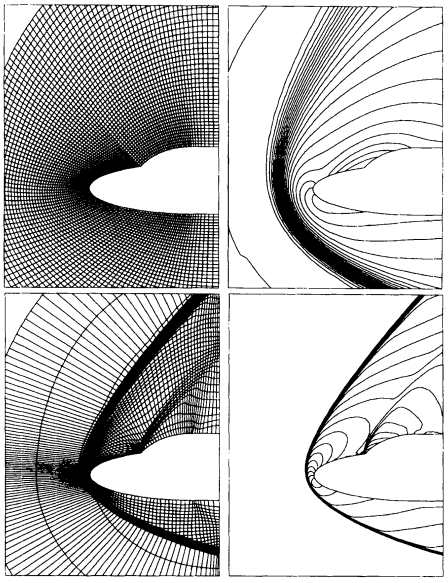
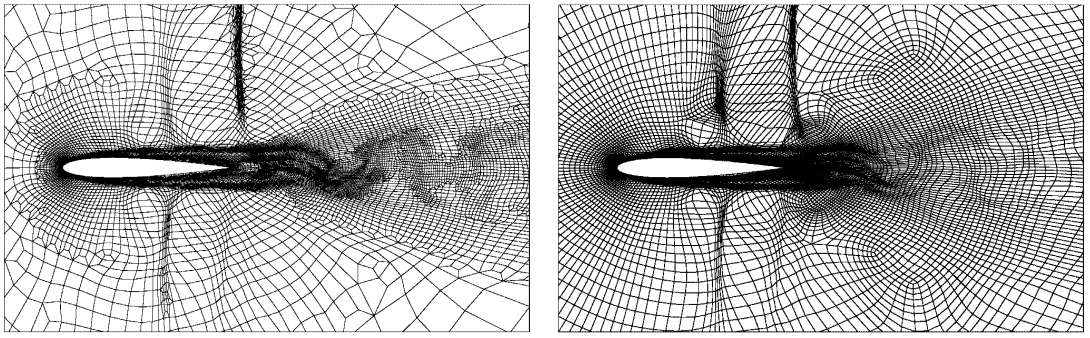


FIGURE 2.1: *Riemannian*-metric based mesh movement around a double ellipse at hypersonic flow conditions (*LHS* mesh, *RHS* are *Mach* number contours) [29][1].

As previously mentioned, flow quantities are difficult to utilise on their own to adapt the mesh successfully. Therefore [20] not only combined the metrics of multiple variables, but also modified these at the wall and in its vicinity to have good orthogonality and allow the user to select a certain wall distance. The finalised error was then used in an edge swap, collapse, split and movement process. The same strategy in terms of grid mechanics, but without any artificial modification of the *Hessian* matrix or derived metrics, was employed by [17, 31–34]. In their work, they iteratively applied the mesh adaptation process based on the *Riemannian*-metric and were able to show how starting from different grids, the same final solution could be achieved. Moreover, the final adapted grid produced the same answer when utilised by different flow solvers. However, they did note that throughout the adaptation process, the solver settings had to be changed to include higher artificial damping to allow satisfactory convergence on the poor initial grids that were created. This requirement was then unnecessary on the resulting grid. In particular, in [17], they show the initial and final error distribution over the mesh. It is clear from the curves that this has been appropriately *equidistributed* over the domain. Later, in [32], they actually employed the strategy they had devised for 3D flows, yet were not able to show the same sort of improvement as for the 2D cases they had analysed till then. One of the possible reasons behind the reduced performance is the treatment of turbulent regions near the wall. In fact, as is generally the case,

[1]Reproduced with kind permission of *John Wiley & Sons*.

wall functions are used and require a suitable grid spacing in order to appropriately capture the entire velocity profile. This is generally not know during the grid generation process, and therefore should be taken into account in the adaptation process. This was the central aspect of the work in [35]. Here the authors propose to modify the near-wall metric derived from the *Hessian* matrix in two different ways to accommodate cases where *wall functions* are employed and cases where they are not. To this end they devise two separate strategies. In the cases where the near wall velocity profile is modelled, they force the near wall layers to have a spacing equal to that specified by the user. In regions such as stagnation points, the y^+ requirement is neglected as it would be too small, and the actual error metric derived is employed instead. For cases where the *wall functions* are not used, e.g. fully tetrahedral grids, the wall metric is modified to have the largest eigenvalue perpendicular to the surface, while the other two are aligned tangent to the wall itself. Another approach employing the *Riemannian*-metric and applying special modifications to the grid in the near wall region is discussed in [36, 37]. A different *Riemannian*-metric based approach to the refinement and movement was proposed by [38, 39]. In fact, nodes are added while managing to keep the grid fully quadrilateral (in 2D or hexahedral in 3D) without any *hanging nodes*, and deploying the mesh movement to smooth out the overall grid. They employed the so-called *pillowing* approach as described by [40] alongside the ideas of [41]. Examples of their strategy applied to an unstructured and structured multi-block fully quadrilateral grid are shown in figures 2.2a and 2.2b, respectively.



(A) Unstructured quadrilateral mesh. (B) Structured multi-block mesh.

FIGURE 2.2: Adapted grids for an unsteady viscous flow around *NACA* 0012 [39][2].

Given the successful results achieved employing *Hessian*-related metrics to adapt the grids, mesh regeneration techniques based on this quantity started to be favoured. It should be clear that two different approaches may be undertaken. The first flags domain features and attempts to build a fully quadrilateral/hexahedral block around it, essentially treating it as a geometrical entity. The remainder of the computational field may then be filled up with a combination of hybrid elements. Another approach which

[2]Reproduced with kind permission of *John Wiley & Sons*.

appears to have gained more popularity, is the complete grid regeneration employing the *Hessian*-metric to guide the location, size and orientation of tetrahedral elements. The former technique, assures an improved feature resolution as the quadrilateral/hexahedral elements allow perfect alignment of the grid with flow complexities (in the case of shocks, as previously discussed, this allows to fulfil the *Rankine-Hugoniot* conditions). Additionally, these elements are more efficient *w.r.t.* their *simplicial* counterparts, as with a fixed number of nodes, the quantity of edges, faces and volumes are less. On the other hand though, the triangle/tetrahedral flexibility consents the filling of a generic complex domain in the easiest manner. For this reason these latter approaches have been pursued further.

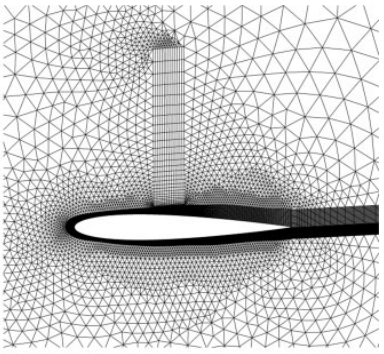


FIGURE 2.3: Example of quadrilateral block generation around boundary layer, wake and shock of *NACA* 0012 [21][3].

Examples of mesh generation employing quadrilateral blocks where the detected flow features are can be found in [21, 42], whist a 3D extension of the work is reported in [23]. One of the test-cases considered in [21] is shown in figure 2.3. As it can clearly be seen, the grid has been regenerated with good quality quadrilateral blocks around the aerofoil, wake and shock, while the remnant of the domain has been filled with relatively smooth triangular elements.

Concerning the complete 3D mesh regeneration with tetrahedral elements, much work has been carried out at the *Institut National de Recherche en Informatique et Automatique* (*INRIA*, France), where significant developments to the methodology have been researched. Following from the metric definition employing the *Hessian* matrix, they managed to determine an upper bound to the interpolation error that is independent of the problem at hand [43]. Moreover, they have related the *anisotropic* measure to the shape, size and orientation of a tetrahedral element. However, the error field is discrete, in the sense that it is only available at the points where the solution has been evaluated by the flow solver. Given that regenerating the mesh to cluster points according to the metric will place them in different locations, a continuous version of the field is required.

[3]Reproduced with kind permission of *John Wiley & Sons*.

To this end [44] employs interpolation techniques to achieve this continuous representation, while in other publications, [45–47], rigorous mathematical derivations along with applications are provided. Within the mesh regeneration process, an optimisation step has been included to be able to produce a grid that minimises the linear interpolation error for a given amount of nodes [48].
One of the main characteristics of the *Hessian* matrix computation is the difficulty in being able to achieve a smooth and accurate field. This is in part due to the test-case complexity, along with the starting mesh resolution and the accuracy of the flow solver[4]. A comparison of various approaches to carry out the *Hessian*-matrix determination is reported in [49, 50]. Nevertheless, it is often necessary to employ artificial strategies to be able to smooth the derivatives or the metric itself. Examples of various techniques are provided by [51–54].
Given the test-cases treated in this work, a final mention should be made concerning 3D feature-based mesh adaptation applied to turbomachinery. The first tests were carried out by [32], who adapted a hybrid prismatic-tetrahedral mesh for a viscous *NASA Rotor* 37 simulation, run with the $k-\omega$ turbulence model. They employed the relative *Mach* number as their sensor to form the *Hessian* matrix. A combination of movement, edge collapse, swap and split was used to adapt the grid. Despite improving the clustering towards the main features of the flow, the results were not completely satisfactory. More recently, [48] considered the *LS*89 turbine blade in 2D and *NASA Rotor* 37 in 3D and regenerated the mesh for both cases. In particular, in the latter case, the authors employed prismatic elements in the boundary layer, that were left untouched by the adaptation. The rest of the domain was filled up with tetrahedral cells, that were actually regenerated employing a *Riemannian*-metric. Of note, is also the fact that they did not modify the periodic boundary of the starting mesh, that had a node count of 2.8M. The resulting adapted meshes show how the main flow features are captured, however, there is no indication of how the final grid performs in terms of quantities of interest (such as efficiency or pressure ratio).

2.3.2 Adjoint Approaches

Adjoint error estimation and grid adaptation techniques appeared towards mid-90s (see for e.g. [55, 56]). They attracted particular interest, as unlike feature based approaches, they allow the definition of an error threshold [57] and are more reliable in determining an accurate quantity of interest. In fact, feature-related techniques tend to cluster nodes in parts of the flow such as shocks, shear/boundary layers and wakes, not taking into account the error that may originate in other regions of the flow [58]. Moreover,

[4]Most *FVM* codes are $\mathcal{O}(2)$ accurate, therefore the $\mathcal{O}(2)$ derivatives are on the limit of the flow solver's capability.

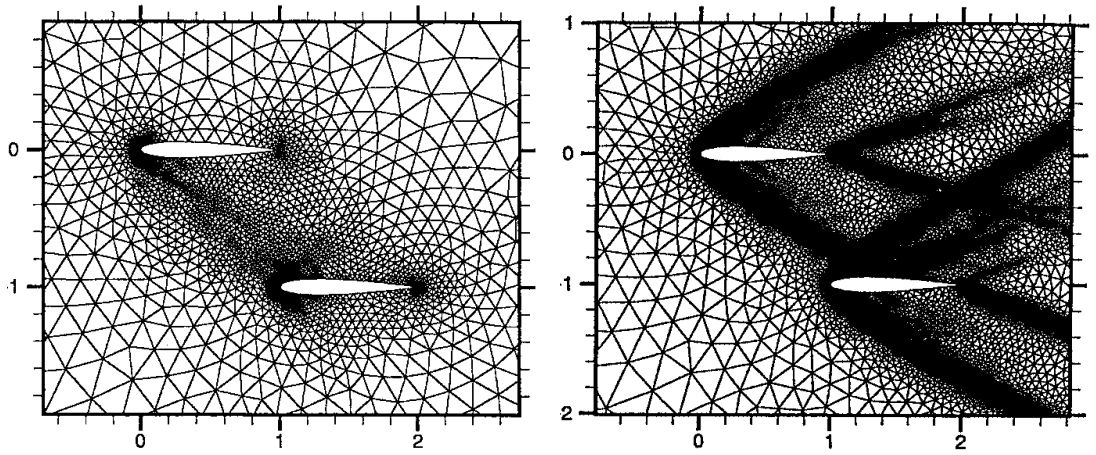
the very same author, argued that even *te* procedures do not consider the relative local effect that this may have on the global quantity. By including the adjoint weighting into the error analysis, [58] showed that it is possible to relate the *te* in the solution to that of the quantity of interest. In particular, the *superconvergent* property was proven for *Finite Element Techniques* (*FEM*), by which the functional has double the rate of convergence towards the exact solution *w.r.t.* the flow quantities. At a later stage, [59] achieved *superconvergence* of the performance parameter for all types of discretisation: *FDM*, *FVM* and *FEM*, for both linear and nonlinear problems.

The same approach of weighting the flow residuals vector by the adjoint solution was later employed by [60]. This was applied to a 2D subsonic/transonic flow over *NACA* 0012 profile. In particular, they attempted to smooth out the flow residuals due to *checkerboard* behaviour they had, by introducing a control volume gather and scatter approach. Unfortunately, there appeared to be issues in deciding the amount of smoothing iterations that may be applied, as too many caused solution degradation. Nevertheless, they did show better performance of the adjoint weighted methods *w.r.t.* feature and residual-only sensors, as these were incapable of detecting all sources of inaccuracies in the domain.

In the meantime, in a series of articles [9, 19, 57, 61], the most popular adjoint-based adaptation process to date was presented and applied to *FVM* solvers. Their work was based on that of [59], with the main difference being that the error on a grid with average spacing H was evaluated on an embedded grid with spacing $\frac{H}{2}$. They determined two types of error: a *computable correction* and a *non-computable correction*[5]. Both of them were evaluated on the embedded grid, with the former simply being the flow residuals vector multiplied by the adjoint. The *non-computable correction* term is an estimation of the remaining error due to the prolongation of the adjoint solution, instead of using the exact quantity. They argued that, as the *computable* version could be reliably calculated if the grid is in *asymptotic range* and used to improve the functional estimate, the *non-computable* version should be employed for the actual cell-based refinement. It is important to note that their *non-computable* sensor, is composed of the average of both primal and dual computations of the remaining error, and therefore can be used to reduce the *duality gap*, thus decreasing nonlinearities present in the flow. One of the requirements to evaluate the error is the need to interpolate both flow and adjoint solutions linearly and quadratically. In [9] they also proved significant benefits *w.r.t.* *Hessian*-metric-based procedures for 2D inviscid flows. An example of their results is shown in figure 2.4. In an attempt to improve the drag evaluated on the lower aerofoil of the configuration, it was proved how a pressure-sensor (2.4b) significantly over-refines the grid trying to sufficiently resolve all the flow features, thus wasting valuable resources. On the other hand adjoint-based processes (2.4a) only flagged regions where

[5]For an in-depth analysis and derivation of these quantities the reader is referred to section 6.2.4.

the functional was affected by errors. In particular they highlighted how the leading edge (LE) of the two aerofoils was refined in a different manner by the adjoint error estimate, but identically by the feature-based approach.



(A) Adjoint-based mesh adaptation. (B) *Hessian* of pressure mesh adaptation.

FIGURE 2.4: *NACA* 0012 tandem aerofoil configuration (inviscid flow conditions) mesh adaptation to improve drag estimate on lower component as shown in [9][6].

In their final article concerning adjoint-adaptation ([19]), $2D$ viscous flow conditions were considered. In their previous work, the mesh adaptation had been carried out by regenerating fully triangular grids by determining each element's size using the *non-computable* term. In this case, they included *anisotropy* into the grid re-generation procedure. This was done utilising the standard *Hessian*-based *Riemannian*-metric calculation (see [20, 24, 25, 33] for examples of this error computation). In their case, they employed the *Mach* number as the flow quantity of interest, to be able to determine the stretching and orientation of the mesh cells. On the other hand, the element size was computed by *equidistributing* the remaining error in the adjoint adaptation formulation. One of the test-cases they employed was the *RAE* 2822 in turbulent flow. They were able to show the improvement of the devised methodology *w.r.t.* the pure *Hessian*-based mesh re-generation. In particular, it was consistently proven that the quantity of interest is better approximated with fewer nodes when taking the adjoint error into account. Figure 2.5, shows one of the test-cases analysed by the authors. The adjoint-related processes were better at reducing the inaccuracies in the inviscid part of the flow, also improving the wake resolution. It should be noted, that the latter clustering is probably due to the combination of the *Hessian* with the flow solution residuals present in the remaining error. In fact, adjoint approaches do not tend to highlight downstream regions as those requiring refinement. On the other hand, the lower image shows how the pure *Hessian* technique tends to over-cluster the regions near the geometry, thus neglecting any inaccuracy due to the rest of the domain.

[6]Reproduced with kind permission of *Elsevier*.

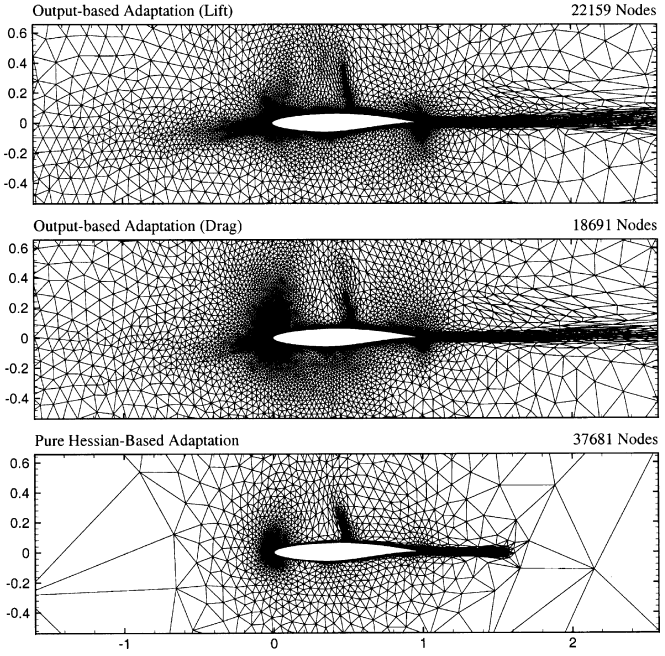


FIGURE 2.5: *RAE* 2822 resulting adapted meshes for lift and drag *anisotropic* adjoint error procedure (top and middle) and using pure *Hessian*-based technique (lower image) from [19][7].

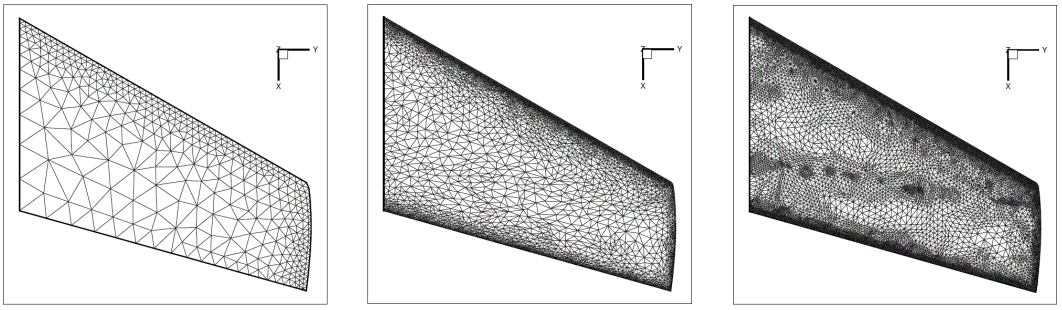
A similar approach was taken by [62, 63]. They employed a method to include the adjoint error estimate as derived by [64] into a *Riemannian metric* formulated through the *Hessian* for *FEM* solvers. They also managed to combine the metrics of different quantities.

The first successful attempts in adapting the mesh based on the adjoint-error in 3D are those of [65] for inviscid and [66] for viscous simulations. In [65], the author paid particular attention to the interpolation scheme employed to transfer the flow and adjoint solutions from coarse to embedded grids. In fact, while the linear prolongation can be carried out by simply averaging values along edges, faces or cells, the quadratic is more involved. [19] had originally proposed *local-least squares* process with error minimisation on the coarse mesh, thus achieving a discontinuous quadratic reconstruction. The embedded node values were then determined by averaging the coarse grid parameters of the points to which they were connected to. [65], instead proposed using a *cubic-fit* along each edge employing the gradients determined with the *Least-Squares* (*LSQ*) approach (see [67] on how the gradient is calculated). However, the main finding concerned the effect that using the linear or high-order interpolation had on the *computable* error term. In the test-case considered (*ONERA M*6), it was seen how using the lower-accuracy interpolation for the correction factor would improve the functional estimate, but not as much as using the quadratic operator.

In [66], *anisotropy* was merged with the adjoint error analysis as in [19]. This was done

[7]Reproduced with kind permission of *Elsevier*.

by employing the *Mach* number *Hessian* and scaling the maximum eigenvalue of each node's second derivatives matrix by the adjoint sensor. Results for the *ONERA M6* inviscid transonic case showed that this approach tends to converge to a more accurate result sooner and with less nodes *w.r.t.* the procedure formulated in [65]. Comparison of the starting and adapted meshes achieved employing the two adjoint error techniques to better the wing's drag estimate, are reproduced in figure 2.6. These confirm the improvement obtained by combining the adjoint with the feature-based adaptation technique: it is clear that figure 2.6b is coarser than figure 2.6c. It is of interest to note, however, that the inclusion of *anisotropy* has not caused any visible refinement where the shock is, unlike the *isotropic* approach, where part of the shock structure is clearly visible. The author suggested that the cause of this may be related to the shock location being more important than its resolution to appropriately determine the functional of interest.



(A) Starting grid. (B) Anisotropic adaptation. (C) Isotropic adaptation.

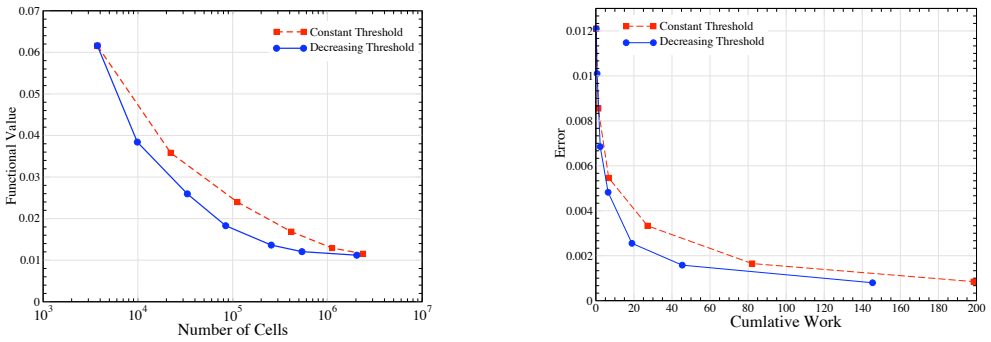
FIGURE 2.6: *ONERA M6* transonic inviscid flow mesh adaptation as shown in [66][8].

Concerning the viscous case analysed, this consisted of the extruded *NACA* 0012 with the functional of interest being drag. In this case as well, the author was able to show how the adjoint error combined with the *anisotropic* approach successfully reduced the error in the performance quantity. However, it was observed that the *computable correction* evaluated on the fine grid was not as accurate as in the inviscid cases due to oscillations in the interpolation that caused non-physical negative turbulent viscosity to appear.

A detailed study on the effect of interpolation techniques relative to the adjoint solution for the *computable* and *non-computable* errors, was carried out in [68]. In their simulation they employed an *Embedded Boundary Cartesian* method. They applied two sets of interpolation techniques for the adjoint solution from coarse to fine grids: a piecewise constant and a higher order reconstruction involving the gradients calculated using a *LSQ* solver with a *Barth-Jespersen* limiter ([69]); the second procedure made use of *trilinear* and *triquadratic* polynomials. Their results showed that using the better interpolation approach (*tricubic*) improved the error correction estimate, but it was

[8]Reproduced with kind permission of *AIAA*.

sufficient to use the linear interpolation to determine the adaptation parameter. Additionally, they also compared the error estimate using the adjoint solution solved on the fine grid. From the data they concluded that the error correction is better estimated this way, but the *tricubic* interpolation is close in terms of performance. Of note, is also the fact that their refinement algorithm was cell-based and that they started the adaptation process by adding a small amount of nodes progressively increasing the quantity. They argued, that at the start the error will not be as accurate, therefore limiting the node addition will avoid wasting resources. To this end, the very same authors in [18] employed the varying threshold philosophy in an attempt to devise a robust and generic strategy. Utilising a similar philosophy to [70], they were able to show, for an inviscid transonic *ONERA M6* simulation, that employing a varying threshold achieved the same accuracy of the final grid with 15% less nodes *w.r.t.* a constant threshold approach. In fact, their technique attempted to *equidistribute* the error over the entire mesh by attacking the highest sources of it in the first few adaptation steps. By gradually reducing the value, more and more low error cells were split. This procedure also allowed to have a relatively limited refinement in the beginning and only in the last few adaptation steps would the node count substantially increase. An example of the validity of their results is reproduced in figure 2.7a. It is clear how the changing threshold procedure always has a lower node count *w.r.t* the constant valued procedure, still reaching the same error level. This then results in a reduced run time as shown in figure 2.7b.



(A) Node count against functional per adaptation step .

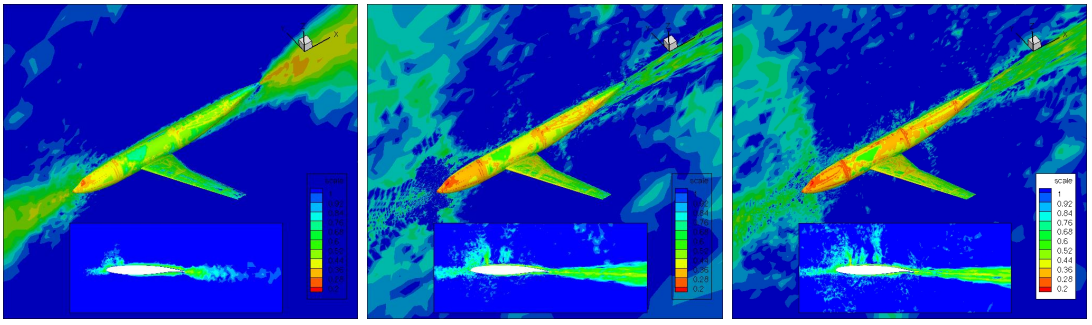
(B) Wall clock time normalised by w-cycle multigrid against functional error.

FIGURE 2.7: Effect of using a varying threshold for adjoint error adaptation applied to *ONERA M6* [18][9].

As the complexity of the cases grew, there was the necessity to parallelise the adaptation process. In fact, the need for an embedded mesh along with linear and quadratically prolonged flow and adjoint solutions, requires significant computational memory capabilities. To this end, [71] were the first to employ a parallel mesh adaptation code. In particular, it is interesting to note that they modified the *non-computable correction*

[9]Reproduced with kind permission of *AIAA*.

term to include the quadratically interpolated flow and adjoint solution residuals. According to the authors, this allowed a much more accurate error map, although at the expense of the technique's robustness. It is interesting to note that, as they did not have *flux limiters* available in the adjoint solver, the adaptation process had to be halted due to poor convergence performance. Nevertheless, they applied the adjoint-procedure to the *DLR F*6 case in turbulent flow. The drag adjoint error maps evaluated on starting and two adapted grids are shown in figure 2.8. As it can clearly be seen, the sensor map is very noisy, but does in general present a combination of both, flow and adjoint features.



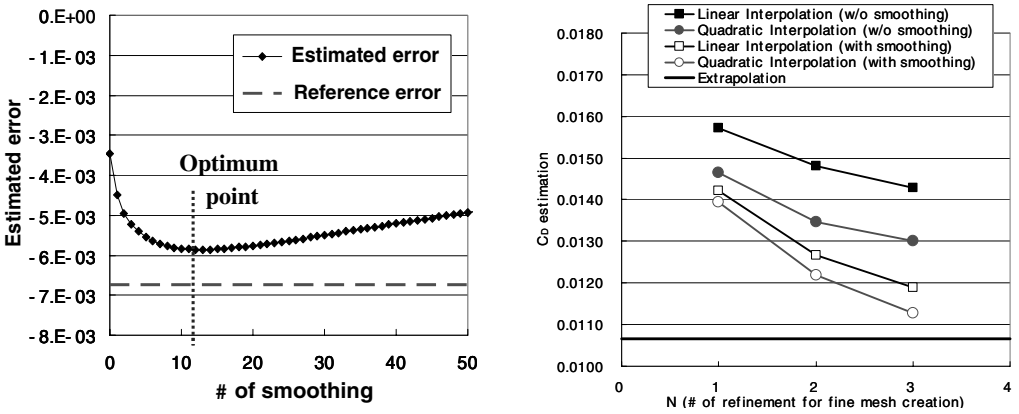
(A) Hand generated medium mesh of 3M nodes. (B) Adapted medium mesh of 3.7M nodes. (C) Adapted fine mesh of 9.4M nodes.

FIGURE 2.8: Symmetry plane and surface mesh adjoint error on the *DLR F*-6 configuration along with a mid-span grid cut as shown in [71][10].

Investigations into methods to reduce the noisiness of the sensor map were carried out by [72]. In particular, the cases of interest were the *ONERA M*5 and *M*6, simulated with an *Euler* solver. Concerning the adaptation procedure chosen, they employed the *computable error* correction due to the extra burned that computing the fine grid adjoint residuals would cause in terms of memory consumption. This was combined with its standard deviation to determine an error threshold. The interpolation parameters from coarse to fine grid were those employed by [65]. The actual refinement was carried out by marking nodes with high error on the fine mesh and consequently any cell on the coarse mesh containing them. There were two interesting aspects reported by their research. Firstly, they employed the volume weighted gather and scatter approach used by [60] to smooth out the flow residuals, as high frequency noise due to the interpolation were present. This allowed them to improve the *computable correction* estimate. Their main result is reproduced in fig. 2.9a, where it can be seen that there is an optimal number of residual smoothing iterations: too few and the high frequency interpolation will dominate the *computable error*, too many and the smoothing will dampen it. Additionally, they also showed that using multiple refinement steps to generate the grid onto which

[10]Reproduced with kind permission of *AIAA*.

to interpolate and evaluate the error helps achieve a more accurate solution. The result confirming this is shown in fig. 2.9b, where the error calculated using linear or quadratic prolongation, with or without smoothing is always improved by using multiple layers of refinement.



(A) *Computable error correction* against residual smoothing iterations.

(B) *Computable error correction* against levels of refinement.

FIGURE 2.9: Effect of smoothing and multiple refinement levels on the *computable error correction* [72][11].

Despite their very promising findings, the same authors were unable to show the same improvement for viscous flow computations [73]. The performance quantity of interest for their study was drag for the *ONERA M*6 case, simulated with the *Spalart-Allmaras* (*SA*) turbulence model. The main finding of this work concerned the use of the residuals smoothing approach. In fact, they stated that even one iteration of it will significantly dampen the error map and remove significant features that are necessary for appropriate adaptation.

Adjoint mesh adaptation procedures related to the technique developed by [64], require the starting grid to be in *asymptotic range* of the exact solution. This requirement is necessary due to the interpolation to the embedded mesh (just as for *Richardson extrapolation* process [16]). While this requirement is true in a strict sense, [74] was able to show that the process will work even when the starting mesh is too coarse. In particular, they employed the strategy outlined by [64] to adapt a 3D *Euler* flow around various conical configurations to accurately predict the pressure signal at a certain location in the domain. Despite starting from a grid with limited node count, the authors were able to successfully adapt it. What is interesting, is that they always found that the first few adaptation steps would have relatively strong oscillations of the error. The reason behind this, they claim, relates to the starting mesh not being in *asymptotic range w.r.t.* the correct solution, and therefore the calculated error will not decrease *monotonically* as in [9, 19, 61].

[11] Reproduced with kind permission of *H. Kim*.

A detailed comparison of feature and the adjoint-based error estimation (as developed by [64]) in 3D inviscid/viscous flows can be found in [10]. The main change *w.r.t.* the approach of [64] was the calculation of the interpolated solution to the fine grid: this was done using a *Moving Least Squares Approach* (for e.g. see [75]). Additionally, they did not employ mesh-regeneration but *isotropic* refinement, as each primal volume was marked to be split if the average of its nodal adjoint error values was greater than the threshold. Their work showed how the adjoint adaptation procedure consistently outperformed that using the feature related approach. Moreover, they also discussed how the latter is much more sensitive to the starting grid *w.r.t.* the adjoint-related technique.

Similarly, [76] analysed inviscid flow around the *NACA* 0012, *ONERA M*6 and *DLR F*6 wing-nacelle-pylon configuration with the *DLR* flow solver *TAU*, and reached the same conclusions. In particular, they showed how the technique proposed by [57] with the interpolation suggested by [65] outperformed feature-based approaches, in terms of accuracy and final node count. As in [10], they too concluded that adjoint techniques are far superior if the starting mesh is very coarse. Additionally, they compare the *computable* and *non-computable* errors effectiveness for the adaptation. In particular, they removed the adjoint residuals multiplied by the difference of interpolated flow quantities in the *non-computable correction* evaluation. From their results they concluded that the *non-computable correction* is the best option for the grid refinement, as it consistently created coarser meshes to match the desired accuracy.

The burden of time and memory usage of adjoint adaptation techniques described by [9] was considered by [77]. In their work they devised two strategies to be able to overcome this issue. The first consisted in what they call *adaptation sub-iterations*. These were employed within the overall mesh improvement process by alternating them with the standard and more accurate technique. The error for the sub-iterations was evaluated using the approximate flow and adjoint solution of the adapted mesh. As usual, these are then interpolated to an embedded grid, where the adjoint quantities are smoothed using a *block-Jacobi* process. The sensor formulation they considered was that consisting of the flow residuals multiplied by the difference of low and high order prolonged dual solutions. While the smoothed adjoint formed the lower accuracy term, the other is constructed in a different manner. In fact, for every complete error estimation step as per [9], they actually solved the embedded grid adjoint to remove the remaining error. This expensive solution, is then recycled and interpolated to the following adaptation step's embedded grid, to be used as the high order term. This cheaper technique was successfully applied to a 2D and 3D extruded *NACA* 0012 case, simulated using a *Discontinuous Galerkin* (*DG*, see [78] for an overview of the methodology) steady solver. In all cases they were able to show the gain in terms of time without any final loss in accuracy.

The second method they proposed to reduce time and memory consumption consisted in modifying the process of [64] by interpolating from the starting grid to a coarser one. For their work they utilised a lower order interpolation state rather than coarsening the physical mesh. The test-case used was an unsteady 2D advection *Gaussian* wave with inflow on the lower and *LHS* of a quadrilateral domain. This was simulated using the *Active-Flux* methodology ([79]) and showed positive results.

A separate mention is required for cases where the base strategy of [9] was modified. [80] combined the use of adjoint error techniques with a *defect error correction* approach. While the former employed flow and adjoint solutions, duly interpolated to a fine grid, to increase the functional accuracy by one order of magnitude, the latter would attempt to improve the overall flow solution. To be able to employ the *defect correction* approach, a reconstructed flow solution was required (in this case they employed *cubic splines*), that was then fed into an iterative process to determine the base solution's accuracy or improve the functional's estimate. However, coupling the two techniques allowed to achieve a higher order of accuracy *w.r.t.* the reconstruction procedure. In particular they proved this concept for linear and nonlinear *quasi*-1D and 2D inviscid problems.
[81], studied the effects of using a *continuous* or *discrete* formulation of the adjoint solver in the grid error calculation proposed by [57]. They showed that the presence of shocks is negligible on the mesh adaptation process, for both *continuous* and *discrete* adjoint formulations. Concerning the performance of the two different dual solver formulations, they concluded that the *discrete* approach was possibly the best option, as unless the grid was suitably fine, the *continuous* counterpart would under-perform.
[82] attempted to overcome the issue of having different errors occurring when perturbing the input, thus allowing to produce insensitive grids. This is particularly important for *uncertainty quantification* purposes, as these require varying the system parameters to be able to carry out a statistical analysis. Their procedure consisted in modifying the techniques of [9], by introducing unknown perturbations. Therefore minimising the error equations, would also allow to reduce the discretisation error variability. Again the refinement process was cell-based.
Finally, [83] considered the effect of constraints in the error reduction of a functional of interests, while [84] managed to combine the errors of multiple functionals and [85] employed adjoint error estimation techniques with the $k - \omega$ turbulence model.

Till this point, methods and applications related to the work of [9, 19, 57, 59] have been outlined. In fact, their techniques have proven time and time again to consistently produce accurate results with minimal node addition. Nevertheless, as pointed out by the literature, it is an expensive process due its need of an embedded grid, two interpolation

operators, two flow and adjoint solutions with their relative residuals. A few attempts have been made to try and reduce the burden of it, as previously described, however, other researches devised completely different adjoint-based error estimators.

One of the most original techniques is that developed by [86, 87]. In this case, the error indicator was based on the idea that *artificial dissipation* can be the cause of up to 90% of the error in a functional of interest [87]. The analysis was based on the *Jameson-Schmidt-Turkel* (*JST*, [88]) flux reconstruction scheme, where the $k^{(2)}$ and $k^{(4)}$ coefficients dampen the higher-order terms where the *shock switch* detects a flow complexity. In fact, where strong variation of quantities or discontinuities appear, the inclusion of high-order terms causes oscillations that can eventually lead to instability and divergence of the solution. Apart from the error reduction, this procedure presented other advantages, such as avoiding the generation of a fine grid and the consequent interpolation onto it. However, not all flow solvers have a *JST* flux discretisation, and actually there are more accurate schemes *w.r.t.* this, questioning the main idea behind the error formulation [87].

To avoid the necessity to run the adjoint solver and have separate adaptation processes, [89] devised a method involving the entropy variables. In their work, they showed that these satisfy an adjoint relation, able to target regions with numerical dissipation. The procedure to actually determine the error is equivalent to that of the *computable correction* calculated by [57], however, given the flow solver nature (*DG*), the embedded mesh interpolation was substituted by the injection of the solution into a higher-order state. Once the error had been computed, cells were marked for refinement, with all their edges being split. The procedure was tested on a *NACA* 0012 case, with low *Mach* number in either inviscid or viscous conditions. In both cases improvement of the flow solution was shown, especially *w.r.t.* traditional adjoint techniques. In particular, the entropy-adjoint approach refined the grid in a noticeably different manner *w.r.t.* the other adjoint cases. Issues did arise when they adapted the same test-case run in transonic flow conditions. In fact, the shock presence caused problems in terms of entropy production, as it was no longer conserved. To be noted, is the fact that their traditional adjoint adaptation cycles showed oscillatory behaviour, and in some cases, they had difficulty adapting the wake, due to the adjoint weighting.

[90, 91] attempted to devise a process to be able to achieve *anisotropic* adjoint mesh movement. Firstly, they defined the functional error as the difference between the coarse value and the corrected embedded grid one, as calculated in [64]. To be noted is the linear interpolation they employed for the prolongation of quantities from coarse to fine mesh. Following this, they formulated a *Lagrangian* relationship, containing both the error value and the *NS* eq. as the constraint. Minimising this *w.r.t.* the grid coordinates, allowed it to be combined with the original error equation, thus producing a modified adjoint relation. The result of this, may then be utilised in a steepest descent search

algorithm to minimise the overall mesh error by moving the nodes. The procedure they devised included also global refinement if the change in the fine grid estimate was sufficiently small. They successfully applied it to a 2D *Poisson* problem and an inviscid subsonic diverging nozzle. The procedure appeared to have greater difficulty when the case is transonic, with the authors arguing it was due to the shock presence. Nevertheless, apart from the functional of interest's adjoint, the process required an additional adjoint solve for the error sensitivity *w.r.t.* the mesh coordinates, and a *steepest descent* algorithm for each adaptation step.
In the series of articles, [92–97], the first use of the *mesh adjoint* output was employed to adapt a grid. In [92], the functional sensitivity *w.r.t.* the grid coordinates was modified, by removing the normal components to the surface of integration of the quantity of interest. The authors claimed that this was required to avoid unnecessarily strong clustering in this region. Moreover, they also forced to 0 the sensitivity at sharp geometrical corners, but leaved the value in the remainder of the flow untouched. They showed the validity of their approach by optimising a grid for the *Poisson* equation with a steepest descent algorithm. Interestingly, from their first tests they realised that there was a need for smoothing of the overall sensitivity, as nodal relocation at the *LE* of the inviscid *NACA* 0012 test-case proved to be erratic. Despite only allowing relocation of small parts of the domain, the authors showed how the functional sensitivity to the mesh reduced after the adaptation. Shortly after, in [93], refinement of the grid was added into the overall adaptation by adding nodes where the mesh adjoint output was strongest. Again, as in the previous case, the flow was set as inviscid, but most importantly, as the adaptation progressed, they were able to confirm their previous finding, i.e. a reduction in the mesh sensitivity once the grid had been adapted. In a later stage, [94] employed the previous findings to develop a structured mesh regeneration strategy based on the mesh adjoint sensitivity. In particular, once the functional derivative *w.r.t.* the mesh coordinates had been modified according to [92], this was multiplied by the edge length to determine a quantity that would be fed into the elliptic mesh generation algorithm. Finally, [96] was able to apply the method that had been developed to unstructured grids and showed the results for an inviscid *NACA* 0012 case. Interestingly, the comparison of the final adapted grid with those obtained by [9, 87] proved that their grid regeneration strategy was able to produce clustering in the same regions of the flow.

Given the test-cases considered in this work a separate discussion is required for adjoint-error turbomachinery applications. In fact, apart from [98, 99] which refer to the results of the work related to this thesis, only two examples of adjoint mesh adaptation for these cases exist.
The first is that of [100]. The authors modelled a 2D laminar compressor cascade and

run the procedure for entropy generation and total pressure losses. The grids were all triangular and adapted by applying a combination of movement/smoothing and refinement all based on the adjoint error. The movement approach consisted in a weighted modified *Laplacian* algorithm developed in [101] that would equidistribute the adjoint error.

A more recent attempt was that of [102] who calculated an adjoint-related error on a 3D viscous turbomachinery component in the *Hydra* ([103]) flow solver. In this case, the authors modified the adjoint weighting of the residuals by replacing the latter with a truncation error formulation. Instead of being evaluated on fine grid, this was calculated using the coarser multigrid levels residuals. Restriction and prolongation of the error was then carried out using the flow solver's multigrid interpolation operators. The authors consistently found the need to apply *pseudo-Laplacian* smoothing to remove the oscillatory behaviour, appearing due to the truncation-error calculation. No adaptation on the turbomachinery component using this sensor map was performed.

2.3.3 Truncation Estimation Approaches

In general, the *te* is defined as the difference between the continuous *p.d.e.* and the discretised version [13]. It may be determined by employing the *Taylor-Series* expansion, but this can be quite complicated and long-winded, unless the *p.d.e.* at hand is relatively simple [104]. To this end, [105, 106] employed the *Taylor-Series* to derive expressions relating mesh quality and truncation error for typical 2D and 3D *CFD* grids. Despite the validity of the technique elaborated, it is relatively complex even in its 2D formulation.

The *te* effect, was studied in detail by [12]. In his work, the author showed how this quantity is actually a source of error that is then transported by the *NS* relations to other regions of the domain. To this end, the *Continuous* & *Discrete* versions of the *Linear Error Transport* equation were derived. The first truncation error estimation approach suggested by the author would either require the exact continuous solution or, in its absence, a *Richardson Extrapolation*. The second technique necessitates the construction of the continuous operator. While this process is feasible for the case of *FEM* discretisations, for *FDM* and *FVM* the procedure is more involved as it would require a curve fitting process.

Another example of *te*-base error estimation procedure is the so-called τ methodology [107], related to multigrid approaches [104]. In the latter publication, the authors suggested using a *te* calculated using the fine grid flow solution residuals on the coarsest grid of the multigrid scheme. To be able to achieve a reliable quantity, a high-order interpolation scheme is required, but as in their case the coarse mesh nodes coincide

with the fine grid ones, it is unnecessary. This feature also allows to achieve consistency of the *te* as the two meshes will have the same characteristics (as would be required in a *Richardson Extrapolation* [13]). The quantity obtained is then employed in a sensor containing information concerning the order of the discretisation and each element volume. This is then smoothed employing a *Laplacian* operator and compared to a user-defined threshold. Finally the coarse mesh is refined in a cell-based manner, and instead of applying a similar process to the finer mesh of the multigrid, the adapted coarse grid is refined uniformly and is used as the fine grid in the next flow solver calculation.
For other material concerning *te*-based approaches, the interested reader is referred to the references in [108], while other examples relating the *te* to adjoint adapted techniques can be found in [109].

2.4 Adaptation Mechanics

According to [11], there are mainly 4 grid modification approaches:

- *r-methods* ⇒ movement;
- *h-methods* ⇒ refinement and coarsening;
- *p-methods* ⇒ order enrichment;
- *m-methods* ⇒ regeneration.

An overview of these methods will be discussed in the sections that follow.

2.4.1 *r*-Methods

As discussed [110], a mesh movement algorithm has three components:

1. *monitor function* ⇒ quantity driving the node relocation, in general the computed error;
2. mesh equations ⇒ map between computational and physical domain;
3. interpolation ⇒ estimate solution at the new position of the nodes.

The *monitor function* may be determined in three ways: *a-priori*, *a-posteriori* and based on physical arguments [111]. In the case of an *isotropic* movement, the node relocation

driving quantity is a scalar, as it does not have any directionality information. On the other hand, to achieve *anisotropy* a matrix-valued quantity is required. Once the *monitor function* has been determined, it is necessary to *equidistribute* it over the physical domain. This can be achieved in different manners, i.e. *geometric conservation laws*, *optimal transport* or moving mesh *p.d.e.*s [111]. The resulting equation is then discretised and can be solved simultaneously with the physical problem at hand. Alternatively, it can be evaluated sequentially. It is clear that it is the latter approach that requires interpolation of the flow field. Once the movement equation has been solved, it will either produce the node velocity or location for the adapted mesh.
As described by [112], an alternative approach to *equidistribution* to derive the mesh equations, is that employing error *direct minimisation*. By formulating the minimisation in terms of node locations and *p.d.e.* variables, an extra equation may be solved simultaneously to those of the problem at hand. This adds a further unknown to the overall system, i.e. the node position. This technique is generally known as *Moving FEM*.
One of the most successful mesh movement methods employs a spring-stiffness approach. In this case the grid edge lengths are the spring-stiffness, while the error can be seen as the potential energy at each node [29]. In this case, the *equidistribution* is achieved by determining the node locations to achieve system equilibrium. The spring potential energy equations are solved separately to that of the *NS* relations and in an iterative fashion. Therefore they will require interpolation of the flow solution or error quantity.
According to [11], there is a further r-adaptation approach: once the error has been determined, this may be employed in an optimisation approach determining the node location minimising the error. An example of such a method is that employed by [91].
A final mention should be made for the mesh movement/deformation methods devised to account for geometry changes. In fact, this may apply in the case of an optimisation, or in an unsteady simulation with moving items (e.g. to study fluid-structure interaction). Examples of these techniques are the linear spring method and elasticity, *Laplacian* smoothing and procedures using the interpolation analogy [113]. The very same authors also provide an overview and classification of the various methodologies.

2.4.2 *h*-Methods

Possibly the most popular mesh adaptation approaches, refinement and coarsening methods have found widespread use in the *CFD* community. A while back, [114] summarised the advantages of h-adaptation as:

1. physical conservation of quantities;
2. robustness;

3. parallelisation.

In their study on tetrahedral meshes, only 3 refinement patterns were allowed, that would be even more constrained in subsequent splitting of the resulting elements. This with the aim of maintaining mesh quality. On the other hand, for the coarsening, they allowed more possibilities, independently of whether adaptation had been previously applied to the cells. An interesting point concerning their work, is the effort to try and reduce the amount of information stored from the previous grid, in an attempt to minimise the memory burden.
Again employing tetrahedral grids, [115] studied the refinement effect on the resulting element's quality, and showed that regardless of the amount of adaptation steps, a limited set of similar tetrahedra can be created.
Shortly afterwards, [116] attempted to refine fully tetrahedral grids in an *isotropic* fashion to improve control over mesh quality. One of the author's conclusions relates to the difficulty in employing refinement of simplicial elements while achieving alignment with the flow *anisotropy*. Therefore they deducted that hexahedral elements would be better suited for this purpose. Moreover, in a later article ([117]), they claimed that by using these cell types, there is a gain in terms of accuracy and resulting grid quality. In this work, an *anisotropic* h-refinement strategy was developed: edges below a user-defined threshold would be removed (coarsening), while those above it would be split (refinement). It should be noted, that coarsening of the grid was only carried out on elements marked to be removed that appeared as a consequence of previous adaptation steps' refinement. This means that, no parts of the mesh were coarser than the starting grid. Additionally, to avoid *hanging nodes*, hybrid elements were employed at the interface between finer and coarser regions of the mesh. If these elements required subdivision in future refinement steps, they were removed and the starting hexahedra would be split instead. An important mention concerns the refinement propagation into undesired regions: in fact, it was noted that extra edges would be marked in an attempt to fulfil the splitting templates available. To avoid this issue, unless the edge marking allowed hexahedra to be split into smaller hexahedra, the cell was split into 6 pyramids by placing a node at its centroid.
As for the previous case, [118] used a *hierarchical* approach requiring template-splitting definition. Concerning tetrahedral volumes, only three valid configurations were chosen: each of these cell types could only be refined once without splitting it in an *isotropic* fashion. A similar approach was chosen for hexahedral elements, these could only be split into smaller cubes, unless they formed the interface between fine and coarse mesh zones. In total the refinement algorithm allowed 8 possible subdivisions of hexahedral and pyramidal cells, while 9 were permitted for prisms. The actual code implementation

consisted in splitting each volume into its faces and determining the new cells according to the face marking combinations. To avoid coding repeated templates for different orientations of each cell, rotation of their connectivity data was applied.

As mentioned in section 2.3.1, [38, 39] developed a methodology to be able to refine a fully hexahedral mesh without introducing new types of cells, yet maintaining the grid conformal. This was an extension of the approach discussed in [40, 41]. The process consisted in splitting each edge of an hexahedra into three and applying a *"shrinking"* of the hexahedral layers. In a final step, the grid connectivity would be updated. Shortly afterwards, [119] extended the method just outlined by allowing local refinement of either nodes, edges, faces and cells, thus increasing the flexibility. As a progression of this work, [120], included the possibility of coarsening, maintaining a conformal grid with solely hexahedral elements. Two ways of doing this were proposed, the first one, a global process, consisted in removing a sheet of connected cells. However, given that this could cause issues further away from the actual coarsening region, they also devised a local operation. [121] further improved the sheet-refinement algorithm of [119] by including the element-by-element enrichment approach of [41].

The issue relating to the generation of *hanging nodes* and different levels of refinement when employing fully hexahedral grids, was also discussed by [122]. As in previous cases, the interface elements between finer and coarser regions were never refined: in subsequent adaptation steps they would be removed to minimise poor quality. Moreover, the authors managed to reduce the amount of possible transition cases between the two different mesh density regions.

The effect of over-refinement due to the error directionality not being parallel to the cell faces was recently discussed by [122]. In this case, the refinement templates were defined by their edges, but the constraints on the possible combinations would be governed by the faces. For e.g., in their case they allowed a quadrilateral to have all 4, 2 parallel or 2 adjacent (i.e. sharing a node) and a single edge to be marked, but not 3, in this case the fourth would be forced to be split, propagating the refinement. Again, as in previous articles, the authors decided to remove any interface element between the refined hexahedral and the coarser mesh regions at the start of the next adaptation step.

A final publication of interest is [123]: here a discussion of the differences between splitting edges of a fully hexahedral mesh into 2 or 3 sub-edges is explored, with particular attention to the parallel implementation of the former. In fact, once the domain has been split, it is difficult to achieve the same refined mesh regardless of how the parallel division is carried out.

2.4.3 *p*-Methods

This type of mesh adaptation, sometimes called *order-enrichment*, consists in the increase or reduction of polynomial order in the flow solver. From an intuitive point of view, it is a *"flow solver refinement"* process as no change is applied to the the physical grid. These techniques are not applicable to *FDM/FVM* discretisations, but only to *high-order* methods[12], such as *FEM* or related methods (such as *DG* or *Spectral Elements*). For this reason, according to [11], they have found limited use in *CFD*, as in general *higher-order* solvers have greater complications in terms of *monotonicity* near discontinuities and in turbulent flow. Additionally, the author states that they are more difficult to code as there are more possible templates for each type of cell. Nevertheless, in more recent years, much of the flow solver-related research has been carried out with *high-order* methods, meaning that *p*-adaptation techniques have started to gain more and more momentum. Examples of the use of *order enrichment* can be found in [7, 14].

2.4.4 *m*-Methods

Mesh regeneration is now considered the best choice for grid adaptation, as it allows the greatest freedom, flexibility and accuracy in achieving a suitably modified mesh for a given amount of nodes. As discussed by [42], there are several paths one may take. The simplest approach consists in a partial regeneration of the grid. Once features have been detected, portions of the starting mesh are removed and filled with good quality elements, ideally aligned with the feature. An example of such an approach is provided by [124]. In their work, the authors employed a sensor to determine the feature location and then locally reconstructed the mesh around it using an *advancing front* approach. Effectively, extracting flow complexities and defining them as geometrical entities, allows this information to be employed in a complete mesh regeneration process. Therefore, the procedure may consist in placing a good quality, aligned quadrilateral/hexahedral mesh around the flagged complexity. The rest of the domain can then be filled with any combination of *hybrid* elements, as long as no *hanging nodes*[13] or negative volumes appear. As previously mentioned when discussing feature-based error estimation techniques (see section 2.3.1), examples of mesh regeneration treating the flow features as geometrical entities and regenerating the entire mesh are [21–23].
Over time, fully triangular/tetrahedral approaches to mesh regeneration have started to gain popularity due to the shape flexibility in handling very complex solutions and the feasibility in coding such methods. Unlike the previously mentioned techniques, these do not require any pseudo-geometrical information identifying the location of shocks,

[12] In general *high-order* methods have an order greater than 2, i.e. that of standard *FVM* approaches.
[13] It should be noted that flow solvers able to deal with *hanging nodes* are more frequent nowadays.

wakes ..., they simply employ a continuous metric field, as was described in section 2.3.1. In this approach, given an amount of nodes, the distribution of the elements may be optimised to allow alignment and clustering towards the regions of interest. For a detailed, up to date analysis of these techniques the reader is referred to [24, 125].

2.5 Summary

Within the literature's detailed analysis in the previous sections, the following points should be highlighted:

- Feature-based approaches:
  - flow complexities are *anisotropic* thus allow more efficient adaptation if this is considered in the error calculation;
  - throughout several decades of research, it has consistently been found that the best approach relates the second derivatives of a flow quantity to a *Riemannian* metric, thus achieving directional grid shrinking and stretching according to the local *anisotropy*;
  - *Hessian*-based techniques are sensitive to noisiness, particularly in the boundary layer, and may require artificial smoothing/eigenvector modification to remove this behaviour;
  - they do not provide any real stopping criteria to optimise the grid as flow complexities will continue to attract resources, regardless of whether they have been appropriately resolved or not;
  - they may attempt to adapt to spurious behaviour resulting in increased error;
  - a unique flow parameter is generally unable to target all flow features;
  - they are well suited for mesh movement and regeneration.
- Adjoint-based techniques:
  - consistently outperform their feature-based counterpart in minimising the error of a single quantity of interest with reduced number of nodes;
  - they increase solver run time per adaptation step as they require fully converged flow and adjoint solutions;
  - the most successful method devised (i.e. [64]) is expensive in terms of memory requirements, interpolation noisiness and has not been applied to turbomachinery cases;

  - there has been limited effort in attempting to devise an adjoint error to be employed with mesh movement techniques;
  - refinement strategies have generally made use of cell-based splitting algorithms without attempting to extract any error directionality from the adjoint quantities;
  - *anisotropy* has been included into the adjoint error estimation by simultaneous combination with a flow quantity's second-order derivative matrix.

- Adaptation mechanics:
  - mesh movement has been successfully combined with feature error estimation techniques while keeping the node count constant;
  - grid refinement is the most widespread technique and various different strategies have been devised to achieve *anisotropic* splitting of hexahedra while minimising the presence of hybrid elements;
  - local order enrichment is only applicable to high-order solvers;
  - complete mesh regeneration is possibly the most promising area of research, however it is limited to tetrahedral elements and is far more complex to code *w.r.t.* other techniques.

From these considerations, the first part of the research carried out in this work, focussed in combining the strengths of feature and adjoint based approaches. In fact, *Hessian*-based *Riemannian* metric, has shown good robustness when employed in mesh movement strategies and does not require any adjoint solution or even a perfectly converged flow field. However, possibly the main advantage of these techniques *w.r.t.* adjoint error estimation approaches, relates to their capability in attempting to improve the entire flow field, not only where a functional may be affected. On the other hand, adjoint adaptation may be useful in targeting unexpected regions of the domain and allow to achieve greater accuracy of the quantity of interest.
The second part of the work concentrated in attempting to include *anisotropic* information into the adjoint error estimation technique. In fact, this has not been achieved without the inclusion of feature-based error calculations. To this end, it would be interesting to be able to devise methodologies to be employed in mesh movement and edge-based grid refinement algorithms, thus improving the overall efficiency.

Chapter 3

Mesh Generation & Flow Solver Pre-Processing

3.1 System Overview

The ensemble of software utilised in this research project along with their order is shown in fig. 3.1. The overall system is very similar to what would be expected, however a breakdown of the capabilities and characteristics will be provided in this chapter (mesh generation and pre-processing) and following ones (flow and adjoint solvers, mesh adaptation software).

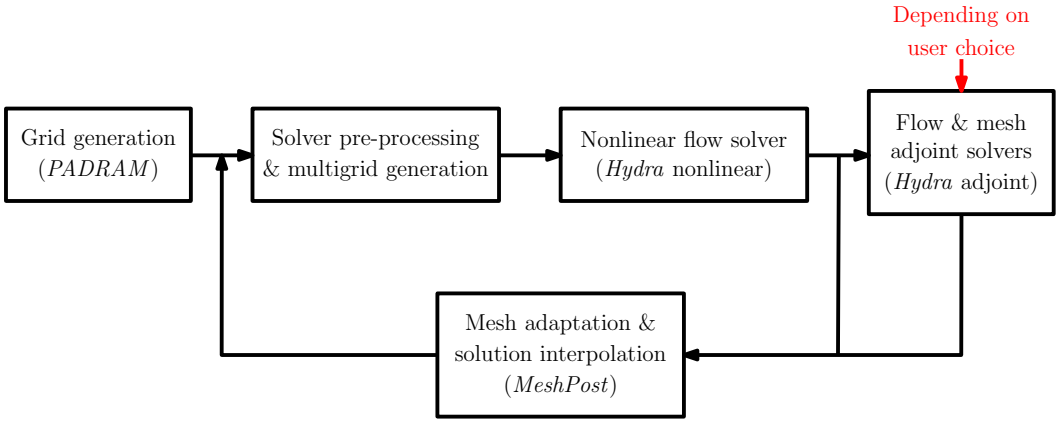


FIGURE 3.1: Overview of the *CFD* software employed.

3.2 Mesh Generation

The *RR* in-house meshing tool is *PADRAM*, short for *PArametric Design and RApid Meshing* for turbomachinery configurations [126]. The main characteristic of this software is the employment of templates to define various geometries present in a jet engine. Therefore a geometry will be divided into pre-defined blocks and consequently meshed in a *structured*, conformal manner. This procedure has the advantage of containing the user's effort in grid generation and additionally has a low run time. On the other hand, this method does reduce the overall flexibility of the software. Throughout its history, various configurations found in a typical jet engine have been added to the list of available templates, such as fans, compressors, turbines, cooling holes and many others. An example of a *multi-block structured* mesh at blade mid-span generated using *PADRAM* is shown in fig. 3.2.

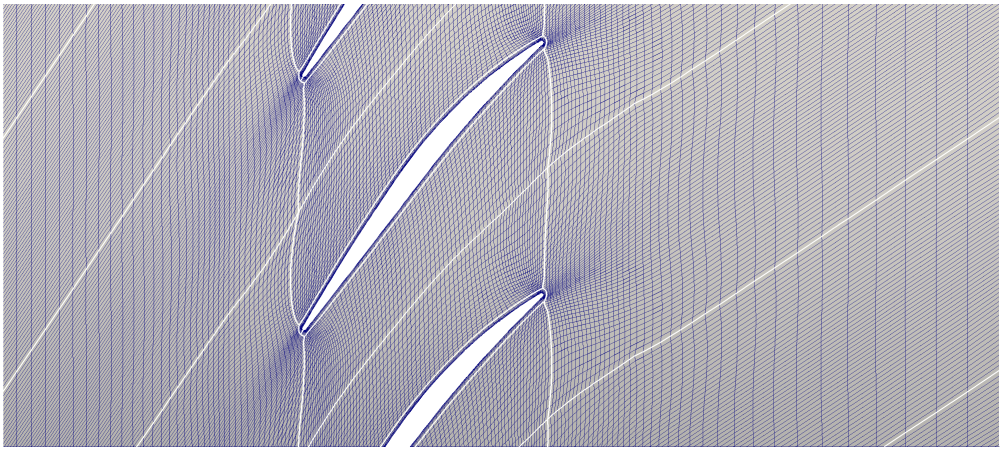
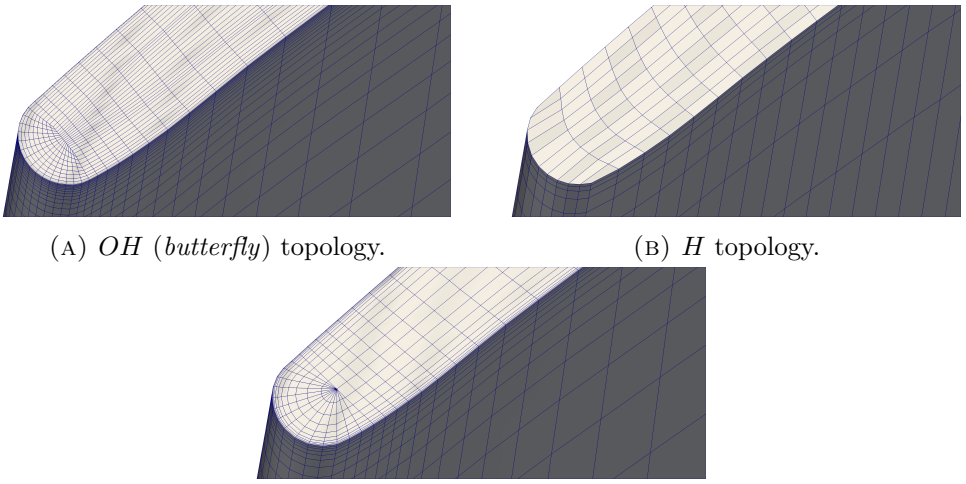


FIGURE 3.2: Example of blocks for *NASA Rotor* 37 at mid-span with periodic repeats.

In general, a rotational turbomachinery component will present the most complex flow features in the tip-gap region. Reason for which *PADRAM* is equipped with varied options to mesh this region of the domain. Examples of some choices available applied to a compressor blade are given in fig. 3.3.
The user has control over settings that would allow to change the location or smoothen the interfaces between blocks. In particular the grid smoothing algorithm employs the following equation:

$$\overrightarrow{x}^{i+1} = \overrightarrow{x}^{i} + \omega \sum_{j \in \text{neighbour}} (\overrightarrow{c}_j - \overrightarrow{x}^{i}) \tag{3.1}$$

Where $\overrightarrow{x}^{i}$ and $\overrightarrow{x}^{i+1}$ are the node's positions at the current and next iteration, ω is a weighting factor, *neighbour* is the set of all neighbouring cells and $\overrightarrow{c}^{j}$ are the cell centroids. Effectively this is a primal volume mesh smoothing algorithm, with the weighting



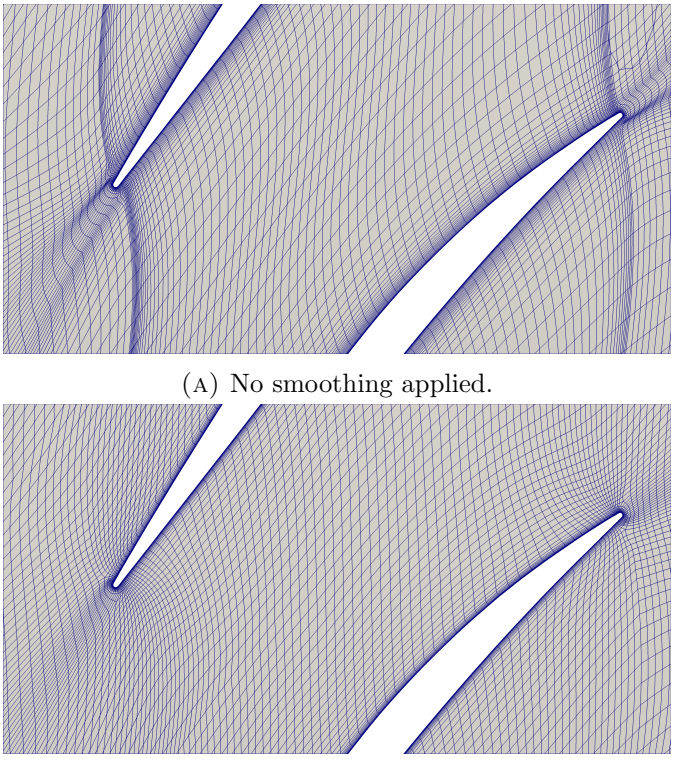
(A) *OH* (*butterfly*) topology.

(B) *H* topology.

(C) *Polar* topology.

FIGURE 3.3: Example of the various tip-gap mesh options in *PADRAM* applied to *NASA Rotor* 37.

being reduced in regions such as the near-wall layers. An example of the algorithm's effectiveness is clearly visible by comparing Figs. 3.4a and 3.4b.



(A) No smoothing applied.

(B) Smoothing on.

FIGURE 3.4: Example of *PADRAM* smoothing effect at mid span of a *NASA Rotor* 37 grid.

PADRAM has also been designed to have the capability to be employed in an automatic optimisation loop. In fact, parametrisation of the geometry using tri-cubic *splines* allows

the user to modify the design by using varied techniques such as *Free-Form Deformations* (*FFD*) [127] or *Hicks* and *Henne* bumps [128].

3.3 Solver Pre-Processing

This step in the overall simulation process consists in converting the grid from a *structured-multi-block* data format to an *unstructured* one, setting up boundary conditions, flow initialisation and multigrid coarse mesh generation.

3.3.1 Unstructured Data Conversion

The *structured multi-block* data imported from the mesh generator will contain information about block connectivity, boundary surface groups as well as the grid coordinates. It should be noted that for turbomachinery cases, a periodic angle is required, as the full annulus will be split into a single periodic repeat to be able to minimise simulation time. The flow pre-processor will convert the data to an *unstructured*[1] fashion, where extra information will be generated in order to determine the neighbouring nodes to a point in the mesh, for example. While this process will increase the grid files size, it is necessary to achieve a more flexible and generic approach. In the first part of the process, the connectivity is determined, as such:

1. *cell*→*node*
2. *edge*→*node*
3. *boundary node*→*node*
4. *boundary node*→*boundary type*
5. *boundary face*→*node*
6. $\cdots$

Once all the necessary connectivity arrays are determined, the mesh metrics may be computed:

1. *median-dual control surfaces*

[1] Often, the term *unstructured grid* is incorrectly employed to indicate a wide variety of meshes that are not fully hexahedral. However, in this work, it will refer to the way data is stored.

2. *median-dual control volumes*
3. *median-dual boundary control surfaces*
4. *wall distance*

The output grid file will be written in *HDF*5 (*Hierarchical Data Format* 5 [129]) as per the *CGNS* (*CFD General Notation System* [130]) guidelines.

3.3.2 Flow Initialisation & Boundary Conditions Setup

The boundary conditions and initial flow estimate files employed by the flow solver are determined by the pre-processing software using experimental data. In general this consists of a series of radial samples of axial, tangential and radial velocities along with static pressure and temperature taken at four stations along the domain: inlet, blade leading-edge (*LE*), trailing-edge (*TE*) and outlet. The flow is assumed turbulent to start with, while its intensity and length scale may be set by the user or determined by "*best-practice*" guidelines. The initial flow file estimate is computed by a linear interpolation of the experimental data provided. An initial flow file example for *National Aeronautics and Space Administration* (*NASA*) *Rotor* 37 is shown in fig. 3.5.

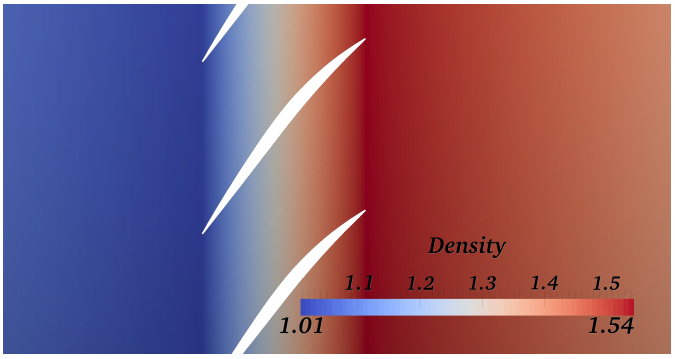


FIGURE 3.5: Example of density distribution in the initial flow file guess at mid-span of *NASA Rotor* 37.

3.3.3 Multigrid Mesh Generation

The coarser grid generation required for the employment of this kind of technique is that described by [131–134], generally known as *edge-collapsing*. The idea behind it is quite simple, i.e. if an edge is marked, then remove it along with its nodes and place a new point halfway between the original nodes. An example of this procedure is shown in fig. 3.6.

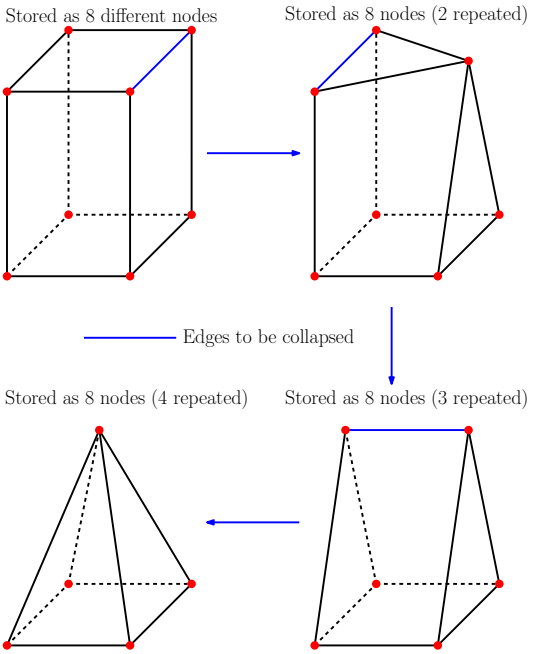
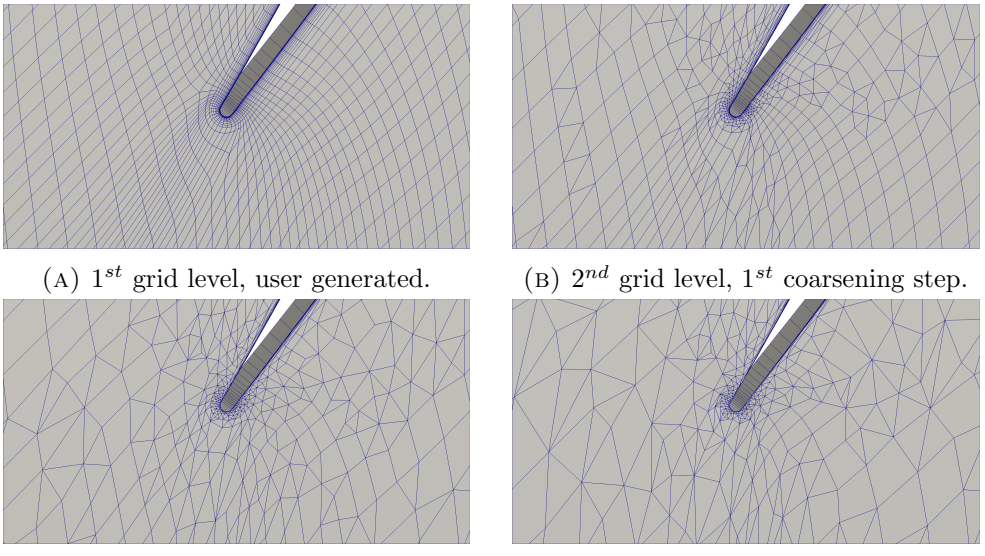


FIGURE 3.6: Example *edge-collapsing* applied to an hexahedron.

In the software employed to generate the coarsened grids, the resulting cells obtained from collapsing are stored in memory with repeated node indices where original points have been removed. This means that, for e.g., collapsed hexahedra will still be stored as having 8 nodes in memory, but may have identical indices in more than one position. Whilst this will increase the file size, it does allow much simpler and generic code implementation. An example of the resulting grids is given in figure 3.7.



(A) 1^{st} grid level, user generated.

(B) 2^{nd} grid level, 1^{st} coarsening step.

(C) 3^{rd} grid level, 2^{nd} coarsening step.

(D) 4^{th} grid level, 3^{rd} coarsening step.

FIGURE 3.7: Example of multigrid generation using *edge-collapsing* on a *NASA Rotor 37 multi-block structured* grid at blade *LE* and mid-radius height.

The original implementation of this kind of technique had limited success, because of the struggle to cope with geometry conservation, non-*simplex* shapes, *anisotropic* boundary layer grids and poor quality cell generation [132, 133]. To improve the methodology, [134] proposed a modified version: as in the case of the original algorithms, an *edge collapse* was allowed if the resulting edges did not exceed a certain length and generated cells had positive volume. Additional constraints concerning the maximum number of nodes collapsible into a single one (8 for 3D) and the requirement for a full set of parallel edges of an element to be removed, were included. Therefore instead of the shortest edge, the cell with the smallest primal volume was removed. It should be noted that the neighbouring cells to those collapsed will have had nodes removed, hence the resulting grid will contain unorthodox volumes. For this reason the nodes belonging to a collapsed volume have their identity replaced with the repeated index of the new point, thus allowing a more generic and simple implementation of the mesh metric functionality.
To be able to deal with *anisotropy* typically present in grids employed for *RANS* calculations, a modification to this procedure is required. Firstly, to detect the boundary layer grid, the fact that edges normal to the wall are short and aligned with each other as one moves away from the surface, while having longer sides connected to their nodes, is used. Additionally, a further constraint is added: if a short edge is marked to be removed, any of the long edges connected to its vertices must be kept. Once the stretched region of the grid has been dealt with, the rest of the domain is treated with the *isotropic* algorithm. A final remark should be made concerning the mapping between the fine and coarse mesh nodes. In fact, at this stage, extra data should be stored in the grid files to be able to map the coarse mesh nodes containing the fine mesh control volumes.

Chapter 4

Flow Solver

4.1 Introduction

The flow solver employed is *Hydra* [103], parallelised utilising the *Oxford Parallel Library for Unstructured Solvers* (*OPLUS*) as described in [135–137] and coded in *Fortran* 77 with memory allocation handled by *C*. It is a *FEM*-based *FVM* scheme [138] discretising the grid by means of a *median-dual* approach belonging to the category of *vertex-centred* solvers. The data is stored in an *unstructured* fashion, and to improve its efficiency operations are carried out in an edgewise manner. The flux reconstruction is determined by means of the *JST* with *matrix dissipation* approach [139]. The discrete equations are solved by marching in time employing an explicit *Runge-Kutta* (*RK*) technique, preconditioned using *block-Jacobi*. The reason behind choosing an explicit approach rather than an implicit[1] one to solve the flow relations is incidental to the memory consumption. In fact, as described by [140], despite the latter having better stability, it does require larger memory storage, it is more demanding in terms of *Central Processing Unit* (*CPU*) usage and more complex to parallelise. The same reasoning may also be applied to the preconditioner choice: [141] showed that *point-implicit block Jacobi* can aid convergence issues, however it is also subject to increased memory requirements. The next sections will describe the main aspects of the solver settings employed in this work, such as the spatial discretisation, flux reconstructions, parallelisation, and so on. In particular, as the work carried out in the thesis deals with the *RANS* equations, only the steady solver along with the *Spalart-Allmaras* (*SA*) turbulence model are described in detail. The choice of the *SA* model, is related to the adjoint code having a consistent,

[1] It should be noted that the latest release of *Hydra* has a semi-implicit scheme option to solve the *NS* relations. However, due to the adjoint code having only an explicit alternative, in order to have a consistent discretisation between the two solvers, the improved version was not considered.

stable and reliable discretisation of it.

4.2 Flow Equations

The *NS* equations can be written as the following system [131]:

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F} = 0 \tag{4.1}$$

The vector $\mathbf{Q}$ contains the flow variables in *conservative* form:

$$\mathbf{Q} = [\rho \;\; \rho u \;\; \rho v \;\; \rho w \;\; \rho E]^T \tag{4.2}$$

while $\mathbf{F}$ represents the flux vector, that may be split into x, y and z components:

$$\mathbf{F}_x = \begin{bmatrix} \rho u \\ \rho u^2 + P - \tau_{xx} \\ \rho uv - \tau_{yx} \\ \rho uw - \tau_{zx} \\ (\rho E + P)u - u\tau_{xx} - v\tau_{yx} - w\tau_{zx} - \widehat{k}\frac{\partial T}{\partial x} \end{bmatrix} \tag{4.3a}$$

$$\mathbf{F}_y = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + P - \tau_{yy} \\ \rho vw - \tau_{zy} \\ (\rho E + P)v - u\tau_{xy} - v\tau_{yy} - w\tau_{zy} - \widehat{k}\frac{\partial T}{\partial y} \end{bmatrix} \tag{4.3b}$$

$$\mathbf{F}_z = \begin{bmatrix} \rho w \\ \rho uw - \tau_{xz} \\ \rho vw - \tau_{yz} \\ \rho w^2 + P - \tau_{zz} \\ (\rho E + P)w - u\tau_{xz} - v\tau_{yz} - w\tau_{zz} - \widehat{k}\frac{\partial T}{\partial z} \end{bmatrix} \tag{4.3c}$$

where ρ is the density, $[u, v, w]$ the velocity vector, P and T the static pressure and temperature, respectively, $\widehat{k}$ is the coefficient of thermal conductivity and E is the total internal energy. The shear stresses τ are symmetric and formulated as:

$$\tau = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} = \tag{4.4}$$

$$\begin{bmatrix} 2\mu\frac{\partial u}{\partial x} + \widehat{\lambda}(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}) & \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \mu(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}) \\ \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & 2\mu\frac{\partial v}{\partial y} + \widehat{\lambda}(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}) & \mu(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}) \\ \mu(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}) & \mu(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}) & 2\mu\frac{\partial w}{\partial z} + \widehat{\lambda}(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}) \end{bmatrix}$$

with the *molecular viscosity* $\mu = \frac{1.461 \cdot 10^{-6} \cdot T^{\frac{3}{2}}}{T+110.3}$, as per *Sutherland*'s law, $\widehat{\lambda} = -\frac{2}{3}\mu$ being the *bulk viscosity* defined by *Stoke*'s hypothesis.
A further relation is required to relate the density and pressure: in *Hydra* this is done through the *ideal state gas law*[2]:

$$p = \rho RT = (\gamma - 1)\rho(E - \frac{1}{2}(u^2 + v^2 + w^2)) \tag{4.5}$$

with R being the *ideal gas constant* and γ the *specific heat ratio*. Due to the use of this relation, the flow solver falls into the *density-based* category.
The final relation required is that to achieve complete closure of the system and is provided by the *SA* turbulence model.

4.3 Grid Discretisation

In *FVM* discretisations the quantities are evaluated in each control volume (where they are constant[3]), however, the way these are constructed varies according to where the values are stored. *Hydra* accumulates flow variables at grid nodes, thus making it a *vertex-centred* solver, needing to compute extra mesh data (*mesh metrics*) to determine the control volumes and surfaces for each node. *Vertex-centred* techniques have various methods of computing these quantities, in this case the *median-dual* approach is employed. In this type of discretisation, the volumes around each node are determined by considering the edge, face and cell centroids that are then joined up. An example of this procedure for a $2D$ case is shown in fig. 4.1.

[2]While many cases are indeed run assuming the gas as ideal, more sophisticated approaches mimic more complex cases, such as gas exiting the combustion chamber, are available, however out of this thesis' scope.

[3]The values actually represent the average across the cell.

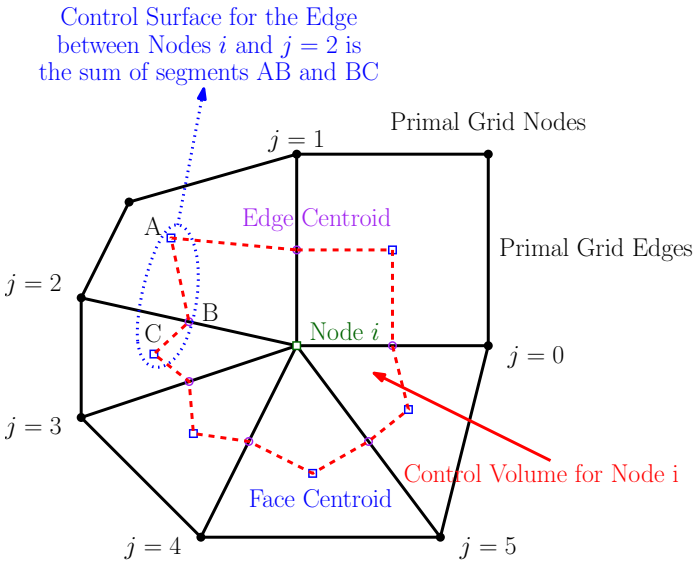


FIGURE 4.1: 2D Median-Dual Control Volume.

As it can be seen, the red lines form the control volume around node i, and are generally referred to as the *dual mesh*, while the original, user-generated grid is the *primal* grid. At this point it should be noted that the reason behind the choice of the *edge-based* information handling of the solver, is due to its *vertex-centred* nature ([67]).

4.4 Navier-Stokes Discretisation

Now that the grid has been split into distinct control volumes, the flow relations may be discretised between them. In this section, the same approach as that in [131] will be followed. Considering the *NS* equations with source terms:

$$\frac{\partial \mathcal{Q}}{\partial t} + \nabla \cdot \mathcal{F}(\mathcal{Q}, \nabla \mathcal{Q}) = \mathcal{S}(\mathcal{Q}, \nabla \mathcal{Q}) \tag{4.6}$$

Time averaging and integration of the above relations over a generic control volume (V) with bounding surface (S) yields:

$$\mathcal{R} = \frac{1}{V}\left[\oint_S \mathcal{F}(\vec{n}, \mathcal{Q}, \nabla \mathcal{Q}) dS - \int_V \mathcal{S}(\mathcal{Q}, \nabla \mathcal{Q}) dV\right] = 0 \tag{4.7}$$

Where the *Green-Gauss* or *Divergence Theorem*[4] was employed to modify the integration of the fluxes over a volume into one over the bounding surfaces. The *L.H.S.* term is the

[4]This is defined as:

$$\int_V \nabla \cdot \mathcal{F} dV = \oint_S \mathcal{F} \cdot \vec{n} dS$$

residual, i.e. the error due to imbalances of the effect of the fluxes and source terms. The flux integration may be discretised employing the control surfaces computed over all the edges to which a node is connected to (see also figure 4.1):

$$\oint_S \mathscr{F}(\overrightarrow{n}, \mathcal{Q}, \nabla\mathcal{Q})dS \approx \sum_{j\in\mathbb{E}_i} \mathbf{F}(\overrightarrow{n}_{ij}, \mathcal{Q}, \nabla\mathcal{Q})\big|_{\frac{1}{2}(\overrightarrow{x_i}+\overrightarrow{x_j})}\Delta s_{ij} \tag{4.8}$$

With the summation being carried out over all the edges connected to node i and Δs_{ij} being the control surface halfway along edge ij. The source term is obtained by simply multiplying its contribution by the actual volume size. Therefore 4.7 may be written as:

$$\mathbf{R}_i = \frac{1}{V_i}\Big(\sum_{j\in\mathbb{E}_i}(\mathbf{F}_{ij}\Delta s_{ij}) - \mathbf{S}_i\, V_i\Big) \tag{4.9}$$

In the case node i sits on one of the domain's boundaries:

$$\mathbf{R}_i = \frac{1}{V_i}\Big(\sum_{j\in\mathbb{E}_i}(\mathbf{F}_{ij}\Delta s_{ij}) + \sum_{j\in\mathbb{B}_i}(\mathbf{F}_j\Delta s_j) - \mathbf{S}_i\, V_i\Big) \tag{4.10}$$

The second accumulation is evaluated over the *median-dual* boundary faces to which the node belongs. Finally, it is possible to split the inviscid ($\mathbf{F}^I$) and viscous contribution ($\mathbf{F}^V$) of the fluxes:

$$\mathbf{R}_i = \frac{1}{V_i}\Big(\sum_{j\in\mathbb{E}_i}(\mathbf{F}^I_{ij} + \mathbf{F}^V_{ij})\Delta s_{ij} - \mathbf{S}_i\, V_i\Big) \tag{4.11}$$

The next sections will describe in detail the computation of $\mathbf{F}^I$ and $\mathbf{F}^V$.

4.5 Flux Discretisation

4.5.1 Inviscid Fluxes

The starting point for the scheme developed for *Hydra* is the first-order *upwind* inviscid flux discretisation approach of *Roe* [142] (part of the *flux difference-splitting* or *Riemann* class), where a *Riemann* problem is solved at the control surface interface between two nodes:

$$\mathbf{F}^I_{ij} = \frac{1}{2}\big(\mathscr{F}^I_{ij}(Q_i) + \mathscr{F}^I_{ij}(Q_j) - |A_{ij}|(Q_i - Q_j)\big) \tag{4.12}$$

With $\mathscr{F}_{ij}^{I}(Q_i)$ being the inviscid flux contribution of node i along edge connecting nodes $i,\ j$, while $|A|$ is the inviscid flux *Jacobian* matrix evaluated using the geometric average of states of nodes $i,\ j$ with the absolute value of the eigenvalues. The procedure actually consists in a linearisation of the *Jacobian* matrix based on the *characteristic* variables, reason for which an *entropy-fix* is required [143].
To be able to improve the accuracy of the method, the state values may be reconstructed using their higher order interpolation from the nodes to the control surface splitting them [26] (fig. 4.2):

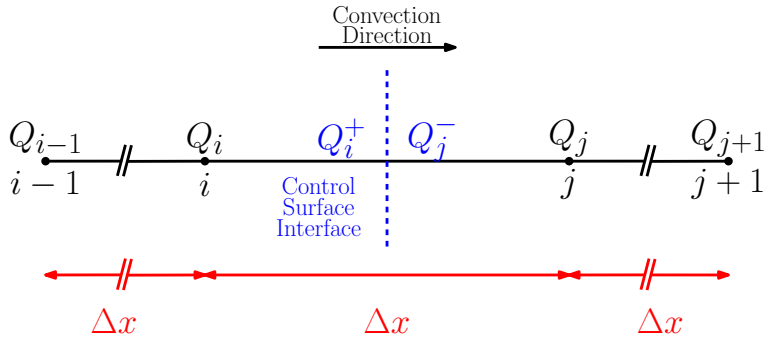


FIGURE 4.2: Flux reconstruction for edge $i,\ j$.

$$\mathbf{F}_{ij}^{I} = \frac{1}{2}\left(\mathscr{F}_{ij}^{I}(Q_i^+) + \mathscr{F}_{ij}^{I}(Q_j^-) - |A_{ij}(Q_i^+ - Q_j^-)|(Q_i^+ - Q_j^-)\right) \tag{4.13}$$

The higher order reconstruction is provided by the *Monotone Upwind Scalar Conservation Law* (*MUSCL*) approach of *van Leer* [144, 145]:

$$Q_i^+ = Q_i + \frac{1}{4}\Big((1-\kappa)(Q_i - Q_{i-1}) + (1+\kappa)(Q_j - Q_i)\Big) \tag{4.14a}$$

$$Q_j^- = Q_j - \frac{1}{4}\Big((1-\kappa)(Q_{j+1} - Q_j) + (1+\kappa)(Q_j - Q_i)\Big) \tag{4.14b}$$

Where κ is a constant that determines the scheme employed (see Table 4.1).

Scheme	upwind	semi-upwind (*Fromm*)	central differencing	upwind $\mathcal{O}(3)$
κ	-1	0	1	$\frac{1}{3}$

TABLE 4.1: *MUSCL* scheme types according to the different values of κ [146].

In the case of *Hydra*, the value of $\kappa = \frac{1}{2}$ was chosen, thus Q^+ and Q^- may be written as:

$$\begin{aligned} Q_i^+ &= Q_i + \frac{1}{4}\Big((1-\kappa)(Q_i - Q_{i-1}) + (1+\kappa)(Q_j - Q_i)\Big) \\ &= Q_i + \frac{1}{4}\Big(\frac{1}{2}(Q_i - Q_{i-1}) + \frac{3}{2}(Q_j - Q_i)\Big) \\ &= Q_i + \frac{1}{8}(3\,Q_j - 2Q_i - Q_{i-1}) \end{aligned} \tag{4.15a}$$

$$\begin{aligned} Q_j^- &= Q_j - \frac{1}{4}\Big(\frac{1}{2}(Q_{j+1} - Q_j) + \frac{3}{2}(Q_j - Q_i)\Big) \\ &= Q_j - \frac{1}{8}(Q_{j+1} + 2Q_j - 3Q_i) \end{aligned} \tag{4.15b}$$

At this point, [131], states that to speed up the overall algorithm, the $\mathscr{F}_{ij}^I$ terms and the *Jacobian* matrix, should be evaluated using the nodal quantities without any reconstruction of Q^+ and Q^-. Therefore equation 4.13 is written as:

$$\mathbf{F}_{ij}^I = \frac{1}{2}\big(\mathscr{F}_{ij}^I(Q_i) + \mathscr{F}_{ij}^I(Q_j) - |A_{ij}|(Q_i^+ - Q_j^-)\big) \tag{4.16}$$

Substituting the results of 4.15a and 4.15b into 4.16:

$$\begin{aligned} \mathbf{F}_{ij}^I &= \frac{1}{2}\big(\mathscr{F}_{ij}^I(Q_i) + \mathscr{F}_{ij}^I(Q_j) - |A_{ij}|\big(Q_i + \frac{1}{8}(3\,Q_j - 2Q_i - Q_{i-1}) - Q_j + \frac{1}{8}(Q_{j+1} + 2Q_j - 3Q_i)\big) \\ &= \frac{1}{2}\big(\mathscr{F}_{ij}^I(Q_i) + \mathscr{F}_{ij}^I(Q_j) - |A_{ij}|\big((Q_i - Q_j) + \frac{1}{8}(Q_{j+1} + 5Q_j - 5Q_i - Q_{i-1})\big) \\ &= \frac{1}{2}\big(\mathscr{F}_{ij}^I(Q_i) + \mathscr{F}_{ij}^I(Q_j) - |A_{ij}|\frac{1}{8}(Q_{j+1} - 3Q_j + 3Q_i - Q_{i-1}) \end{aligned} \tag{4.17}$$

At this point, it is possible to introduce the discrete *Laplacian* operator of Q_i and Q_j into 4.17. In general, for node i this is defined as:

$$L(Q_i) = \frac{1}{n^\circ\ \mathbb{E}_i}\sum_{j\in\mathbb{E}_i}(Q_j - Q_i) \tag{4.18}$$

Where the summation of the difference in values is carried out over all the edges to which nodes i or j are connected to and then averaged. It should be noted that this operator

is *linearity preserving*[5] and indicates the mean difference of values in the neighbouring region to i. For the stencil shown in Fig 4.2, the *Laplacian* operators of Q_i ($L(Q_i)$) and Q_j ($L(Q_j)$) are defined as:

$$\begin{aligned} L(Q_j) - L(Q_i) &= \overbrace{\frac{1}{2}\big((Q_i - Q_j) + (Q_{j+1} - Q_j)\big)}^{L(Q_j)} - \overbrace{\frac{1}{2}\big((Q_j - Q_i) + (Q_{i-1} - Q_i)\big)}^{L(Q_i)} \\ &= \frac{1}{2}(-Q_{i-1} + 3Q_i - 3Q_j + Q_{j+1}) \end{aligned} \tag{4.19}$$

As it can be easily seen, the resulting expression is four times that multiplied by the *Jacobian* matrix in equation 4.17. Therefore:

$$\mathbf{F}^I_{ij} = \frac{1}{2}\big(\mathscr{F}^I_{ij}(Q_i) + \mathscr{F}^I_{ij}(Q_j) - \frac{1}{2}(1-\kappa)|A_{ij}|(L(Q_j) - L(Q_i)) \tag{4.20}$$

It is important to comment on the resulting formulation of the scheme. The first term on the *R.H.S.* is a *central difference* of the fluxes, while the *Jacobian* multiplication by the *Laplacian* terms, can be seen as a smoothing term. Effectively, it is the latter that provides the *upwinding* treatment to the *centred difference* of inviscid fluxes [139, 147]. Original formulations of the *centred scheme* did not contain any information regarding the flow direction, and therefore generally had poor resolution in regions where shocks appeared. On the other hand they significantly outperformed the fully *upwind* discretisations in terms of simulation time (roughly by a factor of two [139]). In an industrial environment, flow solver speed is of crucial importance and therefore the *centred scheme* justified a significant effort. For this reason, a hybrid scheme was developed by [139], generally indicated as *JST* with *Matrix Dissipation*. The former is recovered from equation 4.20 by including the *JST* switch terms (Φ) along with second order differences:

$$\mathbf{F}^I_{ij} = \frac{1}{2}\left(\mathscr{F}^I_{ij}(Q_i) + \mathscr{F}^I_{ij}(Q_j) - |A_{ij}|\left(\Phi \underbrace{(Q_i - Q_j)}_{\substack{\text{Second-Order}\\ \text{Differences}}} - \frac{1}{3}(1-\Phi)\overbrace{(L(Q_i) - L(Q_j))}^{\substack{\text{Fourth-Order}\\ \text{Differences}}}\right)\right) \tag{4.21}$$

[5] This is easily proved considering the continuous 2D *Laplacian* operator:

$$L(Q) = \nabla Q = \frac{\partial^2 Q}{\partial x^2} + \frac{\partial^2 Q}{\partial y^2}$$

for any linear variation the second derivative is by definition 0. Additionally, this operator is also zero for constant functions, i.e. there is no average difference of quantities around point i.

Where the dissipation switch is defined as:

$$\Phi = min\left(\epsilon^{(2)}\left|\frac{P_j - P_i}{P_j + P_i}\right|^2, 1\right) \tag{4.22}$$

With P_i being the static pressure at node i and $\epsilon^{(2)}$ a constant, generally taken to be 8 [131]. The switch will be triggered once a shock is detected, i.e. where substantial differences in static pressure appear. This will remove the *Laplacian* difference effect (i.e. $\mathcal{O}(4)$ terms) increasing the weight of second-order terms multiplied by the *Jacobian* matrix, in turn reducing the scheme accuracy to $\mathcal{O}(1)$ (i.e. satisfying *Godunov's theorem*). This will allow a smooth capturing of the flow discontinuity. On the other hand, when the behaviour is close to linear, the opposite will happen. This has the aim of removing any spurious oscillations due to the higher than first-order terms.
A final comment should be made regarding the *matrix-dissipation* treatment. In fact, *upwinding* is provided by the scaling of each equation's dissipative term by their relative inviscid flux *Jacobian* eigenvalues. Furthermore, an additional modification is required to avoid issues at *sonic* or *stagnation points*, where *entropy condition* may be violated. In fact, according to *characteristic propagation of information* (at the heart of *Roe*'s scheme) at these points, an eigenvalue will approach 0. This will produce no artificial viscosity rendering the scheme unstable and any solution inconsistent (i.e. expansion waves appear). To overcome this, [148] proposed the following modification of the eigenvalue employed for the edge flux computation:

$$|\widetilde{\lambda}| = max(|\widetilde{\lambda}|, 2\Delta\widetilde{\lambda}) \tag{4.23}$$

Where $\widetilde{\lambda}$ is the original eigenvalue, while $\Delta\widetilde{\lambda}$ is the relative variation of the values across each edge. No flux limiting function has been included into the *Hydra* scheme, thus it is not *Total-Variation-Diminishing* (TVD). This means that it will generate new local minima/maxima within the solution.
A final point about this scheme relating to aspects of mesh refinement concerns its stability. In fact as discussed by [149], the *centred scheme* can become unstable when applied to *unstructured* hybrid grids.

4.5.2 Viscous Fluxes

The viscous fluxes computation is carried out by approximations at mid-edge points. For their evaluation the gradients halfway between the nodes are required. A simple

averaging of the nodal values (i.e. *centred difference*) to determine them will cause oscillatory behaviour, particularly in the boundary layer, where viscous terms will have the strongest influence. That is, it does not guarantee *positivity*. Therefore [131] implemented the following modification to determine the mid-edge derivative values:

$$\nabla Q_{ij} = \frac{1}{2}(\nabla Q_i + \nabla Q_j) - \left(\frac{1}{2}(\nabla Q_i + \nabla Q_j) \cdot \frac{\Delta \overrightarrow{x}_{ij}}{|\Delta \overrightarrow{x}_{ij}|} - \frac{\Delta Q_{ij}}{|\Delta \overrightarrow{x}_{ij}|}\right) \frac{\Delta \overrightarrow{x}_{ij}}{|\Delta \overrightarrow{x}_{ij}|} \tag{4.24}$$

Where ∇Q_i is the nodal gradient determined using a *Green-Gauss* (GG) approach, Δx_{ij} is edge ij's length vector and $\Delta Q_{ij} = Q_i - Q_j$.

4.5.3 Gradient Evaluation

Hydra computes the gradients employing a GG computation. This is based on the *divergence* or *Gauss*' theorem, namely:

$$\iiint_V \nabla a \, dV = \oiint_S a \cdot \mathbf{n} \, dS \tag{4.25}$$

This states that the flux of quantity a, in the surface normal direction $\mathbf{n}$, across the closed, bounding surface S of volume V (RHS of equation 4.25), is equal to the gradient of a integrated over the volume (LHS of equation 4.25). Considering the surface discretisation shown in fig. 4.1 [150]:

$$\iiint_V \nabla a \, dV = \sum_i a_i \cdot \mathbf{n}_i \, S_i \xRightarrow{\mathbf{n}_i \cdot S_i = \mathbf{S}_i} \iiint_V \nabla a \, dV = \sum_i a_i \cdot \mathbf{S}_i \tag{4.26}$$

where i is the discrete bounding surface index. In a FVM process, the value at each node is assumed to represent the average of that quantity over the relative control volume. Therefore, it is possible to move the gradient term on the LHS of 4.26 outside the integration:

$$\nabla a \iiint_V dV = \sum_i a_i \cdot \mathbf{S_i} \tag{4.27}$$

As the reader will have noticed, the integral is the cell control volume V:

$$\nabla a \, V = \sum_i a_i \cdot \mathbf{n}_i \, S_i \Rightarrow \nabla a = \frac{\sum_i a_i \cdot \mathbf{S_i}}{V} \tag{4.28}$$

This states that the gradient, in a FVM-sense, is equal to the sum of the value of interest over the surface of element i, multiplied by the control area vector and divided by the control volume.
For the case of an edge-based *unstructured* flow solver, utilising a *median-dual* control volume formulation, an efficient implementation is displayed algorithm 1.

Algorithm 1 Gradient computation for edge-based *unstructured* solvers

procedure GREEN-GAUSS GRADIENT
  for Each grid edge **do**
    Extract edge's node 1 and 2 indices
    Extract edge's dual area vector **S**
    $a|_{x,Mid-edge} S_x = \frac{1}{2}(a_2 - a_1)S_x$
    $a|_{y,Mid-edge} S_y = \frac{1}{2}(a_2 - a_1)S_y$
    $a|_{z,Mid-edge} S_z = \frac{1}{2}(a_2 - a_1)S_z$
    Accumulate contribution for each node
  end for
  for Each boundary node **do**
    Extract boundary area vector
    $a|_{x,Boundary} S_{bx} = a_b S_{bx}$
    $a|_{y,Boundary} S_{by} = a_b S_{by}$
    $a|_{z,Boundary} S_{bz} = a_b S_{bz}$
    Accumulate contribution for each boundary node
  end for
  for Each grid node **do**
    Divide accumulated value by control volume
  end for
end procedure

4.5.4 Linearity Preserving Laplacian Operator

The careful reader will have noticed that eq. 4.18 is a modified version of the discrete *Laplacian* relation, the exact one being:

$$L(Q_i) = \sum_{j \in \mathbb{E}_i} (Q_j - Q_i) \tag{4.29}$$

The summation being carried out over all the edges to which node i is connected. As described by [151], the additional averaging by the number of edges is helpful to remove grid dependencies, particularly where nodes are within uneven cell conformations.
[131, 151], also state that further modifications are necessary to avoid first order terms appearing in their discretisations for unsmooth grids. Considering a 3D *Taylor-Series* expansion of Q_i around Q_j [151]:

$$Q_i = Q_j + \Delta\overrightarrow{x} \cdot \nabla Q_j + \frac{1}{2}\left(\Delta x^2 \frac{\partial^2 Q_j}{\partial x^2} + \Delta y^2 \frac{\partial^2 Q_j}{\partial y^2} + \Delta z^2 \frac{\partial^2 Q_j}{\partial z^2}\right) + \Delta x \Delta y \frac{\partial^2 Q_j}{\partial x \partial y} + \Delta x \Delta z \frac{\partial^2 Q_j}{\partial x \partial z} + \Delta y \Delta z \frac{\partial^2 Q_j}{\partial y \partial z} + \mathcal{O}(\Delta\overrightarrow{x}^3) \tag{4.30}$$

Therefore:

$$\begin{aligned} L(Q_i) &= \frac{1}{n^\circ \mathbb{E}_i} \sum_{j \in \mathbb{E}_i} (Q_j - Q_i) \\ &= \frac{1}{n^\circ \mathbb{E}_i}\left(\sum_{j \in \mathbb{E}_i} \underbrace{\Delta\overrightarrow{x} \cdot \nabla Q_j}_{\mathcal{O}(1) \text{ terms}} + \mathcal{O}(\Delta\overrightarrow{x}^2) + \mathcal{O}(\Delta\overrightarrow{x}^3)\right) \end{aligned} \tag{4.31}$$

As it can be seen, first order terms do appear in the *Laplacian* operator computation. The modification applied by *Hydra* to avoid this issue is simply to subtract the problematic term from the overall operator computation [131]:

$$L^{lp}(Q_i) = L_i(Q_i) - L(\overrightarrow{x}_i) \cdot \nabla Q_i \tag{4.32}$$

the suffix *lp* stands for *linearity preserving* while $L(\overrightarrow{x}_i)$ is eq. 4.18 applied to the edge length vectors to which node i is connected to (i.e. this will result in a *Laplacian* in x, y and z). At the boundaries, this approach will cause convergence issues, as the whole term may approach zero. Therefore the *lp* alteration is limited by modifying the $L(\overrightarrow{x})$ at the boundaries [131]:

$$L^{bnd}(\overrightarrow{x}_i) = L(\overrightarrow{x}_i) - (L(\overrightarrow{x}_i) \cdot \overrightarrow{n})\overrightarrow{n} \tag{4.33}$$

With *bnd* indicating *boundary node* and $\overrightarrow{n}$ being the surface normal at that node. This may cause the lower order terms to re-appear in the flux dissipative term computation, however the contribution will be negligible at smooth surfaces.
Further issues in the computation of the *Laplacian* terms may materialise when the grid is stretched, e.g. in the boundary layer. In these cases the $L(\overrightarrow{x})$ may cause the gradient term included to have a larger contribution *w.r.t.* the $\mathcal{O}(2)$ terms. To overcome this issue $L(Q_i)$ is changed to [131]:

$$L(Q_i) = \left(\sum_{j \in \mathbb{E}_i} \frac{1}{|\overrightarrow{x}_j - \overrightarrow{x}_i|}\right)^{-1} \sum_{j \in \mathbb{E}_i} \frac{Q_j - Q_i}{|\overrightarrow{x}_j - \overrightarrow{x}_i|} \tag{4.34}$$

4.6 Time-Stepping Approach

Hydra utilises a 5-step, low-storage[6] *RK* explicit iterative scheme to reach convergence (although 3 and 7-step algorithms are also available). In general, this approach can be written as [152]:

$$\begin{aligned} \mathbf{Q}_j^{(0)} &= \mathbf{Q}_j^{(n)} \\ \mathbf{Q}_j^{(k)} &= \mathbf{Q}_j^{(0)} - \alpha^{(k)} \Delta t_j \mathbf{R}_j^{(k-1)} \; \forall \, k = 1, .., 5 \\ \mathbf{Q}_j^{(n+1)} &= \mathbf{Q}_j^{(5)} \end{aligned} \tag{4.35}$$

With k representing the *Runge-Kutta* step, $\mathbf{Q}_j^{(n)}$ the flow solution at node j for iteration n, α a multiplier between 0 and 1, Δt_j the local time-step for point j and $\mathbf{R}_j^k$ the residual evaluated at step k of the iterative scheme comprising all fluxes. By splitting the residual evaluation into convective ($\mathbf{C}$) and dissipative ($\mathbf{D}$) terms, it is possible to avoid updating the entire $\mathbf{R}_j^k$ vector at every iteration, thus reducing the memory burden of the scheme [131]. This kind of approach is known as *Additive Runge-Kutta*, initially developed by [153], and employed in *Hydra*. Here the residual is modified as follows:

$$\mathbf{R}_j^{(k-1)} = \mathbf{C}_j^{(k-1)} - \mathbf{B}_j^{(k-1)} \tag{4.36}$$

Where $\mathbf{B}_j^{(k-1)}$ is a combination of the dissipative residuals of the current and previous step:

$$\mathbf{B}_j^{(k-1)} = \beta_k \mathbf{D}_j^{(k-1)} + (1 - \beta_k) \mathbf{B}_j^{(k-2)} \tag{4.37}$$

Like α, β also represents a coefficient between 0 and 1: the complete set of values of both for every *Runge-Kutta* iteration can be found in table 4.2.

Step	1	2	3	4	5
α	0.25	$\frac{1}{6}$	0.375	0.5	1
β	1	0	0.56	0	0.44

TABLE 4.2: 5-step *Runge-Kutta* coefficients

[6] This means that memory requirements are reduced, but the scheme is only $\mathcal{O}(2)$ accurate [138].

The reduced number of evaluations of the dissipative fluxes is due to the values of β being 0 at the 2^{nd} and 4^{th} steps of the scheme, therefore only the effect of the $k-2$ previous iteration will be accounted for.

4.7 Multigrid Cycle and Interpolation

The multigrid process in *Hydra* is based on the work of [131] and falls into the *Full Approximation Storage* (*FAS*) category. The preprocessing software will perform the *edge collapsing* algorithm to generate the coarser grids as discussed in section 3.3.3. With reference to the numerical implementation of the procedure, this is performed in two steps: a *restriction* and a *prolongation* process. In the first part, solution and residual vector are interpolated from fine to coarse grids. The reason for also considering the residuals, is to allow their high-frequency modes to be damped out and to maintain the finer grid accuracy. For these reasons, a conservative approach is required, i.e. the residuals of each node are weighted by the control volume size (V) such that their value increases along with that of the cell [67]. Indicating as h and H the fine and coarse grid spacing, respectively, the restricted residuals $\mathbf{R}$ for node j can be written as:

$$\mathbf{R}_j^H = \frac{\sum_{i \in K_j} V_i^h \mathbf{R}_i^h}{max(V_j^H, \sum_{i \in K_j} V_i^h)} \tag{4.38}$$

with the summation being carried out over the set (K_j) of all the fine grid control volumes contained within the coarse one. The denominator is modified to be able to consider discrepancies between the sum of the fine mesh control volumes and the coarse mesh volume containing them, with consideration that, non-negligible effects may occur particularly at the boundaries [131]. In general, [67] states that it is better to apply $\mathcal{O}(1)$-accurate solvers on the coarse grids as they provide reduced simulation time, increased robustness and damping properties, while not affecting the finer grid solution accuracy. This is confirmed by [131], as only in certain cases did $\mathcal{O}(2)$ accuracy on the coarser levels allow to achieve convergence sooner.

The second part of the procedure is that of interpolating quantities back from coarse to fine mesh, i.e. the *prolongation*. In this case the difference between the starting and final solutions on the coarse grid are prolonged, generating a correction for the fine grid flow variables [131]:

$$\Delta\mathbf{Q}_i^h = \Delta\mathbf{Q}_j^H + (\mathbf{x}_i^h - \mathbf{x}_j^H)\nabla(\Delta\mathbf{Q}^H)_j \tag{4.39}$$

where the gradient of the solution variation on the coarse grid is computed using a GG approach and is multiplied by the distance between the fine and coarse mesh nodes, with the former belonging to set K_j.
It should be pointed out that both the interpolation processes are linear. There is, in fact, no need for an increased accuracy scheme, as according to [154], their order should fulfil:

$$(1 + \mathcal{O}_P + \mathcal{O}_R) > \mathcal{O}_E \tag{4.40}$$

i.e. the sum of the order of prolongation ($\mathcal{O}_P$) and restriction ($\mathcal{O}_R$) operators plus one should be greater than the order of the derivatives of the *p.d.e.* being modelled ($\mathcal{O}_E$) to avoid introducing errors.
To be able to further speed up the convergence process, the multigrid V- or W-cycles are generally employed (fig. 4.3). These require multiple coarsened grid levels where the flow equations are resolved. Their difference relies on whether the solution is interpolated from fine to coarse and vice-versa (V) or whether sub-multigrid iterations are used, employing only the coarse grid levels (W). In general, the former is better for super- and hypersonic flows, while the latter is more suited for transonic flows [67].

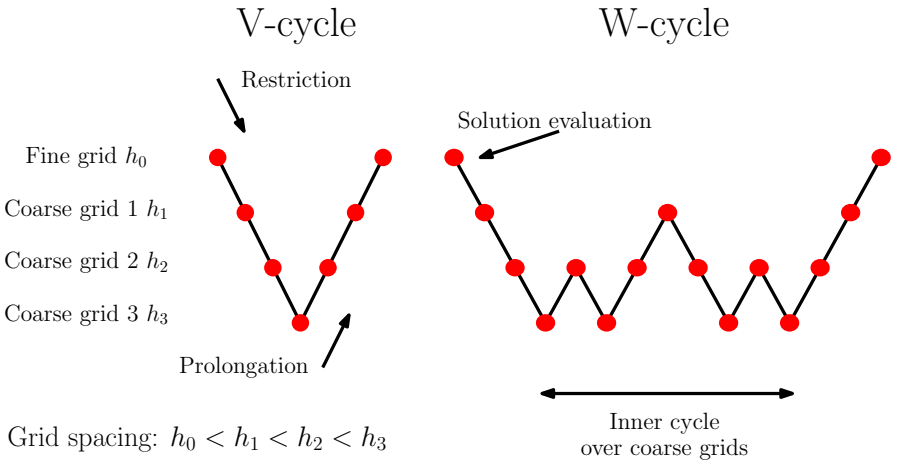


FIGURE 4.3: V- (*LHS*) and W- (*RHS*) multigrid cycles.

4.8 Turbulence Model

In this work, the *SA* turbulence model was applied to provide an approximation to the *Reynolds Stresses* in the steady, compressible *NS* conditions. The reason behind this choice is twofold: it has proven to be a more robust approach and the adjoint solver has a complete and stable linearisation of it.

This technique is based on a single transport equation to determine the *eddy viscosity* $\tilde{\nu}$ and was originally developed by [155]. [67] summarises its advantages as follows:

- good prediction of turbulent separated flows with adverse pressure gradient;
- at any one point its evaluation is independent on other nodes, simplifying the implementation;
- good robustness and convergence behaviour;
- limited near-wall grid resolution requirements.

The transport relation for this model is written as [131]:

$$\frac{\partial \tilde{\nu}}{\partial t} + \underbrace{u\frac{\partial \tilde{\nu}}{\partial x} + v\frac{\partial \tilde{\nu}}{\partial y} + w\frac{\partial \tilde{\nu}}{\partial z}}_{\text{Advection Term}} = \underbrace{\frac{1}{\sigma}\left(\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] + c_{b2}(\nabla\tilde{\nu})^2\right)}_{\text{Diffusion Term}} + \underbrace{\widehat{\mathcal{S}}}_{\text{Source Term}} \tag{4.41a}$$

$$\widehat{\mathcal{S}} = \underbrace{c_{b1}\tilde{S}\tilde{\nu}}_{\text{Production}} - \overbrace{\left(c_{w1}f_w - \frac{c_{b1}}{\widehat{\kappa}^2}f_{t2}\right)\left(\frac{\tilde{\nu}}{d_w}\right)^2}^{\text{Destruction}} + \underbrace{f_{t1}\Delta u^2}_{\text{Trip}} \tag{4.41b}$$

with the turbulent viscosity being defined as:

$$\nu_t = \tilde{\nu} f_{v1} \tag{4.42}$$

and ν is the *molecular kinetic viscosity*. The *production* and *destruction* variables are summarised in table 4.3, while those for the *trip* evaluation may be found in table 4.4.

$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}$	$\chi = \frac{\tilde{\nu}}{\nu}$	$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}$
$f_w = \widehat{g}\left(\frac{1+c_{w3}^6}{\widehat{g}^6 + c_{w3}^6}\right)^{\frac{1}{6}}$	$\widehat{g} = \widehat{r} + c_{w2}(\widehat{r}^6 - \widehat{r})$	$\widehat{r} = \frac{\tilde{\nu}}{\tilde{S}\widehat{\kappa}^2 d_w^2}$
$\tilde{S} = \widehat{S} + \frac{\tilde{\nu}}{\widehat{\kappa}^2 d_w^2} f_{v2}$	$\widehat{S} = \sqrt{\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right)^2}$	

TABLE 4.3: *SA* turbulence model *production* & *destruction* terms [131].

It should be noted that d_w and d_t are the distance of the node from the closest wall and the wall trip location, respectively, S_t is the vorticity at the trip location, Δu and Δx_t

$f_{t1} = c_{t1}\, g_t\, e^{\left(-c_{t2}\frac{S_t^2}{(\Delta u)^2}[d_w^2 + g_t^2 d_t^2]\right)}$	$f_{t2} = c_{t3}\, e^{-c_{t4}\chi^2}$	$g_t = min\left(0.1, \frac{\Delta u}{S_t \Delta x_t}\right)$

TABLE 4.4: *SA* turbulence model *trip* terms [131].

are the velocity and coordinate difference with that at trip, while the remaining terms are constants and are displayed in table 4.5. Finally, at the wall $\tilde{\nu} = 0$.

$c_{b1} = 0.1355$	$c_{b2} = 0.622$	$\sigma = \frac{2}{3}$	$c_{v1} = 7.1$
$c_{w1} = \frac{c_{b1}}{\widehat{\kappa}^2} + \frac{1+c_{b2}}{\sigma}$	$c_{w2} = 0.3$	$c_{w3} = 2$	$\widehat{\kappa} = 0.41$
$c_{t1} = 1$	$c_{t2} = 2$	$c_{t3} = 1.2$	$c_{t4} = 0.5$

TABLE 4.5: *SA* turbulence model constant terms [131].

In the code implementation, the trip contribution is not included, as for turbomachinery applications it can be assumed that the entire flow is turbulent [131]. For situations where this is not the case, $\widehat{\mathcal{S}}$ may be multiplied by a coefficient increasing linearly from 0 to 1 in the region where turbulent transition is expected to appear. Additionally, complications may arise when discretising $\nabla\tilde{\nu} \cdot \nabla\tilde{\nu}$ present in the diffusion term, as inconsistent (negative) values of the turbulent viscosity appear. For this reason the procedure in [155] was followed by [131], i.e. a conservative discretisation of this term, assuming that ν is constant, was applied:

$$\begin{aligned} &\frac{1}{\sigma}\Big[\nabla\cdot(\nu\nabla\tilde{\nu}) + \nabla\cdot(\tilde{\nu}\nabla\tilde{\nu}) + c_{b2}(\nabla\tilde{\nu})^2\Big] = \\ &\quad \frac{1}{\sigma}\nabla\Big[(\nu + (1+c_{b2})\tilde{\nu})\nabla\tilde{\nu}\Big] - \frac{c_{b2}}{\sigma}\tilde{\nu}\nabla^2\tilde{\nu} \end{aligned} \tag{4.43}$$

The convective part is in non-conservative form, due to the presence of the flow velocity term preceding of the *eddy viscosity* derivative. For this reason, the turbulence model flux discretisation is different to that of the *NS p.d.e.*s [131]:

$$\int_{V_i} \mathbf{u}\cdot\nabla\tilde{\nu}|_{x_i} dV \approx \frac{1}{2}\sum_{j\in\mathbb{E}_i}[(\mathbf{u}_i\cdot\mathbf{n}_{ij})(\tilde{\nu}_j+\tilde{\nu}_i)] - |\mathbf{u}_i\cdot\mathbf{n}_{ij}|\{(1-\phi)(L_j^{lp}(\tilde{\nu}_j) - L_i^{lp}(\tilde{\nu}_i)) + \phi(\tilde{\nu}_j - \tilde{\nu}_i)\}\Delta s_{ij} \tag{4.44}$$

where V_i is the control volume, the summation is carried out over all edges $\mathbb{E}_i$ connected to node i, L_i^{lp} is the *linearity-preserving pseudo-Laplacian* term applied to the *eddy-viscosity*, Δs_{ij} is the control surface belonging to edge ij and ϕ is the *artificial dissipation* switch. It should also be noted that the $\left|\frac{\partial\widehat{\mathcal{S}}}{\partial\nu}\right|$ term will dominate the relation

in the boundary layer regions: this is useful to aid coarser grids turbulence resolution in the multigrid scheme.

Further issues concerning negative values of turbulent viscosity appear due to the solution updating scheme. Firstly, the *SA* source term may cause the turbulence to decay too quickly, therefore the time step for node i was modified to achieve an implicit formulation [131]:

$$\overbrace{\Delta t}^{\substack{\text{Implicit}\\ \text{time step}}} = \underbrace{\frac{1}{\frac{1}{2V_i}\left(\sum_{j\in\mathbb{E}_i}|\mathbf{u}_i\cdot\mathbf{n}_{ij}|\Delta s_{ij}+\frac{2(\nu+\tilde{\nu})}{\sigma}\frac{\Delta s_{ij}}{|\mathbf{x}_j-\mathbf{x}_i|}\right)-|\frac{\partial\widehat{\mathcal{S}}}{\partial\nu}|}}_{\text{Explicit time step}} \tag{4.45}$$

with a lower limit set by the *Harten-entropy fix* proposed in [156]. It is also necessary to limit the turbulence quantities update provided by the *RK* scheme to avoid any unphysical behaviour [131]:

$$\Delta\tilde{\nu}_{\text{lim}} = \begin{cases} \Delta\tilde{\nu}, & \Delta\tilde{\nu}\leq 0, \\ \frac{(\tilde{\nu}^n-\tilde{\nu}_{\text{min}})\Delta\tilde{\nu}}{(\tilde{\nu}^n-\tilde{\nu}_{\text{min}})+\Delta\tilde{\nu}}, & \Delta\tilde{\nu}>0 \end{cases}$$

where $\tilde{\nu}_{\text{min}}=\tilde{\nu}_\infty$.

4.9 Boundary Conditions

4.9.1 No-Slip Wall

In all simulations run in this work, no-slip was the default choice for walls. This simply consists in enforcing *Dirichlet* conditions for the u,v and w velocity components: the values are zero except for cases where rotating components are present. Due to the simulation being in the relative frame of reference, surface static parts will be moving with the same velocity magnitude but in the opposite direction *w.r.t.* real configuration. Along with the primitive variables, the wall residuals must be enforced. For viscous cases utilising the *SA* model, the momentum along with the turbulence *p.d.e.s* are always set to zero. If a wall temperature is set then the energy relation's residuals are also zero.

4.9.2 Periodic Boundary

The use of *periodic boundaries* is very helpful in turbomachinery flow simulations. In fact, this allows partitioning of the computational domain into equal sections each containing a blade. It thus follows that there is a significant reduction of simulation time and the resulting file's size. As specified in [67], *Hydra* makes use of *rotational periodic boundaries*. This means that data transferred from one side to the other will require rotation of all vectorial quantities, such as the gradients or momentum terms through the following matrix:

$$\mathsf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4.46}$$

where θ is the blade's periodic angle. Therefore, for e.g., any vector quantity accumulated across the periodic boundaries would have the y and z components coupled. A final point should be made: the developments within this research, have shown the necessity to enforce equality between the values determined on each side. This means that it is preferable to rotate the quantities on one side, accumulate them with those on the other, and finally rotate them back. Alternatively, if the values are determined separately, small numerical errors can surface. Where iterative schemes are applied, repeated accumulation of these can result in significant noise or even divergence.

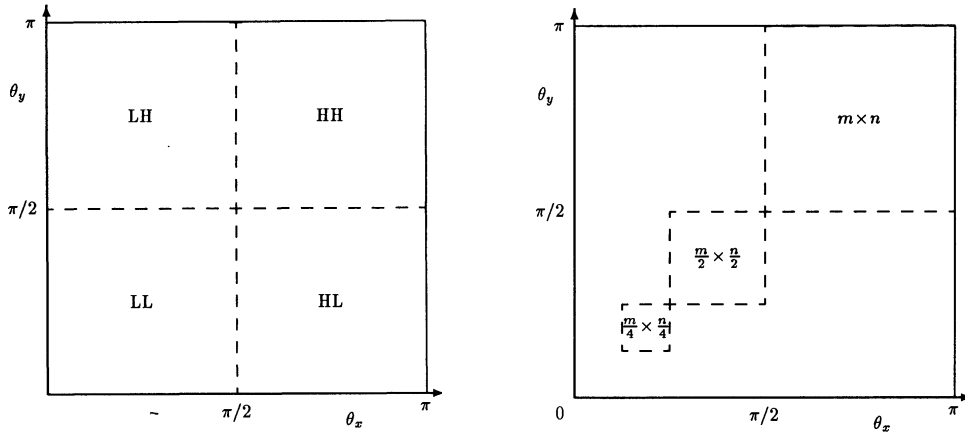
4.9.3 Symmetry Plane

Symmetrical boundary conditions are not often enforced within turbomachinery test-cases, however some clarification should follow. As in the case of periodic boundary conditions, their application relates to a reduction in the simulation domain and thus a faster turn-around time is achieved. It follows that their employment, requires a state of no flux across the symmetry surface. Assuming such a boundary is set-up in the xz plane, the following should be zero [67]:

- v-velocity component;
- y-component of the gradient of a scalar quantity;
- $\frac{\partial u}{\partial y} = \frac{\partial w}{\partial y} = 0$;
- $\frac{\partial v}{\partial x} = \frac{\partial v}{\partial z} = 0$.

4.10 Preconditioning

Despite the effort to stabilise the simulation and reduce run time through the application of multigrid techniques, convergence rate is still unsatisfactory [131]. For this reason preconditioning techniques are employed to aid solver performance. The idea is to be able to cluster eigenvalues of high frequency modes (in the *NS* residuals) as far as possible from the origin [157, 158], where *RK* approaches struggle the most [141]. In particular for an explicit time-stepping technique, the aim is relocate these modes where they may be effectively damped by the scheme itself [140]. Specifically, there are two main approaches for the implementation of such a process. The first is that proposed by *van Leer* (for e.g. see [159]), where a preconditioner matrix is created to reduce the ratio of the largest to the smallest eigenvalue of the *Euler* equations coupled with an optimal *RK* scheme. However, this approach is only suited for structured meshes with cells having an aspect ratio close to $\mathcal{O}(1)$ and is not straightforward to extend to the full *NS* relations [157]. While this procedure is developed starting from the analytic equations, other approaches are constructed by considering the discretised relations along with the scheme employed. One of such schemes is the *block-Jacobi* method as shown by [157, 160].



(A) Splitting of high (H) & low (L) 2D frequency errors in *Fourier* space.

(B) Damped 2D frequency errors in *Fourier* space by an explicit multigrid method.

FIGURE 4.4: *Fourier* domain of the error modes [141][7].

Before proceeding any further, it is important to specify the relationship between the multigrid procedure and the low/high frequency errors. In 2D, for instance, the finest grid employed will have a low ($0 \leq \theta \leq |\pm\frac{\pi}{2}|$) and a high ($|\pm\frac{\pi}{2}| \leq \theta \leq |\pm\pi|$) frequency mode in each direction. Therefore one ends up with four possible combinations of errors: high in both directions (HH), high in one and low in the other (LH, HL) and low in

[7]Reproduced with kind permission of *AIAA*.

both (LL), as shown in fig. 4.4a. The coarsening strategy generally employed in the multigrid system will produce meshes where the lower frequency errors will become high frequency and therefore may be easily dealt with by the RK method. On the other hand, the highest frequencies will be *aliased*, i.e. they will not be resolved and therefore damped as required to increase the convergence rate. This has to be the case for the three components containing at least a high frequency mode in one direction. Explicit methods, such as that utilised in *Hydra*, are problematic in handling the LH or HL components [141]. For e.g., if one considers an explicit multigrid sequence where the finest mesh is size $m \times n$ and each coarsened level is obtained by removing every odd node, the frequency spectrum covered will result in that of fig. 4.4b. This issue relates to the alignment of fluxes with the coarse mesh edges, thus causing a strong coupling in their direction, whilst generating a weak coupling normal to the edge itself. Any high frequency occurring in the latter direction will not be damped out by the multigrid procedure, potentially causing significant convergence issues [141]. This behaviour generally occurs in the tangential direction to the viscous wall, where the grid has large spacing and is usually perpendicular to the surface. The problem may also materialise in regions where convective terms of the NS relations align with the mesh. Contrary to expectation, [161] states that the discretisation of the turbulence model can be dealt with by an explicit multigrid approach without affecting convergence.

In *Hydra*, the preconditioning procedure employed is the *block-Jacobi* ([131, 140, 158, 161]), coupled with an appropriate multigrid generation procedure, and it is able to handle boundary layer cases where combinations of HL or LH modes may appear [161]. In this procedure a residual linearisation is performed *w.r.t.* the node in consideration. The block diagonal matrix is then formed by considering only terms containing the node of interest. The approach followed is that of [131], i.e. of splitting the inviscid and viscous residual terms and then re-combining them to achieve the overall matrix. The solver's inviscid flux is computed employing relation 4.21. A local linearisation yields:

$$\frac{\partial \mathbf{F}_{ij}^{I}}{\partial Q} = \frac{1}{2}\left(A_{ij}^{L} Q_i + A_{ij}^{L} Q_j - |A_{ij}|\left(\Phi(Q_i - Q_j) - \frac{1}{3}(1-\Phi)(L(Q_i) - L(Q_j)) \right) \right)$$

With A_{ij}^{L} being the linearised *Jacobian* matrix. Summing over all the edges connected to node j, as previously done for the residual accumulation (eq. 4.11):

$$(\widehat{P}_j^{I})^{-1} = \frac{1}{V_j} \sum_{i \in \mathbb{E}_j} \frac{1}{2}\left(A_{ij}^{L} Q_i + A_{ij}^{L} Q_j - |A_{ij}|\left(\Phi(Q_i - Q_j) - \frac{1}{3}(1-\Phi)(L(Q_i) - L(Q_j)) \right) \right) \Delta s_{ij}$$

Neglecting all the terms different from j:

$$(\widehat{P}_j^I)^{-1} = \frac{1}{V_j}\sum_{i\in\mathbb{E}_j}\frac{1}{2}\Big(A_{ij}^L Q_j - |A_{ij}|\Big(- Q_j\Phi + \frac{1}{3}(1-\Phi)L(Q_j)\Big)\Big)\Delta s_{ij}$$

Noting that $\sum_{i\in\mathbb{E}_j} A_{ij}^L Q_j = 0$, as it is a constant integrated over a closed volume and removing the nodal value dependencies:

$$\begin{aligned}(\widehat{P}_j^I)^{-1} &= \frac{1}{V_j}\sum_{i\in\mathbb{E}_j}\frac{1}{2}(|A_{ij}|(\Phi + \frac{1}{3}(1-\Phi)))\Delta s_{ij} \\ &= \frac{1}{V_j}\sum_{i\in\mathbb{E}_j}\frac{1}{2}|A_{ij}|(\frac{1}{3}(1+2\Phi))\Delta s_{ij} \\ &= \frac{1}{6V_j}\sum_{i\in\mathbb{E}_j}|A_{ij}|(1+2\Phi)\Delta s_{ij}\end{aligned} \tag{4.47}$$

As noted in [131], there is a difference in the value according to the switch intensity Φ, although it is argued that forcing its contribution in the preconditioner to 1 will only have the effect of reducing the local time step.
For what concerns the viscous contribution, simplifications are required. Firstly, the nonlinear terms $\mu u\frac{\partial u}{\partial x}$ are linearised by assuming constant dynamic viscosity $\bar{\mu}$ and velocity $\bar{u}$. Additionally, the gradients evaluation is approximated by a finite-difference along an edge neglecting any cross-derivatives. In particular, this my be written as:

$$\frac{\partial Q}{\partial \overrightarrow{x}} \approx \frac{Q_j - Q_i}{|\overrightarrow{x}_j - \overrightarrow{x}_i|}\overrightarrow{n}_e \tag{4.48}$$

With $\overrightarrow{n}_e$ being the unit-edge vector. The full contribution of the viscous components is determined by linearising the viscous *Jacobian* matrix (Z) [162]:

$$Z^L = \frac{\partial Z}{\partial U}\frac{\partial U}{\partial Q} \tag{4.49}$$

Where U represents the vector of *primitive* variables. This is then employed to modify the residual equation evaluation:

$$(\widehat{P}_j^V)^{-1} = \frac{1}{V_j}\sum_{i\in\mathbb{E}_j} Z^L\frac{1}{|\overrightarrow{x}_j - \overrightarrow{x_i}|}\Delta s_{ij} \tag{4.50}$$

With the edge vector having been included into the matrix Z^L. The overall preconditioner is then formed by adding the two components and multiplying by the residual's contribution at every step of the RK process. It should be clear that at $M = 0$

(stagnation points) and $M = 1$ (shocks) *Harten's* entropy fix [156] is required to avoid instabilities.

4.11 Parallelisation

The main objective behind the *OPLUS* library is to minimise the code modifications to achieve a complete parallelisation, thus simplifying development, as no knowledge of the *CPU*-to-*CPU* communication is required. Originally coded for *Reduced Instruction Set Computer* (*RISC*) architectures, it has been generalised to deal with any. The library's main characteristics concern [136]:

- the ability to deal with multiple types of data structures, such as *cell* or *edge*-based
- the increased efficiency by transferring data only when modifications have occurred and the overlap transition between computations/messages
- no requirement to have a separate serial program, thus allowing a unique version of the code simplifying development

OPLUS makes use of a generic manner of dividing the unstructured data [136]:

1. *Sets* ⇒ nodes, edges, cells ...
2. *Data on sets* ⇒ numerical values associated with the sets, such as grid coordinates for the nodes.
3. *Pointers between sets* ⇒ grid connectivity, for e.g. edge to node mapping.
4. *Operations of sets* ⇒ calculations performed on the numerical values (loops, matrix products and so on).

The domain is split between processors as shown in fig. 4.5. As it can be seen, each processor owns all data regarding the mesh portion it has received, thus each partition will execute all the operations needed on each separate group of nodes. However, an issue arises at interfaces between the domain split. Here, the developers of *OPLUS* opted for the operations on these nodes to be repeated between processors rather than having their values computed and then transferred, thus reducing the message passing burden. The actual parallelisation process divides itself into the following steps [136]:

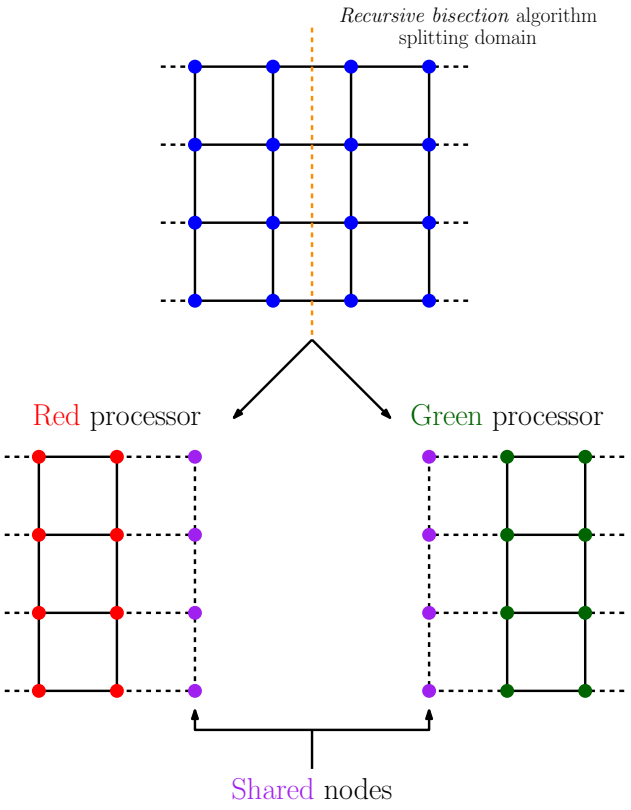


FIGURE 4.5: 2D Example of parallel domain splitting (note: colouring is only for visualisation purposes).

1. *Partitioning* ⇒ The first part consists in a *recursive bisection algorithm* followed by the application of connectivity data to split the domain.

2. *Import/export maps* ⇒ Determine identity of nodes required by current processor outside its partition (import) and of those needed by other *CPU*s (export).

3. *Local renumbering* ⇒ To reduce memory allocation the data held by the processor is all re-catalogued locally, while maintaining the global ordering to write the final solution. Additional information is then required to determine whether imported data is needed.

Chapter 5

Adjoint Solver

5.1 Introduction

Adjoint technology dates from *Lions* ([163]), who first applied it to optimal control theory in the early 70s. Shortly afterwards, *Pironneau* ([164]) employed it for aerodynamic design, then *Jameson* (late 80s to early 90s, [165–167]) managed to systematically modify a geometry to improve certain designated values of interest. Since then, research relative to adjoint techniques has become widespread, mainly related to design optimisation, but in smaller part, also to mesh adaptation[1].

In the past, the main technique for the determination of the gradient of a performance quantity of interest *w.r.t.* a design parameter perturbation, was the *FDM*. This approach did present issues in terms of numerical inaccuracy. In fact, as described by [168], the step size chosen is critical: too large and *te* terms will start to dominate, too small and subtraction inaccuracies will affect the calculation. A significant improvement to this process was the introduction of the *complex-step method*. The only difference with the *FDM* technique, is the substitution of the $\Re$-step with an $\Im$ one in the *Taylor-Series* expansion. This avoids the need for subtraction in the gradient calculation and therefore any cancellation errors [168]. *FDM*-related processes require a *CFD* run for every design parameter change, and therefore can not be considered an efficient optimisation technique, particularly in industrial settings, where time is a crucial factor of the development.

In a component design process, such as a wing's aileron or a jet engine's high-pressure compressor, the engineer is concerned with discerning how critical data, for e.g. lift or drag on the aerofoil surface, may be optimised *w.r.t.* the design space available, such as camber or chord. It is more common than not, for the performance quantities to be

[1] It should be noted that in other areas of research, adjoint procedures are employed for other purposes, such as data fitting.

fewer in number *w.r.t.* the design variables. Therefore, in an optimisation problem it will be more efficient to compute the sensitivity of performance values to the design space based on perturbations in the quantity of interest, rather than perturbing each design variable independently. In other words, to optimise a system with a smaller number of outputs than inputs, it is cheaper to determine the sensitivity of outputs to inputs by perturbing a selected output and tracing the effect in *reverse* (or *adjoint mode*) back to the inputs, rather than varying a single input to see how it affects all the outputs in a *forward* manner (or *direct mode*), as shown in figure 5.1.

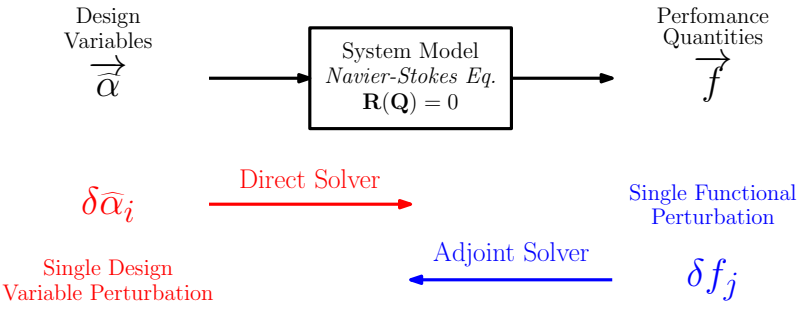


FIGURE 5.1: Direct and adjoint solver schematic.

Both methods are based on a linearisation of the *p.d.e.* modelling the system (in this case the *NS* relations). Depending at which stage the linearisation takes place, the resulting sensitivity equations may fall into the *continuous* or *discrete* category. In the first case, the analytical *NS* eq. are linearised and equations for the functional of interest derived. These are then discretised, without necessarily employing the same strategy as that chosen for the flow solver. In general this approach is the least preferred, as for every performance quantity of interest new equations must be determined. Additionally, the boundary conditions derivation is non-intuitive and, in some cases, non-physical [168].
On the other hand, *discrete* implementations consist of a linearisation of the discretised flow solver. This means that both must have the same underlying procedure of solving the equations. One of the most complicated features of this method relates to code *consistency* between *NS* solver and the direct/adjoint codes. By consistency it is meant that as the grid resolution is increased, the *NS* and direct/adjoint solvers will tend towards the exact solution. To achieve this condition it is not uncommon for the *NS* code to be modified to achieve consistency of the linearised codes. Implementations of the direct or adjoint solvers may be simplified by applying *algorithmic* or *automatic differentiation* (*AD*) of the source code, thus significantly reducing the developer's task. On the other hand, though, they can be slower *w.r.t.* their continuous counterparts.
In the sections that follow the derivation of adjoint relations, together with a description of other main aspects of it will be provided. After the general treatment of the topic, a

more in-depth analysis related to the actual implementation of the discrete version will be carried out.

5.2 General Adjoint Equations

As discussed by [168], every functional of interest ($f_j \;\; j = 1, .., N_f$)[2] in an optimisation problem, has a dependency on the system's state ($\mathbf{Q}$) and N_D design variables ($\widehat{\alpha}_i \;\; i = 1, .., N_D$). Therefore, in a generic way it is possible to write:

$$f_j = f_j(\widehat{\alpha}_i, \mathbf{Q}(\widehat{\alpha}_i)) \tag{5.1}$$

Indicating the system's equations, in this case the *NS* relations, as $\mathbf{R}(\mathbf{Q}) = 0$:

$$\mathbf{R}(\widehat{\alpha}_i, \mathbf{Q}(\widehat{\alpha}_i)) = 0 \quad \forall\, \widehat{\alpha}_i \tag{5.2}$$

As can be seen, the state variables depend implicitly on $\widehat{\alpha}_i$. The first step in the direct and adjoint procedures, is determining the total derivative of the functional of interest *w.r.t.* the design variables [168]:

$$\frac{df_j}{d\widehat{\alpha}_i} = \frac{\partial f_j}{\partial \widehat{\alpha}_i} + \frac{\partial f_j}{\partial \mathbf{Q}} \frac{d\mathbf{Q}}{d\widehat{\alpha}_i} \tag{5.3}$$

While the partial derivatives can be simply evaluated perturbing the denominator and determining how the numerator varies, the total derivatives, specifically the $\frac{d\mathbf{Q}}{d\widehat{\alpha}_i}$ term, are more complex to evaluate. At this point, one can introduce the total derivative of the *NS* relations *w.r.t.* the design parameters, as this will be 0 regardless (eq. 5.2):

$$\frac{d\mathbf{R}}{d\widehat{\alpha}_i} = \frac{\partial \mathbf{R}}{\partial \widehat{\alpha}_i} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \frac{d\mathbf{Q}}{d\widehat{\alpha}_i} = 0 \tag{5.4}$$

Noting that the problematic term is present in both relations:

$$\frac{df_j}{d\widehat{\alpha}_i} = \frac{\partial f_j}{\partial \widehat{\alpha}_i} + \frac{\partial f_j}{\partial \mathbf{Q}} \frac{d\mathbf{Q}}{d\widehat{\alpha}_i} \xrightarrow{\frac{d\mathbf{Q}}{d\widehat{\alpha}_i} = -\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^{-1} \frac{\partial \mathbf{R}}{\partial \widehat{\alpha}_i}} \frac{df_j}{d\widehat{\alpha}_i} = \frac{\partial f_j}{\partial \widehat{\alpha}_i} - \frac{\partial f_j}{\partial \mathbf{Q}} \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^{-1} \frac{\partial \mathbf{R}}{\partial \widehat{\alpha}_i} \tag{5.5}$$

At this point, it is possible to differentiate between direct and adjoint methods. This is determined by which combination of quantities present in the last term are employed.

[2] These may also be indicated as scalar integral quantities of interest.

If $\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^{-1}\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}_i}$ is solved, the direct, or *tangent* solver, has been chosen. In this case, for every different design parameter, a new solution would be required. However, the equations have no dependence on the functional of interest. Therefore, this procedure would efficiently determine the sensitivity for all quantities to be optimised *w.r.t.* one design variable.
On the other hand, opting to solve the term formed by $\frac{\partial f_j}{\partial \mathbf{Q}}\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^{-1}$, the opposite is true: i.e. a single solution to this relation can be used to determine the sensitivity of a parameter of interest *w.r.t.* any design variable. This can be easily verified by noting the direct dependence on the functional of interest, but no presence of the design variables.
At this point it is possible to define the adjoint vector $\mathbf{\Psi}$, utilising the reasoning that has just be outlined:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\mathbf{\Psi}^T = -\frac{\partial f}{\partial \mathbf{Q}} \tag{5.6}$$

5.3 Duality: An Alternative Viewpoint

Often, the direct and adjoint relations are cast in a more generic framework, see for e.g. [14, 169–171]. As this formulation is of interest to clarify certain aspects of the adjoint methodology, it will be briefly discussed here as well.
Considering a perturbation $\delta\mathbf{Q}$ in the flow solution $\mathbf{Q}$, it possible to linearise the dependency of a quantity of interest f_j (e.g. lift) on $\mathbf{Q}$ and then determine its change due to the perturbation:

$$\delta f_j = \frac{\partial f_j}{\partial \mathbf{Q}}\delta\mathbf{Q} = g^T\widehat{u} \tag{5.7}$$

With $\widehat{u}$ being the perturbation and g^T the gradient. Linearising the flow relations, the *primal* system of equations may be determined [170]:

$$\widehat{L}\widehat{u} = \widehat{p} \tag{5.8}$$

The *dual* form consists in computing $\widehat{v}^T\widehat{p}$, where $\widehat{v}$ corresponds to the adjoint vector and is evaluated as follows:

$$\widehat{L}^T\widehat{v} = g \tag{5.9}$$

It is now necessary to prove the identity $\widehat{v}^T\widehat{p} = g^T\widehat{u}$:

$$\widehat{v}^T\widehat{p} = \widehat{v}^T\widehat{L}\widehat{u} = (\widehat{L}^T\widehat{v})^T\widehat{u} = g^T\widehat{u} \tag{5.10}$$

From a physical point of view, it can be seen that the adjoint term $\widehat{v}$ indicates the effect of changes of the source term $\widehat{p}$ on the quantity of interest.

5.4 Continuous vs. Discrete Adjoint

As previously mentioned, the continuous and discrete adjoint difference is due to the order in which they are derived *w.r.t.* the discretisation, as illustrated in fig. 5.2.

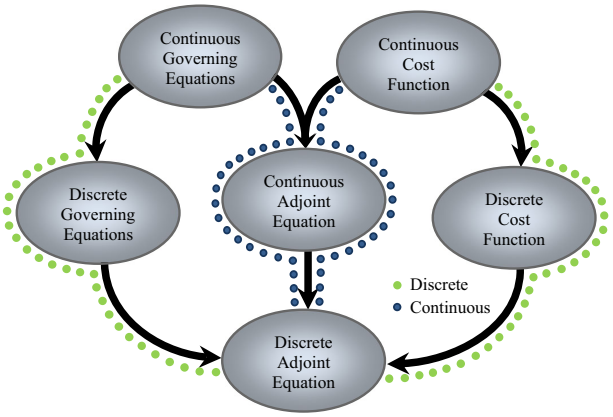


FIGURE 5.2: Continuous & discrete adjoint derivation process [172][3].

In the first case, the flow equations are linearised and applied to determine the functional of interest through the use of a *Lagrangian* formulation, for example. The dual analytical equations along with the boundary conditions are thus determined and may be discretised with an appropriate scheme. It should be clear that the procedure employed may not necessarily be that of the flow solver. This allows a greater degree of freedom that results in a computationally more efficient code. Moreover, the final solution will determine the desired gradient, i.e. the sensitivity of the functional *w.r.t.* the design parameters, without the need of further steps. Finally, it does offer better physical insight due to the direct relation with *Green's functions* [170]. However, great part of the derivation has to be repeated for every different functional or for changes in the location where they are evaluated (for e.g. in a full aircraft configuration it may be necessary to compute drag on the wing and the horizontal tail-plane separately).
On the other hand, in the discrete adjoint approach the discretised nonlinear *NS* equations are used. In this case, the first step consists in differentiating, i.e. linearising, the flow solver's code. This may be done by hand or automatically through *AD* software

[3]Reproduced with kind permission of *Springer*.

Continuous		**Discrete**	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
Execution Speed	Determination of B.C.s	Use of *AD*	Mesh Adjoint Evaluation
Physically Intuitive	New Discretisation for each Functional/-Surface of Integration	Maintainability	Computation of *Jacobian* Matrix at each Step
Flexible Implementation	Code Debugging	Direct & Adjoint Solvers Identical Rate of Convergence	Linearisation Inconsistencies of Flow Solver Discretised Code
No Mesh Adjoint	Code Maintenance	Primal & Dual Consistency Helps Debugging	Complexity of Initial Development
Linear Code Independent of Flow Solver	Validation	Widespread use in Mesh Adaptation	Inefficient Code

TABLE 5.1: Continuous vs discrete adjoint: advantages & disadvantages

packages. In the first case, the developer will have to go through the code line by line and determine the necessary instructions. As one can imagine this process can easily be subject to error, hence extremely time consuming. Moreover, any change to the nonlinear flow solver must be propagated to the linearised codes as well. On the plus side, this allows a faster code *w.r.t.* the *AD* way of deriving it. On the other hand, *AD* provides the easiest way of maintaining the differentiated code. In fact, during compilation time, the code will be differentiated automatically in *forward* or direct way and in *reverse* or adjoint manner, thus being able to generate the source code for both solvers in a single action. Therefore, any change in the nonlinear code will be immediately disseminated to the linearised counterparts. One of the main complications that may occur, is represented by the code *consistency*, i.e. changes to the nonlinear flow solver may be required to ensure that as the grid spacing approaches 0 they both tend to the analytical answer. This issue is avoided in the continuous counterpart, as the solvers are separate. Finally, the discrete code will have the advantage of having its boundary conditions determined automatically, whilst in the other case dealing with them is non-trivial.
A summary of the main variations between discrete and continuous adjoint versions is given in table 5.1.

5.5 Continuous Adjoint: Boundary Conditions, Time & Convection Reversal Example

Adjoint techniques are well known for reversing the flow convection direction, but also the pseudo- and physical- time stepping. One of the most common features of this characteristic is the so-called *reversed-wake* as shown on the *RHS* of fig. 5.3. It is clear from the comparison with the *NS* wake (*LHS* in fig. 5.3) that they propagate in opposite directions.

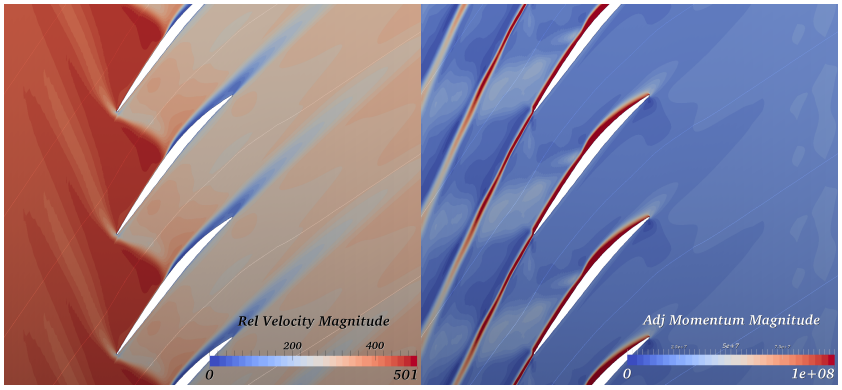


FIGURE 5.3: Comparison of flow velocity magnitude (*LHS*) with that of adjoint momentum equations (of mass averaged *adiabatic efficiency* between inlet/outlet, *RHS*) at mid-span for *NASA Rotor* 37.

This behaviour of the dual solution is effectively indicating that a perturbation in the reversed-wake region is going to have a greater effect on the functional of interest (*adiabatic efficiency* in the case shown in fig. 5.3) than anywhere else in the domain. To understand the reason behind this behaviour, the simplest explanation employs the continuous adjoint operator definition as defined by [173]:

$$\int_{\Omega} \widehat{v}\widehat{L}\widehat{u}\, d\Omega = [...]\big|_{\partial\Omega} + \int_{\Omega} \widehat{u}\widehat{L}^{*}\widehat{v}\, d\Omega \tag{5.11}$$

Where integration by parts has been employed. Considering the $1D$ *linear convection-diffusion* equation [170], with constant velocity $\widehat{a}$:

$$\widehat{L}\widehat{u} = \frac{d\widehat{u}}{dx} - \widehat{a}\frac{d^2\widehat{u}}{dx^2} \quad 0 < x < 1 \tag{5.12}$$

Subject to the boundary conditions $\widehat{u}(0) = \widehat{u}(1) = 0$, one may determine the *dual* or adjoint formulation as:

$$
\begin{aligned}
(\widehat{v}, \widehat{L}\widehat{u}) &= \int_0^1 \widehat{v}\left(\frac{d\widehat{u}}{dx} - \widehat{a}\frac{d^2\widehat{u}}{dx^2}\right)dx \\
&= \int_0^1 \widehat{v}\frac{d\widehat{u}}{dx}dx - \int_0^1 \widehat{v}\widehat{a}\frac{d^2\widehat{u}}{dx^2}dx \\
&= \cancelto{0}{\widehat{u}\widehat{v}\Big|_0^1} - \int_0^1 \widehat{u}\frac{d\widehat{v}}{dx}dx - \widehat{a}\widehat{v}\frac{d\widehat{u}}{dx}\Big|_0^1 + \int_0^1 \widehat{a}\frac{d\widehat{v}}{dx}\frac{d\widehat{u}}{dx}dx \\
&= -\int_0^1 \widehat{u}\frac{d\widehat{v}}{dx}dx - \left[\widehat{a}\widehat{v}\frac{d\widehat{u}}{dx} - \cancelto{0}{\widehat{a}\widehat{u}\frac{d\widehat{v}}{dx}}\right]_0^1 - \int_0^1 \widehat{u}\widehat{a}\frac{d^2\widehat{v}}{dx^2}dx \\
&= -\int_0^1 \widehat{u}\left[\frac{d\widehat{v}}{dx} + \widehat{a}\frac{d^2\widehat{v}}{dx^2}\right]dx - \widehat{a}\widehat{v}\frac{d\widehat{u}}{dx}\Big|_0^1 = (\widehat{L}^*\widehat{v}, \widehat{u})
\end{aligned}
\tag{5.13}
$$

The adjoint operator $\widehat{L}^*$ can therefore be expressed as:

$$
\widehat{L}^* = -\frac{d}{dx} - \widehat{a}\frac{d^2}{dx^2} \tag{5.14}
$$

With adjoint boundary conditions:

$$
\widehat{a}\left[\widehat{v}(0)\frac{d\widehat{u}}{dx}(0) - \widehat{v}(1)\frac{d\widehat{u}}{dx}(1)\right] = 0 \tag{5.15}
$$

Which allow the adjoint *p.d.e.* to be *homogeneous*. By enforcing $\widehat{v}(0) = \widehat{v}(1) = 0$ the boundary conditions are fulfilled.

By comparing the primal operator $\widehat{L} = \frac{d}{dx} - \widehat{a}\frac{d^2}{dx^2}$ with the *dual* one, it is clear that odd derivatives will have the sign inverted, unlike their even counterparts that will maintain their direction. This simple example has allowed to demonstrate the mathematical reason behind the reversal in space and time of the adjoint solver *w.r.t.* the original *p.d.e.*. In fact, considering the full *NS* equations, the reader will easily recall that the time and convective terms space derivatives are all first order, thus being reversed by the adjoint operator.

From an intuitive point of view, the reason behind directionality reversal is trivial to understand. In fact, considering a typical subsonic *CFD* simulation, the adjoint solution is indicating that the highest sensitivity is located in the upstream part of the domain, as values in these regions are then convected downstream. Therefore, in general, perturbing a quantity at the inlet will have a greater effect on the functional than applying this to the outlet.

5.6 Explicit Runge-Kutta Scheme with Preconditioning

As previously mentioned, the continuous adjoint operator can be determined from the primal one by applying integration by parts. The procedure was shown for the linear diffusion equation in section 5.5 and it has been proven that any odd derivative has its sign reversed. This results in time and convection having opposite direction in the linearised *NS dual* relations. The discrete counterpart of integration by parts is *summation by parts* [174]:

$$\sum_{i=0}^{n-1} a^{i+1}(b^{i+1} - b^i) = a^n b^n - a^0 b^0 - \sum_{i=0}^{n-1}(a^{i+1} - a^i)b^i \tag{5.16}$$

Where a and b are two generic quantities. This equation will be used to show how an explicit *RK* scheme is changed to be employed appropriately in a steady discrete adjoint solver[4]. To be able to determine how this time-marching scheme should be used for the dual solver, [174] first determines the duality equivalence introducing extra terms into the continuous formulation[5]:

$$\begin{aligned}
&(g,\widehat{u}) - \underbrace{(\widehat{v},\widehat{L}\widehat{u}-\widehat{p})}_{\widehat{L}\widehat{u}=\widehat{p}\,\therefore\,\widehat{L}\widehat{u}-\widehat{p}=0} = \\
&(g,\widehat{u}) - (\widehat{v},\widehat{L}\widehat{u}) + (\widehat{v},\widehat{p}) = \\
&(g,\widehat{u}) - (\widehat{L}^*\widehat{v},\widehat{u}) + (\widehat{v},\widehat{p}) = \\
&-(\widehat{L}^*\widehat{v} - g,\widehat{u}) + (\widehat{v},\widehat{p}) = \\
&(\widehat{v},\widehat{p}) - (\widehat{L}^*\widehat{v} - g,\widehat{u})
\end{aligned} \tag{5.17}$$

Therefore to fulfil the duality requirement $\widehat{L}^*\widehat{v} - g = 0$.
The discrete unsteady primal problem may be formulated as:

$$\widehat{u}^{n+1} = \widehat{u}^n + \widehat{R}(\widehat{p} - \widehat{L}\widehat{u}) \tag{5.18}$$

Where $\widehat{R}$ is an operator defined by the *RK* scheme and the preconditioning employed. The functional is thus computed at the last time step N_t, such that $f = (g, \widehat{u}^{N_t})$ with $\widehat{u}^0 = 0$. Employing the technique previously outlined in 5.17 and setting $\widehat{\omega}$ as the unsteady adjoint variable, the last relation may be expanded as:

[4]This derivation follows the ideas of [174], unlike the publication, in this case a detailed explanation of every step is provided to aid the reader's understanding.

[5]This is effectively the *Lagrangian* formulation with linear algebra notation.

$$
\begin{aligned}
f =&(g, \widehat{u}^{N_t}) - \sum_{n=0}^{N_t-1} (\widehat{\omega}^{n+1}, \widehat{u}^{n+1} - \widehat{u}^{n} - \widehat{R}(\widehat{p} - \widehat{L}\widehat{u}^{n})) \\
=&(g, \widehat{u}^{N_t}) - \sum_{n=0}^{N_t-1} (\widehat{\omega}^{n+1}, \widehat{u}^{n+1} - \widehat{u}^{n}) - \sum_{n=0}^{N_t-1} (\widehat{\omega}^{n+1}, -\widehat{R}(\widehat{p} - \widehat{L}\widehat{u}^{n})) \xRightarrow[\text{by parts}]{\text{summation}} \\
=&(g, \widehat{u}^{N_t}) - (\widehat{\omega}^{N_t}\widehat{u}^{N_t} - \cancelto{0}{\widehat{\omega}^{0}\widehat{u}^{0}}) - \sum_{n=0}^{N_t-1} (-\widehat{\omega}^{n+1} + \widehat{\omega}^{n}, \widehat{u}^{n}) - \\
&- \sum_{n=0}^{N_t-1} (\widehat{R}^{T}\widehat{\omega}^{n+1}, -\widehat{p}) - \sum_{n=0}^{N_t-1} (\widehat{L}^{T}\widehat{R}^{T}\widehat{\omega}^{n+1}, \widehat{u}^{n}) \\
=&(g, \widehat{u}^{N_t}) - (\widehat{\omega}^{N_t}, \widehat{u}^{N_t}) - \sum_{n=0}^{N_t-1} (-\widehat{\omega}^{n+1} + \widehat{\omega}^{n} + \widehat{L}^{T}\widehat{R}^{T}\widehat{\omega}^{n+1}, \widehat{u}^{n}) + \sum_{n=0}^{N_t-1} (\widehat{R}^{T}\widehat{\omega}^{n+1}, \widehat{p}) \\
=&(g - \widehat{\omega}^{N_t}, \widehat{u}^{N_t}) + \sum_{n=0}^{N_t-1} (\widehat{\omega}^{n+1} - \widehat{\omega}^{n} - \widehat{L}^{T}\widehat{R}^{T}\widehat{\omega}^{n+1}, \widehat{u}^{n}) + \sum_{n=0}^{N_t-1} (\widehat{R}^{T}\widehat{\omega}^{n+1}, \widehat{p})
\end{aligned}
\tag{5.19}
$$

For this relation to have the same form as 5.17, the following must hold[6]:

$$
\widehat{\omega}^{n} = \widehat{\omega}^{n+1} - \widehat{L}^{T}\widehat{R}^{T}\widehat{\omega}^{n+1} \tag{5.20}
$$

along with the adjoint boundary condition $\widehat{\omega}^{N_t} = g$. Employing these, the objective becomes:

$$
f = \sum_{n=0}^{N_t-1} (\widehat{R}^{T}\widehat{\omega}^{n+1}, \widehat{p}) \tag{5.21}
$$

To be able to link the unsteady adjoint to the steady version, $\widehat{v}^{n}$ is defined as:

$$
\widehat{v}^{n} = \sum_{m=n}^{N_t-1} \widehat{R}^{T}\widehat{\omega}^{m+1} \qquad \forall\, n < N_t \tag{5.22}
$$

with $\widehat{v}^{N_t} = 0$ so that the objective function may be written as:

$$
f = (\widehat{v}^{0}, \widehat{p}) \tag{5.23}
$$

Finally, it is possible to remove the dependency of $\widehat{v}$ on $\widehat{\omega}$ and determine its full equation:

[6]Time reversal has also occurred.

$$\begin{cases} \widehat{v}^n = \sum_{m=n}^{N_t-1} \widehat{R}^T \widehat{\omega}^{m+1} \\ \widehat{v}^{n+1} = \sum_{m=n+1}^{N_t-1} \widehat{R}^T \widehat{\omega}^{m+1} \end{cases} \quad \therefore \widehat{v}^n - \widehat{v}^{n+1} = \widehat{R}^T \widehat{\omega}^{n+1} \tag{5.24}$$

Consequently:

$$\begin{aligned} \widehat{v}^n - \widehat{v}^{n+1} &= \widehat{R}^T \widehat{\omega}^{n+1} \xrightarrow{\sum_{m=n+1}^{N_t-1} \widehat{\omega}^{m+1} - \widehat{\omega}^m = \widehat{\omega}^{N_t} - \widehat{\omega}^{n+1}} \\ &= \widehat{R}^T \left(\widehat{\omega}^{N_t} - \sum_{m=n+1}^{N_t-1} (\widehat{\omega}^{m+1} - \widehat{\omega}^m) \right) \xrightarrow{\widehat{\omega}^{N_t} = g} \\ &= \widehat{R}^T \left(g - \sum_{m=n+1}^{N_t-1} (\widehat{\omega}^{m+1} - \widehat{\omega}^m) \right) \xrightarrow{\widehat{\omega}^{n+1} - \widehat{\omega}^n = \widehat{L}^T \widehat{R}^T \widehat{\omega}^{n+1}} \\ &= \widehat{R}^T \left(g - \sum_{m=n+1}^{N_t-1} (\widehat{L}^T \widehat{R}^T \widehat{\omega}^{m+1}) \right) \xrightarrow{\widehat{v}^n = \sum_{m=n}^{N_t-1} \widehat{R}^T \widehat{\omega}^{m+1}} \\ &= \widehat{R}^T (g - \widehat{L}^T \widehat{v}^{n+1}) \end{aligned} \tag{5.25}$$

As the matrices $\widehat{R}$ and $\widehat{L}$ are only transposed *w.r.t.* the primal equation, the same rate of convergence may be expected.
Having gone through the general idea behind the dual solver's time-stepping approach, it is possible to derive the detailed relations for the *RK* scheme [174]. The linearised *NS* equations may be split between the dissipative $\widehat{D}$ and convective $\widehat{C}$ operators:

$$\widehat{L}\widehat{u} = \widehat{C}\widehat{u} + \widehat{D}\widehat{u} \tag{5.26}$$

As is the case in a partial updating *RK* explicit time stepping scheme. A generic version of such schemes may be written as:

$$\begin{aligned} &\widehat{d}^0 = 0 \\ &\widehat{u}^0 = \widehat{u}^n \\ &\begin{cases} \widehat{d}^m = & \beta_m \widehat{D}\widehat{u}^{m-1} + (1-\beta_m)\widehat{d}^{m-1} \\ \widehat{u}^m = & \widehat{u}^0 + \alpha_m \widehat{P}(\widehat{p} - \widehat{C}\widehat{u}^{m-1} - \widehat{d}^m) \end{cases} \quad m = 1 \text{ to } \widehat{M} \text{ stage } RK \\ &\widehat{u}^{n+1} = \widehat{u}^{\widehat{M}} \end{aligned} \tag{5.27}$$

Where α_m and β_m are the $\widehat{M}$-stage *RK* coefficients while $\widehat{P}$ is the *block-Jacobi* preconditioning matrix and $\widehat{d}$ is the dissipative contribution. [174] then defines the residuals

of the primal solver as $r = \widehat{p} - \widehat{L}\widehat{u}$ and perturbations to the dissipative and convective terms as:

$$\widetilde{d}^m = \widehat{d}^m - \widehat{D}\widehat{u}^n \;\therefore\; \widehat{d}^m = \widetilde{d}^m + \widehat{D}\widehat{u}^n \tag{5.28a}$$

$$\widetilde{u}^m = \widehat{u}^m - \widehat{u}^n \;\therefore\; \widehat{u}^m = \widetilde{u}^m + \widehat{u}^n \tag{5.28b}$$

Effectively, these represent the updates to the quantities provided by the *RK* scheme. In other words they are system perturbations in forward mode. Employing these in a generic *RK* scheme:

$$\begin{aligned}
&\widetilde{d}^0 = \cancelto{0}{\widehat{d}^0} - \widehat{D}\widehat{u}^n = -\widehat{D}\widehat{u}^n \\
&\widetilde{u}^0 = \widehat{u}^0 - \widehat{u}^n = 0 \\
&\begin{cases}
\widetilde{d}^m = \beta_m \widehat{D}(\widetilde{u}^{m-1} + \widehat{u}^n) + (1-\beta_m)(\widetilde{d}^{m-1} + \widehat{D}\widehat{u}^n) - \widehat{D}\widehat{u}^n = \\
\quad = \beta_m \widehat{D}\widetilde{u}^{m-1} + \cancel{\beta_m \widehat{D}\widehat{u}^n} + \widetilde{d}^{m-1} + \cancel{\widehat{D}\widehat{u}^n} - \beta_m \widetilde{d}^{m-1} - \cancel{\beta_m \widehat{D}\widehat{u}^n} - \cancel{\widehat{D}\widehat{u}^n} = \\
\quad = \beta_m \widehat{D}\widetilde{u}^{m-1} + (1-\beta_m)\widetilde{d}^{m-1} \\
\widetilde{u}^m = \cancel{\widehat{u}^0} + \alpha_m \widehat{P}(\widehat{p} - \widehat{C}\widetilde{u}^{m-1} - \widehat{C}\widehat{u}^n - \widetilde{d}^m - \widehat{D}\widehat{u}^n) - \cancel{\widehat{u}^n} = \\
\quad = \alpha_m \widehat{P}(-\widehat{C}\widetilde{u}^{m-1} - \widetilde{d}^m + \underbrace{\widehat{p} - (\widehat{C}\widehat{u}^n + \widehat{D}\widehat{u}^n)}_{=r}) = \\
\quad = \alpha_m \widehat{P}(r - \widehat{C}\widetilde{u}^{m-1} - \widetilde{d}^m)
\end{cases} \\
&\widehat{u}^{n+1} = \widehat{u}^n + \widetilde{u}^{\widehat{M}}
\end{aligned} \tag{5.29}$$

As it can be easily seen, the convective term update is equivalent to the form $\widehat{u}^{n+1} = \widehat{u}^n + \widehat{R}(\widehat{p} - \widehat{L}\widehat{u})$. Therefore:

$$\begin{aligned}
\widetilde{u}^m &= \alpha_m \widehat{P}(r - \widehat{C}\widetilde{u}^{m-1} - \widetilde{d}^m) \\
&= \alpha_m \widehat{P} r - \alpha_m \widehat{P}(\widehat{C}\widetilde{u}^{m-1} - \widetilde{d}^m) \;\therefore \\
\widetilde{u}^m &+ \alpha_m \widehat{P}(\widehat{C}\widetilde{u}^{m-1} + \widetilde{d}^m) = \alpha_m \widehat{P} r
\end{aligned} \tag{5.30}$$

By employing this relationship and that for the perturbed diffusive terms, it is possible to write the system as:

$$\underbrace{\begin{bmatrix} I & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -\beta_2\widehat{D} & I & 0 & \cdots & \cdots & \cdots & 0 \\ \alpha_2\widehat{P}\widehat{C} & \alpha_2\widehat{P} & I & 0 & \cdots & \cdots & 0 \\ 0 & -1+\beta_3 & -\beta_3\widehat{D} & I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1+\beta_{\widehat{M}} & -\beta_{\widehat{M}}\widehat{D} & I & 0 \\ 0 & \cdots & \cdots & 0 & \alpha_{\widehat{M}}\widehat{P}\widehat{C} & \alpha_{\widehat{M}}\widehat{P} & I \end{bmatrix}}_{\widehat{B}} \cdot \begin{bmatrix} \widetilde{u}^1 \\ \widetilde{d}^2 \\ \widetilde{u}^2 \\ \widetilde{d}^3 \\ \cdots \\ \widetilde{d}^{\widehat{M}} \\ \widetilde{u}^{\widehat{M}} \end{bmatrix} = \begin{bmatrix} \alpha_1\widehat{P} \\ 0 \\ \alpha_2\widehat{P} \\ 0 \\ \cdots \\ 0 \\ \alpha_{\widehat{M}}\widehat{P} \end{bmatrix} \cdot r \tag{5.31}$$

with I being the *identity matrix*. The system has now been cast as the preconditioned primal residuals weighted by the respective α_m. To determine the forward *RK* operator $\widehat{R}$:

$$\begin{bmatrix} \widetilde{u}^1 \\ \widetilde{d}^2 \\ \widetilde{u}^2 \\ \widetilde{d}^3 \\ \cdots \\ \widetilde{d}^{\widehat{M}} \\ \widetilde{u}^{\widehat{M}} \end{bmatrix} = \widehat{B}^{-1} \cdot \begin{bmatrix} \alpha_1\widehat{P} \\ 0 \\ \alpha_2\widehat{P} \\ 0 \\ \cdots \\ 0 \\ \alpha_{\widehat{M}}\widehat{P} \end{bmatrix} \cdot r \tag{5.32}$$

the *RK* full update is the last term in the *LHS* vector:

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & I \end{bmatrix} \cdot \begin{bmatrix} \widetilde{u}^1 \\ \widetilde{d}^2 \\ \widetilde{u}^2 \\ \widetilde{d}^3 \\ \cdots \\ \widetilde{d}^{\widehat{M}} \\ \widetilde{u}^{\widehat{M}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & I \end{bmatrix} \cdot \widehat{B}^{-1} \cdot \begin{bmatrix} \alpha_1\widehat{P} \\ 0 \\ \alpha_2\widehat{P} \\ 0 \\ \cdots \\ 0 \\ \alpha_{\widehat{M}}\widehat{P} \end{bmatrix} \cdot r \tag{5.33}$$

therefore:

$$\widehat{R} = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 & I \end{bmatrix} \cdot \widehat{B}^{-1} \cdot \begin{bmatrix} \alpha_1 \widehat{P} \\ 0 \\ \alpha_2 \widehat{P} \\ 0 \\ \cdots \\ 0 \\ \alpha_{\widehat{M}} \widehat{P} \end{bmatrix} \tag{5.34}$$

As normal, transposing the primal operator produces the adjoint one:

$$\widehat{R}^T = \begin{bmatrix} \alpha_1 \widehat{P}^T & 0 & \alpha_2 \widehat{P}^T & 0 & \cdots & 0 & \alpha_{\widehat{M}} \widehat{P}^T \end{bmatrix} \cdot (\widehat{B}^T)^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ 0 \\ I \end{bmatrix} \tag{5.35}$$

Setting $\widetilde{\omega}$ as the dual RK working variable and recalling the time reversal, it is possible to write $\widehat{v}^n$ as:

$$\widehat{v}^n = \widehat{v}^{n+1} + \sum_{m=1}^{\widehat{M}} \alpha_m \widehat{P}^T \widetilde{\omega}^m \;\; \therefore \;\; \widehat{v}^n - \widehat{v}^{n+1} = \sum_{m=1}^{\widehat{M}} \alpha_m \widehat{P}^T \widetilde{\omega}^m \tag{5.36}$$

that is, the updated adjoint variable is equal to the update accumulated onto the next value. Recalling equation 5.25, the last becomes:

$$\sum_{m=1}^{\widehat{M}} \alpha_m \widehat{P}^T \widetilde{\omega}^m = \widehat{R}^T (g - \widehat{L}^T \widehat{v}^{n+1}) \tag{5.37}$$

substituting the dual RK operator as derived in relation 5.35:

$$\begin{bmatrix} \alpha_1\widehat{P}^T & 0 & \alpha_2\widehat{P}^T & 0 & \cdots & 0 & \alpha_{\widehat{M}}\widehat{P}^T \end{bmatrix} \cdot \begin{bmatrix} \widetilde{\omega}^1 \\ \widetilde{d}^2 \\ \widetilde{\omega}^2 \\ \widetilde{d}^3 \\ \cdots \\ \widetilde{d}^{\widehat{M}} \\ \widetilde{\omega}^{\widehat{M}} \end{bmatrix} = \tag{5.38}$$

$$\begin{bmatrix} \alpha_1\widehat{P}^T & 0 & \alpha_2\widehat{P}^T & 0 & \cdots & 0 & \alpha_{\widehat{M}}\widehat{P}^T \end{bmatrix} \cdot (\widehat{B}^T)^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ 0 \\ I \end{bmatrix} \cdot (g - \widehat{L}^T\widehat{v}^{n+1})$$

re-arranging:

$$\widehat{B}^T \begin{bmatrix} \widetilde{\omega}^1 \\ \widetilde{d}^2 \\ \widetilde{\omega}^2 \\ \widetilde{d}^3 \\ \cdots \\ \widetilde{d}^{\widehat{M}} \\ \widetilde{\omega}^{\widehat{M}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ 0 \\ I \end{bmatrix} \cdot (g - \widehat{L}^T\widehat{v}^{n+1}) \tag{5.39}$$

the matrix $\widehat{B}^T$ is written as:

$$\widehat{B}^T = \begin{bmatrix} I & -\beta_2\widehat{D}^T & \alpha_2\widehat{C}^T\widehat{P}^T & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & I & \alpha_2\widehat{P}^T & \beta_3 - 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & I & -\beta_3\widehat{D}^T & \alpha_3\widehat{C}^T\widehat{P}^T & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & I & \alpha_3\widehat{P}^T & \beta_4 - 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & I & -\beta_5\widehat{D}^T & \alpha_{\widehat{M}}\widehat{C}^T\widehat{P}^T \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & I & \alpha_{\widehat{M}}\widehat{P}^T \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & I \end{bmatrix}$$

Therefore, it is possible to write the adjoint RK scheme as:

$$
\begin{aligned}
&\widetilde{\omega}^{\widehat{M}} = g - \widehat{L}^T \widehat{v}^{n+1} \\
&\widetilde{d}^{\widehat{M}} = -\alpha_{\widehat{M}} \widehat{P}^T \widetilde{\omega}^{\widehat{M}} \\
&\begin{cases} \widetilde{\omega}^m = -\alpha_{m+1} \widehat{C}^T \widehat{P}^T \widetilde{\omega}^{m+1} + \beta_{m+1} \widehat{D}^T \widetilde{d}^{m+1} \\ \widetilde{d}^m = -\alpha_m \widehat{P}^T \widetilde{\omega}^m + (1 - \beta_{m+1}) \widetilde{d}^{m+1} \end{cases} \quad m = \widehat{M} \text{ stage RK to 1} \\
&\widehat{v}^n = \widehat{v}^{n+1} + \sum_{m=1}^{\widehat{M}} \alpha_m \widehat{P}^T \widetilde{\omega}^m
\end{aligned}
\tag{5.40}
$$

As suggested by [174], changing $\widetilde{v}^m = \widehat{P}^T \widetilde{\omega}^m$ the scheme may be re-written as:

$$
\begin{aligned}
&\widetilde{v}^{\widehat{M}} = \widehat{P}^T (g - \widehat{L}^T \widehat{v}^{n+1}) \\
&\widetilde{d}^{\widehat{M}} = -\alpha_{\widehat{M}} \widetilde{v}^{\widehat{M}} \\
&\begin{cases} \widetilde{v}^m = \widehat{P}^T (-\alpha_{m+1} \widehat{C}^T \widetilde{v}^{m+1} + \beta_{m+1} \widehat{D}^T \widetilde{d}^{m+1}) \\ \widetilde{d}^m = -\alpha_m \widetilde{v}^m + (1 - \beta_{m+1}) \widetilde{d}^{m+1} \end{cases} \quad m = \widehat{M} \text{ stage RK to 1} \\
&\widehat{v}^n = \widehat{v}^{n+1} + \sum_{m=1}^{\widehat{M}} \alpha_m \widetilde{v}^m
\end{aligned}
\tag{5.41}
$$

where once again, the time reversal, in the form of backward marching, is clearly visible.

5.7 Flux Evaluation

Given the nature of *Hydra*, i.e. a *vertex-centred* flow solver, most calculations are computed employing edge-based data structures. Therefore, viscous and inviscid routines to replicate the propagation of information across the control boundary surfaces are called for each edge in the mesh. A similar procedure is employed by the adjoint code. The overall *dual* residuals at each grid node can be written as:

$$
\widehat{L}^T \widehat{v} - \widehat{p} = 0 \tag{5.42}
$$

The term $\widehat{p}$ or $-\frac{\partial f}{\partial \mathbf{Q}}\big|^T$ (i.e. functional sensitivity *w.r.t.* flow state) is evaluated at the first adjoint solver iteration by applying the final flow solution in the routine for the functional differentiated in reverse *w.r.t.* $\mathbf{Q}$. This can be considered as a constant source term at each grid node. In particular, this will be non-zero only on the surface of integration of the objective function. The quantity that is actually updated at every iteration is $\widehat{L}^T \widehat{v}$ or $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\big|^T \mathbf{\Psi}$. As previously demonstrated, transposing the flow solver

operators in order to determine the adjoint ones, reverses their order. As expected, this is also the case of the *gather* and *scatter* steps in the edgewise flux evaluation [171]. In the nonlinear and direct solver functions, the states of two nodes of an edge are the inputs, while the actual edge residual is the output, meaning that information is first *gathered* to the edge and then *scattered* to the nodes. On the other hand, for the adjoint, the difference between dual variables is first calculated. This is then employed in the flux routine and multiplied by the linearised *Jacobian* of the edge. The end output is then the left and right residual as separate values. This effectively reverses the *gather* and *scatter* operations.

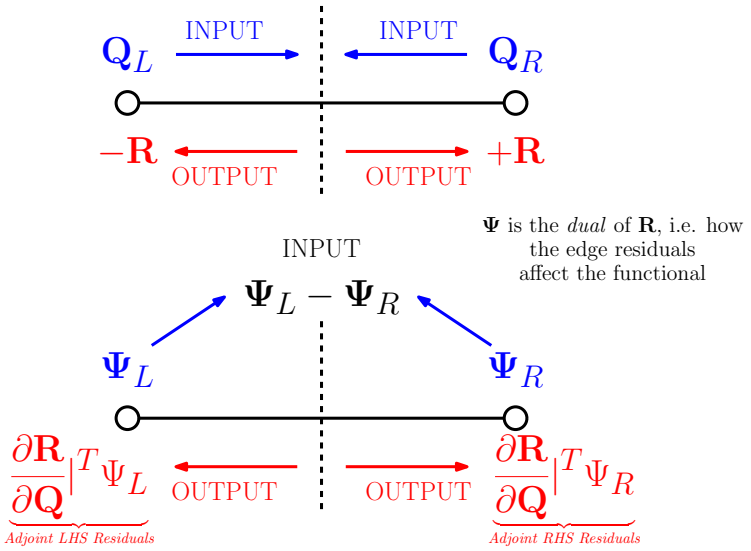


FIGURE 5.4: Difference between flow and adjoint solver of the *gather* and *scatter* operations.

By the same reasoning for the derivation of the various adjoint operators from those of the discrete flow solver, it is also possible to define the order of the calls to the viscous and inviscid flux functions. In fact, while in the *NS* solver the dissipative flux routine is called before convective/inviscid one, their order is reversed in the *dual* code. This was proven in section 5.6, and may be verified by comparing equations 5.27 and 5.41 i.e. the flow and adjoint *RK* schemes, respectively.

A final point concerning the flux evaluation, relates to the dissipative function. Writing the overall flux contribution as in eq. 5.26, the linearised viscous routine may be split into [175]:

$$\widehat{D}\widehat{u} = \widehat{V}G\widehat{u} \tag{5.43}$$

where G is the gradient and *pseudo-Laplacian* operator, while $\widehat{V}$ is the actual dissipative flux computation. The adjoint operator will transpose relation 5.43 [175]:

$$\widehat{D}^T\widehat{v} = G^T\widehat{V}^T\widehat{v} \tag{5.44}$$

Therefore the gradients and *pseudo-Laplacian* terms will be evaluated once the viscous fluxes have been computed.

5.8 Multigrid

As described by [174], the multigrid procedure for the adjoint solver requires modification. In simple terms, the transposition of the operators is used to derive it. The flow solver multigrid process may be indicated as:

$$c_h = I_H^h S_H I_h^H \tag{5.45}$$

where:

- c_h is the fine grid correction
- I_h^H is the *restriction* from fine to coarse grid
- S_H is the solution evaluated on the coarse mesh
- I_H^h is the *prolongation* from coarse to fine of the correction

Taking the transpose of this relation yields:

$$c_h^T = I_h^H|^T S_H^T I_H^h|^T \tag{5.46}$$

meaning that the adjoint prolongation is the transpose of the flow solver restriction and vice-versa. In the *NS* solver, the restriction operator will determine the coarse mesh values utilising those of the fine mesh as described in detail in section 4.7. This operator was written as (eq. 4.38):

$$\mathbf{R}_j^H = \frac{\sum_{i\in K_j} V_i^h \mathbf{R}_i^h}{max(V_j^H, \sum_{i\in K_j} V_i^h)}$$

As shown in [176], the matrix version of operator I_h^H is[7]:

[7] Assuming that the denominator term is always the coarse mesh control volume.

$$[\mathbf{R}]_H = \underbrace{\begin{bmatrix} 0 & \cdots & \frac{V_i^h}{V_1^H} & \frac{V_{i+1}^h}{V_1^H} & \cdots & 0 \\ \frac{V_1^h}{V_2^H} & 0 & \cdots & \cdots & 0 & \frac{V_{N_F}^h}{V_2^H} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{V_1^h}{V_{N_C}^H} & \frac{V_2^h}{V_{N_C}^H} & 0 & \cdots & \cdots & 0 \end{bmatrix}}_{I_h^H} \cdot [\mathbf{R}]_h \tag{5.47}$$

where N_C and N_F are the coarse and fine number of mesh nodes, respectively. The only non-zero entries are for fine mesh points contained in the relative coarse mesh control volume[8]. It is clear that, this matrix has dimensions of N_C-by-N_F, i.e. the number of rows is equal to the coarse mesh nodes, whilst the number of columns is that of the fine grid. Transposing:

$$[\mathbf{R}(\mathbf{\Psi})]_h|^T = \underbrace{\begin{bmatrix} 0 & \frac{V_1^h}{V_2^H} & \cdots & \frac{V_1^h}{V_{N_C}^H} \\ \cdots & 0 & \cdots & \frac{V_2^h}{V_{N_C}^H} \\ \frac{V_i^h}{V_1^H} & \cdots & \cdots & 0 \\ \frac{V_{i+1}^h}{V_1^H} & \cdots & \cdots & \cdots \\ \cdots & 0 & \cdots & \cdots \\ 0 & \frac{V_{N_F}^h}{V_2^H} & \cdots & 0 \end{bmatrix}}_{I_h^H\big|^T} \cdot [\mathbf{R}(\mathbf{\Psi})]_H|^T \tag{5.48}$$

At this point the number of rows is equal to the number of fine mesh nodes and vice-versa for the columns. A similar idea applies to the restriction operator, that may be achieved by transposing relation 4.39.

5.9 Boundary Conditions

The discrete adjoint requires extra care at solid walls, both for inviscid and viscous flow components [175, 177, 178]. In this work the full *NS* equations were considered, therefore its adjoint formulation for wall boundary conditions will be discussed. As described in section 4.9.1, at solid surfaces, the momentum fluxes and turbulence model, along with all velocity components and turbulent viscosity, are forced to zero[9]. This can be cast in matrix format as follows [178][10]

[8] In the matrix of eq. 5.47 the non-zero values are only indicative.
[9] This corresponds to a *Dirichlet* or *strong* boundary condition.
[10] The following derivation is a much more detailed version of that proposed by [178].

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\widetilde{B}} \cdot \underbrace{\begin{bmatrix} \rho \\ u \\ v \\ w \\ P \\ \nu_t \end{bmatrix}}_{U} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.49a}$$

$$\left(\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{I} - \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\widetilde{B}} \right) \cdot \underbrace{\begin{bmatrix} R(\text{continuity}) \\ R(\text{x-momentum}) \\ R(\text{y-momentum}) \\ R(\text{z-momentum}) \\ R(\text{energy}) \\ R(\text{turbulence-model}) \end{bmatrix}}_{R(U)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.49b}$$

or in short:

$$\widetilde{B}U = 0 \tag{5.50a}$$

$$(I - \widetilde{B})R(U) = 0 \tag{5.50b}$$

While eq. 5.49a forces to zero the velocity components and turbulent viscosity, 5.49b indicates that the only relations that are going to the be solved are continuity and energy, with the momentum and turbulence model residuals being discarded. Linearising the last two relations allows to determine the direct solver strong BC:

$$\widetilde{B}\widehat{u} = 0 \tag{5.51a}$$

$$(I - \widetilde{B})(\widehat{L}\widehat{u} - \widehat{p}) = 0 \tag{5.51b}$$

Combining this result yields:

$$((I - \widetilde{B})\widehat{L} + \widetilde{B})\widehat{u} = (I - \widetilde{B})\widehat{p} \tag{5.52}$$

At this point to determine the *dual* operator, the terms pre-multiplying $\widehat{u}$ have to be transposed:

$$(\widehat{L}^T(I^T - \widetilde{B}^T) + \widetilde{B}^T)\widehat{v} = g \xrightarrow[I^T=I]{\widetilde{B}^T=\widetilde{B}} (\widehat{L}^T(I - \widetilde{B}) + \widetilde{B})\widehat{v} = g \tag{5.53}$$

$\widetilde{B}$ and I being symmetric matrices may not be transposed. [178] splits between the components of $\widehat{v}$ and g that lie in the *null* space of $\widetilde{B}$ ($\widehat{v}_\parallel$ and $g_\parallel$)[11] and those that are orthogonal to this ($\widehat{v}_\perp$ and $g_\perp$). Put into simple terms, the *null* space quantities of $\widetilde{B}$ are those on which the matrix acts on, the orthogonal ones are independent of it. Therefore:

$$\widehat{v} = \widehat{v}_\perp + \widehat{v}_\parallel = \widetilde{B}\widehat{v} + (I - \widetilde{B})\widehat{v} \tag{5.54}$$

as:

$$\widehat{v}_\perp = \widetilde{B}\widehat{v} \tag{5.55a}$$

$$\widehat{v}_\parallel = (I - \widetilde{B})\widehat{v} \tag{5.55b}$$

Similarly for g:

$$g = g_\perp + g_\parallel = \widetilde{B}g + (I - \widetilde{B})g \Rightarrow \begin{cases} g_\perp = \widetilde{B}g \\ g_\parallel = (I - \widetilde{B})g \end{cases} \tag{5.56}$$

Pre-multiplying equation 5.53 by $(I - \widetilde{B})$:

$$\begin{aligned} [(I - \widetilde{B})\widehat{L}^T(I - \widetilde{B}) + (I - \widetilde{B})\widetilde{B}]\widehat{v} &= (I - \widetilde{B})g \\ (I - \widetilde{B})\widehat{L}^T(I - \widetilde{B})\widehat{v} + (I - \widetilde{B})\widetilde{B}\widehat{v} &= g_\parallel \\ (I - \widetilde{B})\widehat{L}^T\widehat{v}_\parallel + (\widetilde{B} - \widetilde{B}\widetilde{B})\widehat{v} &= g_\parallel \xrightarrow[\widetilde{B}\widetilde{B}=\widetilde{B}]{\widetilde{B} \text{ is idempotent}} \\ (I - \widetilde{B})\widehat{L}^T\widehat{v}_\parallel &= g_\parallel \end{aligned} \tag{5.57}$$

One can then determine the adjoint wall boundary condition that is solved iteratively as:

[11] This simply means that $\widetilde{B}\widehat{v}_\parallel = 0$ or $\widetilde{B}g_\parallel = 0$.

$$\begin{cases} (I - \widetilde{B})\widehat{L}^T\widehat{v}_{\parallel} = g_{\parallel} \\ \widetilde{B}\widehat{v}_{\parallel} = 0 \end{cases} \tag{5.58}$$

The orthogonal term $\widehat{v}_{\perp}$ is then calculated separately. The equation for this is derived using the same philosophy as 5.57, except pre-multiplying by $\widetilde{B}$ instead of $(I - \widetilde{B})$:

$$\begin{aligned} [\widetilde{B}\widehat{L}^T(I - \widetilde{B}) + \widetilde{B}\widetilde{B}]\widehat{v} = \widetilde{B}g &\xrightarrow[\widetilde{B}\widetilde{B}=\widetilde{B}]{\widetilde{B} \text{ is idempotent}} \\ [\widetilde{B}\widehat{L}^T(I - \widetilde{B}) + \widetilde{B}]\widehat{v} &= \widetilde{B}g \\ \widetilde{B}\widehat{L}^T\widehat{v}_{\parallel} + \widehat{v}_{\perp} &= g_{\perp} \\ \therefore \widehat{v}_{\perp} &= g_{\perp} - \widetilde{B}\widehat{L}^T\widehat{v}_{\parallel} \end{aligned} \tag{5.59}$$

From this information it is possible to determine the functional of interest as:

$$\begin{aligned} \frac{df}{d\widehat{\alpha}} &= \underbrace{\frac{\partial f}{\partial Q}}_{g^T}\underbrace{\frac{\partial Q}{\partial \widehat{\alpha}}}_{\widehat{u}} + \frac{\partial f}{\partial \widehat{\alpha}} \\ &= \underbrace{-\frac{\partial f}{\partial R}}_{\widehat{v}^T}\underbrace{\frac{\partial R}{\partial \widehat{\alpha}}}_{\widehat{p}} + \frac{\partial f}{\partial \widehat{\alpha}} \\ &= \widehat{v}^T\widehat{p} + \frac{\partial f}{\partial \widehat{\alpha}} \\ &= \widehat{v}^T[(I - \widetilde{B})\widehat{p}] + \frac{\partial f}{\partial \widehat{\alpha}} \\ &= \widehat{v}_{\parallel}^T\widehat{p} + \frac{\partial f}{\partial \widehat{\alpha}} \end{aligned} \tag{5.60}$$

where in the penultimate step the *RHS* of eq. 5.52 was applied.

5.10 Mesh Adjoint

One of the main differences between the continuous and discrete formulation of the adjoint procedure concerns the so-called mesh adjoint. In the first case, the functional sensitivity *w.r.t.* the geometry surface is directly evaluated. On the other hand, its discrete counterpart employs a flow solver linearisation and transposition, thus determining how f varies *w.r.t.* the flow residuals, generally indicated as the *flow adjoint* (Ψ_f)[12]. This does not indicate how the geometry surface perturbations will affect f. For this

[12]Till now it has been indicated as $\boldsymbol{\Psi}$ for generality purposes. However for the discrete version of the adjoint it is now necessary to differentiate between the flow component ($\boldsymbol{\Psi}_f$), and as discussed in this section, the mesh counterpart ($\boldsymbol{\Psi}_g$).

reason, there is a need to *"map"* the evaluation of $\frac{\partial f}{\partial \mathbf{R}}$ onto the geometry surface.
To be able to understand how this extra step is computed and the advantages provided by the mesh adjoint (Ψ_g) procedure, the derivation outlined by [179] is described. Considering a *Lagrangian* formulation ($\mathscr{L}$) of a functional f to be optimised:

$$\mathscr{L}(\widehat{\alpha}, \mathbf{Q}, \mathbf{X}, \mathbf{\Psi}_f) = f(\widehat{\alpha}, \mathbf{Q}, \mathbf{X}) + \mathbf{\Psi}_f^T \mathbf{R}(\widehat{\alpha}, \mathbf{Q}, \mathbf{X}) \tag{5.61}$$

Where $\widehat{\alpha}$ is a design variable, $\mathbf{X}$ are the grid coordinates/mesh metrics and the adjoint vector $\mathbf{\Psi}_f$ is acting as the *Lagrangian* multiplier of a quantity that is zero (i.e. the *NS* relations that may be seen as a constraint). Differentiating (i.e. linearising) *w.r.t.* the design variable $\widehat{\alpha}$:

$$\frac{d\mathscr{L}}{d\widehat{\alpha}} = \frac{\partial f}{\partial \widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{Q}}\frac{d\mathbf{Q}}{d\widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}} + \mathbf{\Psi}_f^T\left(\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\frac{d\mathbf{Q}}{d\widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}}\right) \tag{5.62}$$

To be able to remove $\frac{d\mathbf{Q}}{d\widehat{\alpha}}$, i.e. the most expensive quantity to evaluate, its multipliers are grouped:

$$\frac{d\mathscr{L}}{d\widehat{\alpha}} = \frac{\partial f}{\partial \widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}} + \left(\frac{\partial f}{\partial \mathbf{Q}} + \mathbf{\Psi}_f^T\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)\frac{d\mathbf{Q}}{d\widehat{\alpha}} + \mathbf{\Psi}_f^T\left(\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}}\right) \tag{5.63}$$

As $\mathbf{\Psi}_f$ is arbitrary, it may be formulated as:

$$\left.\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right|^T \mathbf{\Psi}_f = -\left.\frac{\partial f}{\partial \mathbf{Q}}\right|^T \tag{5.64}$$

as seen in section 5.2. The term that allows to map the flow sensitivity to that of the design variables is $\frac{d\mathbf{X}}{d\widehat{\alpha}}$. To determine this parameter it is necessary to know how perturbations of the surface mesh influence the entire grid. To determine this quantity a spring-stiffness mesh movement can be employed[13]:

$$\mathbf{K}\mathbf{X} = \mathbf{X}_{\text{surf}} \tag{5.65}$$

Where $\mathbf{K}$ is the stiffness matrix and $\mathbf{X}_{\text{surf}}$ is the surface grid. Linearising this relation about $\widehat{\alpha}$:

$$\mathbf{K}\frac{d\mathbf{X}}{d\widehat{\alpha}} = \frac{d\mathbf{X}_{\text{surf}}}{d\widehat{\alpha}} \tag{5.66}$$

[13]This is the same process utilised for the surface reconstruction in section 6.5.2.

The term on the *R.H.S.* indicates how the design variables influence the surface mesh. This quantity is generally determined by analytically differentiating the geometry parametrisation *w.r.t.* $\widehat{\alpha}$ and is a straightforward computation. The $\frac{d\mathbf{X}}{d\widehat{\alpha}}$ term has a design variable dependency and therefore ought to be computed for every different one. As shown in [180], this procedure is time consuming and computationally expensive. For this reason, [179] devised an alternative way of calculating the mesh sensitivities. In eq. 5.61 an extra adjoint term, called *mesh adjoint* ($\mathbf{\Psi}_g$), multiplying the *linear elasticity* eq. was introduced:

$$\widehat{\mathscr{L}}(\widehat{\alpha}, \mathbf{Q}, \mathbf{X}, \mathbf{\Psi}_f, \mathbf{\Psi}_g) = f(\widehat{\alpha}, \mathbf{Q}, \mathbf{X}) + \mathbf{\Psi}_f^T \mathbf{R}(\widehat{\alpha}, \mathbf{Q}, \mathbf{X}) + \mathbf{\Psi}_g^T(\mathbf{K}\mathbf{X} - \mathbf{X}_{\text{surf}}) \tag{5.67}$$

Again, linearising this relation *w.r.t.* the design variables and grouping the $\frac{d\mathbf{Q}}{d\alpha}$ terms as before:

$$\begin{aligned}
\frac{d\widehat{\mathscr{L}}}{d\widehat{\alpha}} &= \frac{\partial f}{\partial \widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{Q}}\frac{d\mathbf{Q}}{d\widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}} + \mathbf{\Psi}_f^T\left(\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\frac{d\mathbf{Q}}{d\widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}}\right) + \mathbf{\Psi}_g^T\left(\mathbf{K}\frac{d\mathbf{X}}{d\widehat{\alpha}} - \frac{d\mathbf{X}_{\text{surf}}}{d\widehat{\alpha}}\right) \\
&= \frac{\partial f}{\partial \widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}} + \cancel{\left(\frac{\partial f}{\partial \mathbf{Q}} + \mathbf{\Psi}_f^T\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)\frac{d\mathbf{Q}}{d\widehat{\alpha}}}^{0} + \mathbf{\Psi}_f^T\left(\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}}\right) + \mathbf{\Psi}_g^T\left(\mathbf{K}\frac{d\mathbf{X}}{d\widehat{\alpha}} - \frac{d\mathbf{X}_{\text{surf}}}{d\widehat{\alpha}}\right) \\
&= \frac{\partial f}{\partial \widehat{\alpha}} + \frac{\partial f}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}} + \mathbf{\Psi}_f^T\left(\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} + \frac{\partial \mathbf{R}}{\partial \mathbf{X}}\frac{d\mathbf{X}}{d\widehat{\alpha}}\right) + \mathbf{\Psi}_g^T\left(\mathbf{K}\frac{d\mathbf{X}}{d\widehat{\alpha}} - \frac{d\mathbf{X}_{\text{surf}}}{d\widehat{\alpha}}\right)
\end{aligned} \tag{5.68}$$

To be able to remove $\frac{d\mathbf{X}}{d\widehat{\alpha}}$ the same idea as for the flow adjoint is used, i.e. group the terms multiplying it and set $\mathbf{\Psi}_g$ accordingly to force them to 0:

$$\frac{d\widehat{\mathscr{L}}}{d\widehat{\alpha}} = \frac{\partial f}{\partial \widehat{\alpha}} + \left(\frac{\partial f}{\partial \mathbf{X}} + \mathbf{\Psi}_f^T\frac{\partial \mathbf{R}}{\partial \mathbf{X}} + \mathbf{\Psi}_g^T\mathbf{K}\right)\frac{d\mathbf{X}}{d\widehat{\alpha}} + \mathbf{\Psi}_f^T\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} - \mathbf{\Psi}_g^T\frac{d\mathbf{X}_{\text{surf}}}{d\widehat{\alpha}} \tag{5.69}$$

Defining:

$$\mathbf{K}^T\mathbf{\Psi}_g = -\left.\frac{\partial f}{\partial \mathbf{X}}\right|^T - \left.\frac{\partial \mathbf{R}}{\partial \mathbf{X}}\right|^T \mathbf{\Psi}_f \tag{5.70}$$

The term in the brackets is forced to 0 and the full sensitivity equation becomes:

$$\frac{d\widehat{\mathscr{L}}}{d\widehat{\alpha}} = \frac{\partial f}{\partial \widehat{\alpha}} + \mathbf{\Psi}_f^T\frac{\partial \mathbf{R}}{\partial \widehat{\alpha}} - \mathbf{\Psi}_g^T\frac{d\mathbf{X}_{\text{surf}}}{d\widehat{\alpha}} \tag{5.71}$$

Therefore, as required, the mesh adjoint removes the dependency of the mesh sensitivities to the design parameters, and thus it is only necessary to evaluate it once. Within the

final sensitivity chain it is then multiplied by the geometry parametrisation differentiated by $\widehat{\alpha}$, a relatively inexpensive calculation.
As previously discussed, within *Hydra*, the adjoint code is a discrete formulation achieved by employing *AD*. To be able to determine quantities such as $\frac{\partial f}{\partial \mathbf{X}}^T$ the routine developed to compute f is differentiated in *reverse mode*[14] *w.r.t.* the mesh metrics, such as control surfaces and volumes of each dual cell and the coordinates [181]. Therefore the chain differentiation for this term is:

$$\frac{\partial f}{\partial \mathbf{X}}^T = \frac{\partial f}{\partial \mathbf{X}_c}^T + \frac{\partial f}{\partial \mathbf{M}}^T \frac{d\mathbf{M}}{d\mathbf{X}_c}^T \tag{5.72}$$

where $\mathbf{X}_c$ are the grid coordinates and $\mathbf{M}$ are the mesh metrics that will depend on $\mathbf{X}_c$. Similarly, the *NS* flux and source term routines are run through the automatic differentiation software in reverse mode, again *w.r.t.* the grid metrics and coordinates as for eq. 5.72 [181]. Once the flow adjoint vector for each node, $\frac{\partial f}{\partial \mathbf{X}}$ and $\frac{\partial \mathbf{R}}{\partial \mathbf{X}}$ have been calculated, $\mathbf{\Psi}_g$ may be determined. Equation 5.70 is solved setting up the stiffness matrix $\mathbf{K}$ as the inverse of the grid's edge lengths and applying a *conjugate gradient* solver to the system [181]. The terms on the *R.H.S.* of relation 5.70 can be seen as the source terms of the system, while the mesh adjoint indicates how the information from each grid node propagates to the geometry surface. During the iterative process to solve the relation, all wall displacements are forced to zero. Once the system has converged, the surface geometry quantities are reconstructed from those of the entire domain employing the procedure outlined in section 5.9 [181].

5.11 Automatic Differentiation

Automatic or *algorithmic* or *computational* differentiation consists in applying the chain rule of calculus to the source code of the software of interest [168]. For every code, it is possible to define *independent* variables as the inputs to it and the *dependent* ones as those calculated throughout along with the outputs. By grouping these together a set of m variables is achieved, from here on denoted as $\widehat{t}_i$. Therefore, assuming the code has two inputs and outputs, $\widehat{t}_1$ and $\widehat{t}_2$ will be the inputs, while $\widehat{t}_m$ and $\widehat{t}_{m-1}$ will be the outputs. The intermediate variables will span from $i = 3$ to $i = m-2$. It is now possible to define the chain rule applied to the source as [168]:

$$\frac{\partial \widehat{t}_i}{\partial \widehat{t}_j} = \delta_{ij} + \sum_{k=j}^{i-1} \frac{\partial \widehat{t}_i}{\partial \widehat{t}_k} \frac{\partial \widehat{t}_k}{\partial \widehat{t}_j}, \qquad j \leq i \leq m \tag{5.73}$$

[14] Due to the transposition being present.

where δ_{ij} is the *Kronecker* delta, i.e. 1 when $i = j$, 0 otherwise. If the *forward mode* of differentiation is chosen, the summation is carried out by differentiating each instruction in the order in which they appear till the last. In the previous example, if variable $m-1$, i.e. the first output, were to be differentiated *w.r.t.* the first input, then $i = m - 1$ and $j = \{1, 3, \ldots, m - 2\}$ (where 2 and m have been omitted as they represent the other input and output, respectively). As all the derivatives are computed in a sequence *w.r.t.* that particular input, little extra effort is required to be able to compute all the output sensitivities *w.r.t.* that input [168]. On the other hand, if the reverse mode of differentiation had been selected, then index j will span from the output index of interest to the desired input. So in the previous example $j = \{m-1, m-2, ..., 3, 1\}$. As opposed to the *forward mode*, this technique allows the determination of the sensitivity of a single output *w.r.t.* all inputs with a few extra computations. Nevertheless, it is the most expensive of the two approaches memory-wise as all the intermediate variables ought to be stored [168].

An example of the *forward* and *reverse* differentiation process is reported in appendix B, where a similar procedure to that in [168] is proposed.

A final note concerns the software employed in *Hydra* to be able to obtain either the direct or adjoint solver code from the original *NS* source. This is done utilising the *TAPENADE* package provided by *INRIA* [182].

Chapter 6

Mesh Adaptation Tools

6.1 Introduction

The software chosen for this work is *MeshPost* (*MP*), a *RR* in-house error estimation and mesh adaptation code. It is a hybrid software written in *C* and *Fortran* 77, with the *C* part forming the bulk of it, and the *Fortran* sections being inherited from the solver pre-processing software. The latter mainly comprises of routines to determine connectivities, such as face-to-cell, or grid metrics, i.e. control surfaces and volumes. The computation of error sensors, mesh movement and refinement are carried out in *C*. As part of this work the code was upgraded to *HDF*5 ([129]) data format to match that of the latest versions of *Hydra*. Additionally, the program was also extensively debugged/cleaned. The mesh movement previously implemented and integrated, was appropriately corrected. To be able to have a smooth reconstruction of the *Hessian* field, an improved gradient calculation was devised along with ghost nodes implementation on the geometry boundaries, this to reduce any noisiness appearing. Moreover, adjoint error estimation was successfully developed, along with the smooth coarse-to-fine and vice-versa interpolation routines required to compute this quantity. Attempts were made to code processes to reconstruct the surface to allow nodal movement on the geometry and for its refinement. Unfortunately these techniques are still under development at the moment of writing. Lastly, novel *anisotropic* adjoint-error estimation capabilities were devised and successfully implemented into the code. In the sections that follow, the main characteristics of the software will be described in detail.

6.2 Grid Error Calculation

6.2.1 Feature-Based Approaches

These kind of techniques employ values computed by the flow solver to determine variations along edges or derivatives. In particular, the value selected for error estimation is crucial. Certainly, a flow variable may be able to detect different flow features *w.r.t.* another. For e.g., the static pressure allows to capture shocks till the geometry surface, but not the boundary layer. The velocity magnitude gradient, instead, varies within the boundary layer but is zero at the wall. There are a variety of quantities that may be employed, and mostly depend on the case analysed. However, it is generally accepted that the *Mach* number is likely be the best choice. This because it is able to capture most of the flow features and relates to the fluid compressibility.

6.2.2 Gradient & Hessian Values Computation

In general, there are two main approaches pursued in the literature to compute the second-order derivatives of a scalar flow parameter of interest. Both stem from the computation of gradients within the flow solver, meaning that one may use *GG* or *LSQ* methods. In either case, the *Hessian* matrix is obtained by repeating the operation twice, i.e. first to calculate the gradient, second to determine the necessary derivatives. Before proceeding to discuss in more detail the procedure undertaken by the author, an important correction needed for rotating elements simulated with a periodic boundary must be made. In fact, in this case the treatment of these requires a rotation matrix: assuming x as the axial direction (i.e. the blade's centre of rotation), and periodic boundary planes present on each side of the aerofoil, the y and z component are coupled. In the particular case of vertex-centred schemes, the values will be stored on nodes forming the periodic surfaces, meaning that (approximately, for a fully hexahedral grid) half the dual control volume will be on one side, the remainder on the other. Therefore, for all periodic nodes, all quantities require an addition between the two matching points. For the case of a vectorial quantity, such as a the gradient of a scalar parameter q, assuming the periodic angle is θ:

$$\begin{aligned}\nabla q|^{\text{Total}}_{\text{Periodic side 1}} &= \nabla q|_{\text{Periodic side 1}} + \mathsf{R}(\theta) \cdot \nabla q|_{\text{Periodic side 2}} \\ \nabla q|^{\text{Total}}_{\text{Periodic side 2}} &= \mathsf{R}^T(\theta) \cdot \nabla q|_{\text{Periodic side 1}} + \nabla q|_{\text{Periodic side 2}}\end{aligned}$$

where the rotation matrix $\mathsf{R}(\theta)$, as show in section 4.9.2, is:

$$\mathsf{R}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{6.1}$$

Expanding:

$$\left.\frac{\partial q}{\partial x}\right|^{Total}_{\text{Periodic side 1}} = \left.\frac{\partial q}{\partial x}\right|^{Total}_{\text{Periodic side 2}} = \left.\frac{\partial q}{\partial x}\right|_{\text{Periodic side 1}} + \left.\frac{\partial q}{\partial x}\right|_{\text{Periodic side 2}}$$

$$\begin{aligned}
\left.\frac{\partial q}{\partial y}\right|^{Total}_{\text{Periodic side 1}} &= \left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 1}} + \cos(\theta)\left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 2}} + \sin(\theta)\left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 2}} \\
\left.\frac{\partial q}{\partial y}\right|^{Total}_{\text{Periodic side 2}} &= \left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 2}} + \cos(\theta)\left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 1}} - \sin(\theta)\left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 1}}
\end{aligned}$$

$$\begin{aligned}
\left.\frac{\partial q}{\partial z}\right|^{Total}_{\text{Periodic side 1}} &= \left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 1}} - \sin(\theta)\left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 2}} + \cos(\theta)\left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 2}} \\
\left.\frac{\partial q}{\partial z}\right|^{Total}_{\text{Periodic side 2}} &= \left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 2}} + \sin(\theta)\left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 1}} + \cos(\theta)\left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 1}}
\end{aligned}$$

At this point it should be stressed that computing the complete y and z components separately on each side can generate numerical errors that iteratively grow. Therefore, an alternative approach that enforces equality of the two sides should be used instead:

$$\begin{aligned}
\left.\frac{\partial q}{\partial y}\right|^{Total}_{\text{Periodic side 1}} &= \left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 1}} + \cos(\theta)\left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 2}} + \sin(\theta)\left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 2}} \\
\left.\frac{\partial q}{\partial y}\right|^{Total}_{\text{Periodic side 2}} &= \cos(\theta)\left.\frac{\partial q}{\partial y}\right|^{Total}_{\text{Periodic side 1}} - \sin(\theta)\left.\frac{\partial q}{\partial z}\right|^{Total}_{\text{Periodic side 1}}
\end{aligned}$$

$$\begin{aligned}
\left.\frac{\partial q}{\partial z}\right|^{Total}_{\text{Periodic side 1}} &= \left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 1}} - \sin(\theta)\left.\frac{\partial q}{\partial y}\right|_{\text{Periodic side 2}} + \cos(\theta)\left.\frac{\partial q}{\partial z}\right|_{\text{Periodic side 2}} \\
\left.\frac{\partial q}{\partial z}\right|^{Total}_{\text{Periodic side 2}} &= \sin(\theta)\left.\frac{\partial q}{\partial y}\right|^{Total}_{\text{Periodic side 1}} + \cos(\theta)\left.\frac{\partial q}{\partial z}\right|^{Total}_{\text{Periodic side 1}}
\end{aligned}$$

Finally, for second order derivatives, $\mathbf{H}(q)$, computed applying the gradient operator to the ∇q terms:

$$\mathbf{H}(q)\big|^{\text{Total}}_{\text{Periodic side 1}} = \mathbf{H}(q)\big|_{\text{Periodic side 1}} + \mathsf{R}(\theta)\mathbf{H}(q)\big|_{\text{Periodic side 2}}\mathsf{R}^T(\theta) \tag{6.2}$$

$$\mathbf{H}(q)\big|^{\text{Total}}_{\text{Periodic side 2}} = \mathsf{R}^T(\theta)\mathbf{H}(q)\big|^{\text{Total}}_{\text{Periodic side 1}}\mathsf{R}(\theta) \tag{6.3}$$

In general unstructured flow solvers, for the *GG* case, the gradients are computed by looping around the edges to which a node belongs to and accumulating the variation across each one, as detailed in section 4.5.3. The *LSQ* approach instead [183, 184], is subject to a *Taylor-Series* with matrix inversion. As the latter approach only requires the distance and variable variation between any two nodes, it is fairly straightforward to include into each node's derivatives computation multiple layers of neighbouring nodes, i.e. grid points that are connected to the neighbours and not directly to the node itself. In this work, in an attempt to smooth out the derivatives computation and increase accuracy, three layers of neighbouring points were considered. This, significantly increased the number of values in play to compute the derivatives for each node: approximately 50 for the boundaries and 100 for the rest (in a typical *structured multiblock* mesh). This means that the matrix relative to the *LSQ* computation has a sufficient amount of entries to directly calculate the first and second order derivatives (thus achieving a third-order accurate *Taylor-Series*):

$$\underbrace{\begin{bmatrix}
\Delta x_{1,j} & \Delta y_{1,j} & \Delta z_{1,j} & (\Delta x_{1,j})^2 & \Delta x_{1,j}\Delta y_{1,j} & \Delta x_{1,j}\Delta z_{1,j} & \Delta y_{1,j}\Delta x_{1,j} & (\Delta y_{1,j})^2 & \Delta y_{1,j}\Delta z_{1,j} & \Delta z_{1,j}\Delta x_{1,j} & \Delta z_{1,j}\Delta y_{1,j} & (\Delta z_{1,j})^2 \\
\Delta x_{2,j} & \Delta y_{2,j} & \Delta z_{2,j} & (\Delta x_{2,j})^2 & \Delta x_{2,j}\Delta y_{2,j} & \Delta x_{2,j}\Delta z_{2,j} & \Delta y_{2,j}\Delta x_{2,j} & (\Delta y_{2,j})^2 & \Delta y_{2,j}\Delta z_{2,j} & \Delta z_{2,j}\Delta x_{2,j} & \Delta z_{2,j}\Delta y_{2,j} & (\Delta z_{2,j})^2 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\Delta x_{n,j} & \Delta y_{n,j} & \Delta z_{n,j} & (\Delta x_{n,j})^2 & \Delta x_{n,j}\Delta y_{n,j} & \Delta x_{n,j}\Delta z_{n,j} & \Delta y_{n,j}\Delta x_{n,j} & (\Delta y_{n,j})^2 & \Delta y_{n,j}\Delta z_{n,j} & \Delta z_{n,j}\Delta x_{n,j} & \Delta z_{n,j}\Delta y_{n,j} & (\Delta z_{n,j})^2
\end{bmatrix}}_{\mathbf{\Delta x}} \cdot \underbrace{\begin{bmatrix}
\frac{\partial q}{\partial x} \\ \frac{\partial q}{\partial y} \\ \frac{\partial q}{\partial z} \\
\frac{1}{2}\frac{\partial^2 q}{\partial x^2} \\ \frac{\partial^2 q}{\partial x \partial y} \\ \frac{\partial^2 q}{\partial x \partial z} \\ \frac{\partial^2 q}{\partial y \partial x} \\ \frac{1}{2}\frac{\partial^2 q}{\partial y^2} \\ \frac{\partial^2 q}{\partial y \partial z} \\ \frac{\partial^2 q}{\partial z \partial x} \\ \frac{\partial^2 q}{\partial z \partial y} \\ \frac{1}{2}\frac{\partial^2 q}{\partial z^2}
\end{bmatrix}}_{\begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix}} = \underbrace{\begin{bmatrix}
\Delta q_{1,j} \\ \Delta q_{2,j} \\ \dots \\ \dots \\ \dots \\ \Delta q_{n,j}
\end{bmatrix}}_{\mathbf{\Delta q}}$$

Where j is the node about which the derivatives are computed and 1 to n are the neighbouring points considered in the calculation. Additionally, the terms indicated in *red*, represent the quantities that may be computed directly by including additional neighbouring nodes.

At this point, it should be noted that the *Weighted-LSQ* version was adopted. This includes a weighting matrix in front of both, the spatial distance and the variable difference matrices. It is generally employed to deal with high aspect-ratio cells [42]. It is formulated as:

$$\mathbf{W} = \begin{bmatrix} \frac{1}{\sqrt{(\Delta x_{1,j})^2+(\Delta y_{1,j})^2+(\Delta z_{1,j})^2}} & 0 & \dots & \dots & 0 \\ 0 & \frac{1}{\sqrt{(\Delta x_{2,j})^2+(\Delta y_{2,j})^2+(\Delta z_{2,j})^2}} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & \frac{1}{\sqrt{(\Delta x_{n,j})^2+(\Delta y_{n,j})^2+(\Delta z_{n,j})^2}} \end{bmatrix}$$

It should be obvious that this matrix represents the inverse of the *Euclidian* distance between node j and any one of the others considered in the calculation. Therefore, the full formulation of the *Weighted-LSQ* is[1]:

$$\begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix} = (\mathbf{W} \cdot \mathbf{\Delta x})^{-1} \cdot \mathbf{W} \cdot \mathbf{\Delta q} \tag{6.4}$$

From the coding viewpoint, the periodic boundaries require significant change in the data structures: in fact the previous rotation procedure can not be adopted, as both the first and second order derivatives are computed simultaneously. Instead, the variable arrays are increased in size. This to store the rotated and non-rotated values for all the nodes needed to compute the derivatives across periodic boundaries and the three neighbouring layers.

A final remark, concerns the computational effort for the above calculation. In fact, the *LSQ* procedure will require a larger memory storage with respect to the *GG*-technique[2], that is further augmented due to the greater number of neighbours considered. Moreover, extra arrays containing the neighbouring indices need to be evaluated and stored. As this approach is only applied to the mesh adaptation procedure, and hence evaluated only once, the increased time and memory do not place a burden on the entire process, but allow an improvement of the derivative smoothness.

[1] It should be noted that the $\mathbf{\Delta x}$ matrix is not invertible as it is not square. This will be handled by the *QR* factorisation as described in the next section.

[2] The *QR*-factorisation needs to be computed and stored for every node, while for the *GG*-approach, only the volume and area elements of each element need to be determined and saved to memory.

6.2.2.1 *QR* Factorisation

Once the *LSQ* matrices have been determined, a *QR* factorisation with *Modified Gram-Schmidt* decomposition may be brought into play to solve the system of equations [185]. In particular this was necessary because it was an over-determined system and the matrix to be inverted was non-square. The algorithm consists in determining an orthonormal matrix[3] $\mathbf{Q}_{QR}$ that has the same dimensions as $\mathbf{\Delta x}$ along with an upper triangular matrix $\mathbf{R}_{QR}$, with dimensions equal to the rows of $\begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix}$, that may be easily inverted. The overall algorithm is shown in listing 2.

Algorithm 2 Modified *Gram-Schmidt QR* Factorisation

procedure QR_DECOMPOSITION
    %%Determine the diagonal elements of $\mathbf{R}_{QR}$ *and normalise* $\mathbf{Q}_{QR}$
    for $i = 1 : n$ **do**
        $R_{QR}(i,i) = \sqrt{(\sum_j \Delta x(j,i)^2)}$
        for $j = 1 : m$ **do**
            $Q_{QR}(j,i) = \Big(\Delta x(j,i)/R_{QR}(i,i)\Big)$
        end for
        %%Determine non-diagonal terms of $\mathbf{R}_{QR}$ *and orthogonalise* $\mathbf{Q}_{QR}$
        for $j = i + 1 : n$ **do**
            for $k = 1 : m$ **do**
                $R_{QR}(i,j) = R_{QR}(i,j) + \Delta x(k,j) \cdot \mathbf{Q}_{QR}(k,i))$
                $Q_{QR}(k,j) = Q_{QR}(k,j) - R_{QR}(i,j) \cdot \mathbf{Q}_{QR}(k,i)\big)$
            end for
        end for
    end for
end procedure

Once the $\mathbf{\Delta x}$ matrix has been decomposed, the following can be shown:

[3] All columns are normalised and orthogonal *w.r.t.* each other.

$$
\begin{aligned}
\boldsymbol{\Delta}\mathbf{x} \cdot \begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix} &= \boldsymbol{\Delta}\mathbf{q} \\
\mathbf{Q}_{QR} \cdot \mathbf{R}_{QR} \cdot \begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix} &= \boldsymbol{\Delta}\mathbf{q} \\
\cancel{\mathbf{Q}_{QR}^T \mathbf{Q}_{QR}} \cdot \mathbf{R}_{QR} \cdot \begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix} &= \mathbf{Q}_{QR}^T \cdot \boldsymbol{\Delta}\mathbf{q} \\
\begin{bmatrix} \nabla q_j \\ \mathbf{H}(q)_j \end{bmatrix} &= \mathbf{R}_{QR}^{-1} \cdot \mathbf{Q}_{QR}^T \cdot \boldsymbol{\Delta}\mathbf{q}
\end{aligned}
\tag{6.5}
$$

To validate the process, samples of the *LSQ* matrices relative to a few nodes were outputted to a text file and imported to MATLAB®. Employing the $[\mathbf{Q}_{QR}, \mathbf{R}_{QR}] = qr(\Delta\mathbf{x}, 0)$ command [186] it was possible to compute the QR factorisation and compare it with that evaluated in *MP*.

6.2.2.2 Ghost Nodes

One of the main issues with the derivative computation are their evaluation at walls. This is particularly true for *median-dual-vertex* based systems, as the value of each flow parameter will be stored on the boundary. Considering a *GG* formulation of the gradient, it can be easily understood that any value at the wall will have half the size of the control volume[4] *w.r.t.* the first layer of nodes off it, as displayed in figure 6.1. This means that, if slight numerical errors are present, when computing the wall derivatives, these can double in magnitude when applying the control volume division.

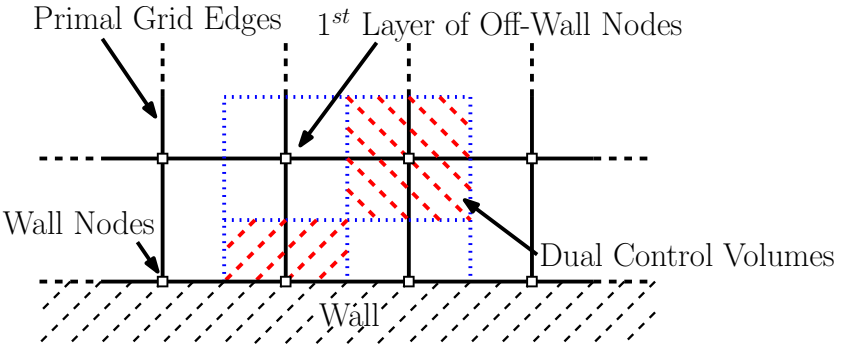


FIGURE 6.1: Wall and near-wall control-volumes.

Additionally, if different types of cells are present at these boundaries, anti-symmetry in the wall tangential direction can appear, thus causing large oscillations in the derivative values. Employing these without care when interpolating, can cause negative viscosity

[4] Assuming a perfectly quadrilateral grid with no cell-size variation.

or density. On the other hand, these effects tend to be attenuated by the *cell-centred FVM* formulation as the control volume size, in the case shown in figure 6.1, would not be halved.

While the problems described have been reduced by the use of multiple layers of neighbouring points, along with a higher order derivative calculation scheme, the issue of having a one-sided gradient computation would further reduce accuracy. Therefore the concept of *ghost-nodes* was introduced, i.e. any near wall node is reflected *w.r.t.* the surface node it is connected to. This means that the non-tangential coordinates are mirrored about the wall, whilst vectorial quantities will have their sign reversed. An example of the effect of mirroring of near-wall nodes *w.r.t.* the surface points is shown in figure 6.2.

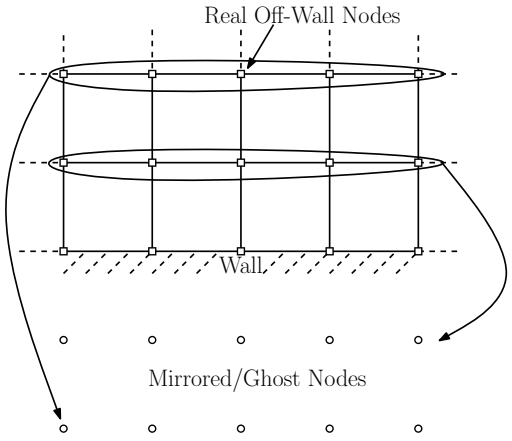


FIGURE 6.2: Mirroring of near-wall nodes to determine ghost-nodes.

Due to the use of *LSQ*-formulation for the derivative computation with three layers of neighbouring nodes, three rows of points off the boundary were mirrored. To achieve this, a list of nodes within the 3-layer distance from the wall was created and then appended to the original list of nodes. Furthermore, their values were determined by copying (in the case of scalar quantities) the variable of interest to the ghost-point IDs. An extra array storing the ghost index of each node was created[5], thus simplifying the overall neighbouring point determination. It is important to note that the mirroring of each point was computed by determining the nearest wall node and reverting the coordinates about this using the surface normal and distance between the two points.

[5]If the node did not have a ghost node it would be flagged with 0.

6.2.3 Riemannian Metric Computation

Once computed, the *Hessian* matrix of second derivatives may be employed to adapt the mesh in an edge-based manner as described by [21, 29, 42]. In a *FVM* discretisation[6], the error in each element can be approximated as the product of its characteristic length and the second order derivatives. As was discussed and proven in appendix A, by splitting the grid error equally between all its elements, the optimal mesh may be achieved (i.e. *principle of equidistribution*). Therefore by being able to redistribute in an homogeneous manner the *Hessian* matrix multiplied by the characteristic length, an improved mesh may be obtained. As stated by [21], this quantity can be identified as a *Riemannian metric* for each edge. However, to accomplish this it is first necessary to decompose the *Hessian* matrix $\mathbf{H}$ into its eigenvalues and eigenvectors:

$$\mathbf{H} = \mathbf{v}\lambda\mathbf{v}^T \tag{6.6}$$

where $\mathbf{H}$ is defined as:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 q}{\partial x^2} & \frac{\partial^2 q}{\partial x \partial y} & \frac{\partial^2 q}{\partial x \partial z} \\ \frac{\partial^2 q}{\partial x \partial y} & \frac{\partial^2 q}{\partial y^2} & \frac{\partial^2 q}{\partial y \partial z} \\ \frac{\partial^2 q}{\partial x \partial z} & \frac{\partial^2 q}{\partial y \partial z} & \frac{\partial^2 q}{\partial z^2} \end{bmatrix} \tag{6.7}$$

with q being any scalar flow quantity of interest. A *Riemannian* metric requires a *positive-semi-definite* formulation of $\mathbf{H}$ [64], therefore the absolute value of the eigenvalues is necessary:

$$\overline{\mathbf{H}} = \mathbf{v}|\lambda|\mathbf{v}^T \tag{6.8}$$

Effectively, to achieve equidistribution of the error, each edge should have unit length in the *Riemannian* metric [14]:

$$\mathbf{d}^T\overline{\mathbf{H}}\mathbf{d} = 1 \tag{6.9}$$

where $\mathbf{d}$ is the edge length. In 3D this corresponds to a unit sphere, (i.e. the error is identical in the three directions), representing an ellipsoid in physical space, as shown in figure 6.3. It is important to note that each axis is stretched by an amount equal to $\mathbf{h} = \frac{1}{\sqrt{\lambda}}$, i.e. the inverse of the square-root of each eigenvalue of $\overline{\mathbf{H}}$. On the other

[6] The reasoning behind this approach is also valid for *FEM* codes.

hand, the axis rotation corresponds to the eigenvectors of $\overline{\mathbf{H}}$. It is this characteristic that allows to align in an *anisotropic* manner the grid edges with the flow features, thus clustering nodes strategically and efficiently.

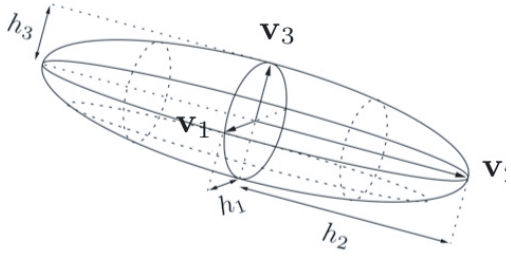


FIGURE 6.3: *Riemannian* metric in physical space [46][7].

.

The actual *anisotropic* edge length in 3D is obtained as:

$$\mathbf{d}^* = \sqrt{\mathbf{d}\overline{\mathbf{H}}\mathbf{d}} \tag{6.10}$$

where $\mathbf{d}^*$ is the new, *anisotropic* edge length in each direction, while $\mathbf{d}$ represents the original spacing. As a final remark on the method, it is important to note that the edge-length is inversely-proportional to the error, i.e. the larger the latter, the shorter the edge, as this would be necessary to improve the solution resolution.
In general the most successfully parameter choice for q is the *Mach* number, or *relative* in the case of turbomachinery. This because it allows to capture the wakes and shocks, that are in general the most complex parts of the flow.
In figures 6.4a, 6.4c and 6.4e the diagonal components of the *Hessian* matrix as reconstructed using eq. 6.8 are shown for the starting mesh of *NASA Rotor* 37 at mid span. As it can clearly be seen, shock and wake are strongly highlighted by the error map. In figures 6.4b, 6.4d and 6.4f the same value is shown but after applying the mesh movement algorithm (described in detail in section 6.5.3) 50 times to equidistribute the error. The grids in question are reported in figure 6.5. It is obvious that the procedure has clustered the nodes at the sources of error, thus allowing an improved resolution. In fact, carefully looking at the final grid of the adaptation process in figure 6.5b, the shocks and wake are clearly visible.
The experienced reader will have noticed an issue employing such an approach: flow features such as shocks are in fact complexities with very high gradient across them[8].

[7]Reproduced with kind permission of the *Society for Industrial and Applied Mathematics* (*SIAM*).
[8]This is for the full *NS* relations case, where viscosity will cause the shock to have a finite gradient. When simulating *Euler*'s relations, the gradient is effectively infinite.

For this reason, regardless of the amount of clustering present, they will keep attracting nodes even if unnecessary. This is one of the reasons behind the improved efficiency of the adjoint-related techniques, since they do not tend to over-refine, but only add the necessary number of nodes to minimise the error in that cell.

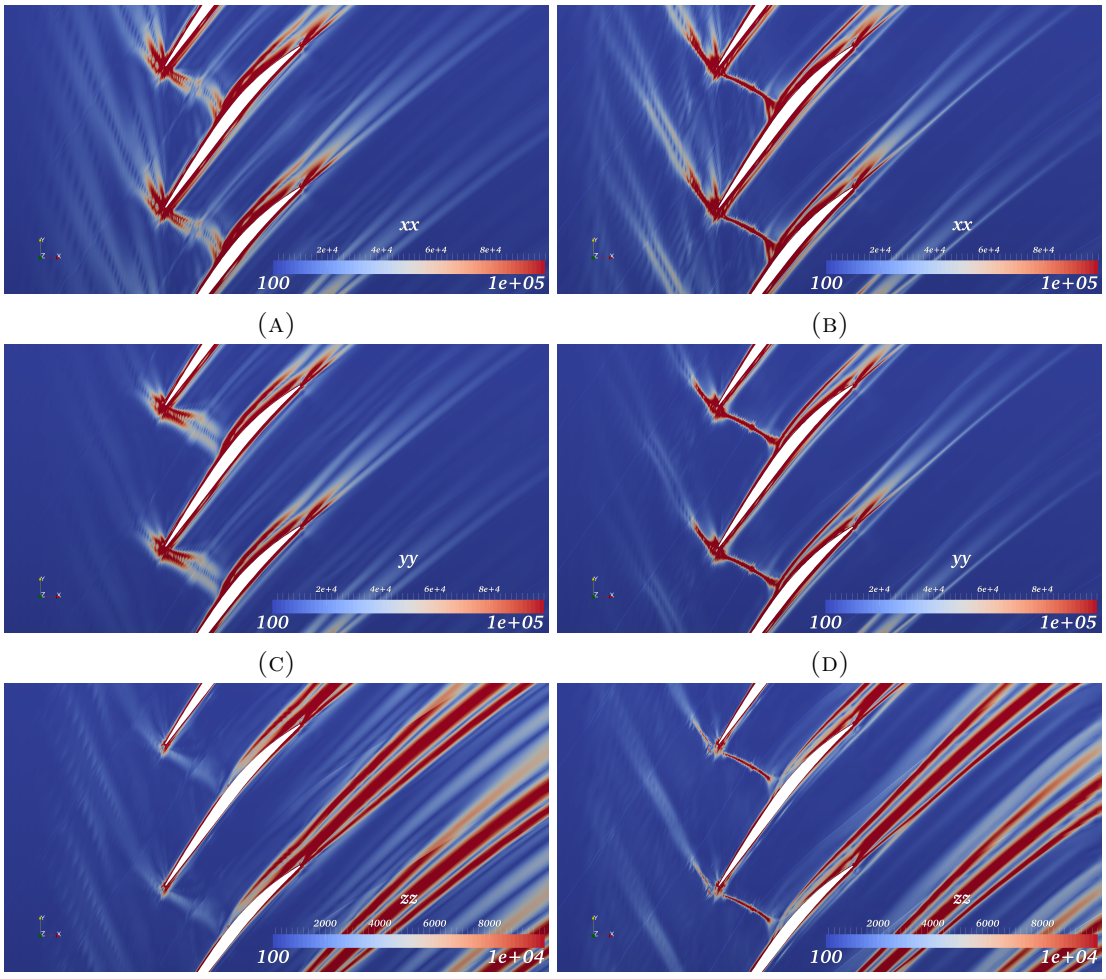
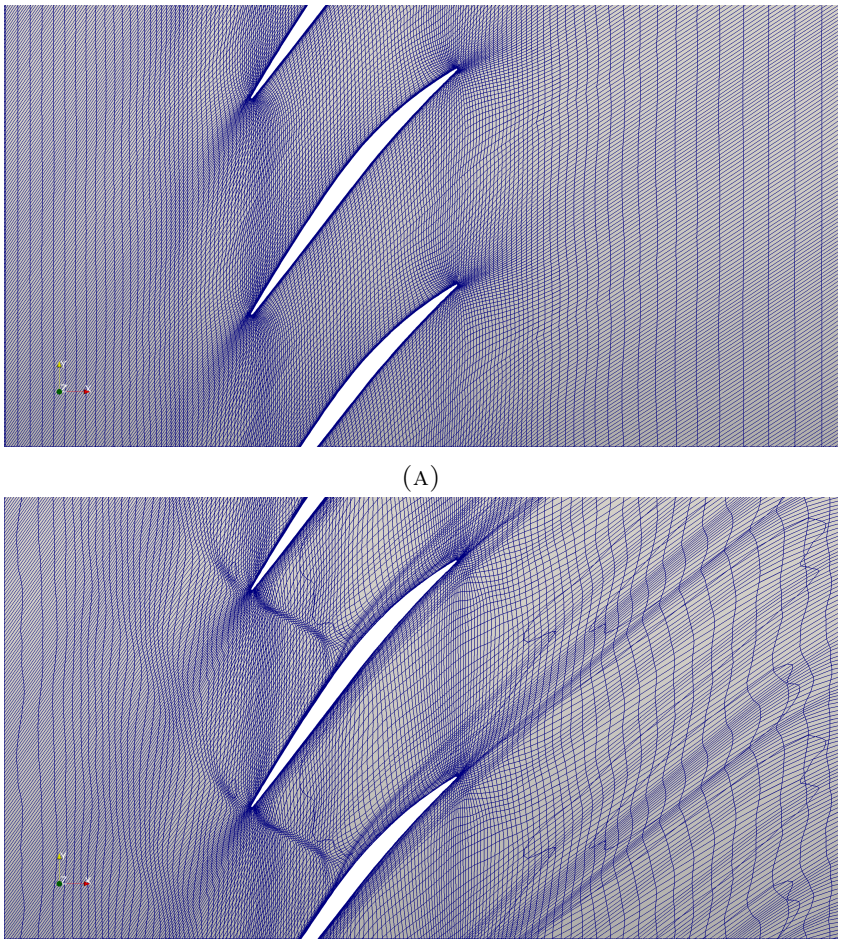


FIGURE 6.4: Reconstructed *Hessian* matrix of relative *Mach* number obtained using eq. 6.8 for the starting mesh (*LHS*) and the same grid moved 50 times (*RHS*) of *NASA Rotor* 37 at mid-span.

6.2.4 Adjoint-Related Techniques

The most popular and robust adjoint mesh adaptation technique is that developed by [9, 19, 57, 61, 64]. It has never really been shown to consistently improve the estimation of the functional of interest for turbomachinery cases. The reason behind this is threefold: firstly, the flow generally presents many complexities that change shape, intensity and interact throughout the blade height. Secondly, there is limited flexibility to adapt the mesh due to geometrical constraints being present on all sides (periodic boundaries, casing and hub, with the last two having different rotational speeds). Finally, there



(A)

(B)

FIGURE 6.5: Starting mesh (top) and the same grid moved 50 times (bottom) using the *Hessian* of relative *Mach* number of *NASA Rotor* 37 at mid-span.

are a limited amount of *CFD* codes capable of solving the adjoint problem for such cases. Therefore, the first step in this work was to develop the approach outlined by [64] and apply it to turbomachinery cases. The error estimation equation will be derived in the next paragraphs following the description in [64] (to aid the reader understand the reasoning behind them).

Assuming a fine grid flow solution to be $\mathbf{Q}_h$, where h is the characteristic cell-edge length, and the perturbation $\delta\mathbf{Q}_h$, one is able to carry out a *Taylor-Series* expansion of the integral functional of interest, f_h, and residual, $\mathbf{R}_h$, *w.r.t.* $\delta\mathbf{Q}_h$:

$$f_h(\mathbf{Q}_h + \delta\mathbf{Q}_h) = f_h(\mathbf{Q}_h) + \frac{\partial f_h}{\partial \mathbf{Q}_h}\delta\mathbf{Q}_h + ... \tag{6.11}$$

$$\mathbf{R}_h(\mathbf{Q}_h + \delta\mathbf{Q}_h) = \mathbf{R}_h(\mathbf{Q}_h) + \frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\delta\mathbf{Q}_h + ... \tag{6.12}$$

Defining a flow solution on a coarse mesh, $\mathbf{Q}_H$, the fine grid perturbation can be seen as the difference between the solution obtained by interpolating the coarse flow onto the finer mesh, $\mathbf{Q}_H^h$, and the exact one, $\mathbf{Q}_h$:

$$\delta\mathbf{Q}_h = \mathbf{Q}_H^h - \mathbf{Q}_h \tag{6.13}$$

with $\mathbf{Q}_H^h = I_H^h \mathbf{Q}_H$, where I_H^h is the prolongation operator. In general, the interpolation should be of the same order of accuracy as the flow solver [61], however, [86] indicates that second order accuracy is the minimum requirement.
Replacing 6.13 into 6.11 and 6.12 results in:

$$f_h(\mathbf{Q}_H^h) = f_h(\mathbf{Q}_h) + \left.\frac{\partial f_h}{\partial \mathbf{Q}_h}\right|_{\mathbf{Q}_H^h} (\mathbf{Q}_H^h - \mathbf{Q}_h) + ... \tag{6.14}$$

$$\mathbf{R}_h(\mathbf{Q}_H^h) = \mathbf{R}_h(\mathbf{Q}_h) + \left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\right|_{\mathbf{Q}_H^h} (\mathbf{Q}_H^h - \mathbf{Q}_h) + ... \tag{6.15}$$

to avoid computing the fine grid solution, it is necessary to assume that its flow residual is 0:

$$(\mathbf{Q}_h - \mathbf{Q}_H^h) \approx -\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\right]^{-1}_{\mathbf{Q}_H^h} \mathbf{R}_h(\mathbf{Q}_H^h) \tag{6.16}$$

this is utilised in 6.14:

$$f_h(\mathbf{Q}_h) \approx f(\mathbf{Q}_H^h) - \left.\frac{\partial f_h}{\partial \mathbf{Q}_h}\right|_{\mathbf{Q}_H^h} \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\right]^{-1}_{\mathbf{Q}_H^h} \mathbf{R}_h(\mathbf{Q}_H^h) \tag{6.17}$$

Hence it is clear where the flow adjoint solution comes into play. In fact, the two terms multiplying the fine grid residuals of the interpolated *CFD* solution form the flow adjoint vector obtained by way of the interpolated coarse-mesh flow quantities[9] :

$$\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\right]^T_{\mathbf{Q}_H^h} \Psi_h|_{\mathbf{Q}_H^h} = \left[\frac{\partial f_h}{\partial \mathbf{Q}_h}\right]^T_{\mathbf{Q}_H^h} \tag{6.18}$$

While the application of a coarse-mesh interpolated adjoint is not necessary, it is convenient in order to avoid its fine-grid computation:

[9]The minus sign in front of the *R.H.S* term has been omitted, this is considered when including the effect of *computable error correction* in equation 6.20.

$$\boldsymbol{\Psi}_h|_{\mathbf{Q}_H^h} \approx \boldsymbol{\Psi}_H^h = I_H^h \boldsymbol{\Psi}_H \tag{6.19}$$

Finally, by this relationship, it is possible to define an enhanced functional evaluation:

$$f(\mathbf{Q}) \approx f_h(\mathbf{Q}_H^h) - \boldsymbol{\Psi}_H^h|^T \mathbf{R}_h(\mathbf{Q}_H^h) \tag{6.20}$$

As it can be seen, the improved estimate does not require any fine grid computation, relying only on the interpolated flow and adjoint solutions from the coarse grid. The term that is subtracted from the functional computed on the fine mesh, using the interpolated coarse solution, is generally indicated as *computable correction*.
The very same authors, in [9, 19, 57, 61, 64], argue that the previously derived error should not be employed for the mesh adaptation, as it may be easily calculated and used to achieve an improved estimate of the functional of interest. Instead, they derive the the *non-computable correction* formulation, that is nothing else other than the inaccuracies introduced into the *computable correction* by the use of an interpolated adjoint solution. Before formulating the mesh adaptation error indicator they developed, the derivation of it should be considered, as shown in [64]. From the definition of the adjoint relations, its residual vector $\mathbf{R}(\boldsymbol{\Psi})$ can be written as:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T \boldsymbol{\Psi} = -\frac{\partial f}{\partial \mathbf{Q}}^T \Rightarrow \mathbf{R}(\boldsymbol{\Psi}) = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T \boldsymbol{\Psi} + \frac{\partial f}{\partial \mathbf{Q}}^T \tag{6.21}$$

Again, it is possible to use the interpolated adjoint solution to form a perturbation of the exact fine mesh one:

$$\delta\boldsymbol{\Psi} = \boldsymbol{\Psi}_H^h - \boldsymbol{\Psi}_h \tag{6.22}$$

Formulating the fine mesh adjoint residuals utilising the interpolated values and assuming $\mathbf{R}(\boldsymbol{\Psi}_h) = 0$:

$$\begin{aligned} \mathbf{R}(\boldsymbol{\Psi}_H^h) &= \frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}^T (\boldsymbol{\Psi}_h + \delta\boldsymbol{\Psi}_h) + \frac{\partial f_h}{\partial \mathbf{Q}_h}^T \\ &= \cancel{\left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}^T \boldsymbol{\Psi}_h + \frac{\partial f_h}{\partial \mathbf{Q}_h}^T \right)}^{\,0} + \left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}^T \delta\boldsymbol{\Psi}_h \right) \\ &= \frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}^T \delta\boldsymbol{\Psi}_h \end{aligned} \tag{6.23}$$

Hence, using the exact fine mesh adjoint $\mathbf{\Psi}_h$, it is possible to formulate the functional perturbation as:

$$\begin{aligned}
\delta f_h &\approx \mathbf{\Psi}_h|^T \mathbf{R}_h(\mathbf{Q}_H^h) \\
&= (\mathbf{\Psi}_H^h - \delta\mathbf{\Psi}_h)|^T \mathbf{R}_h(\mathbf{Q}_H^h) \xrightarrow{\mathbf{R}_h(\mathbf{Q}_H^h) \approx \frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\delta\mathbf{Q}_h} \\
&\approx \underbrace{\mathbf{\Psi}_H^h|^T \mathbf{R}_h(\mathbf{Q}_H^h)}_{\text{eq. 6.20}} - \underbrace{\delta\mathbf{\Psi}_h|^T \frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}}_{\text{eq. 6.23}} \delta\mathbf{Q}_h \\
&\approx \mathbf{\Psi}_H^h|^T \mathbf{R}_h(\mathbf{Q}_H^h) - \underbrace{\mathbf{R}(\mathbf{\Psi}_h^H)|^T \delta\mathbf{Q}_h}_{\substack{\text{non-computable} \\ \text{error correction}}}
\end{aligned} \tag{6.24}$$

As the derivation in 6.24 has shown:

$$\delta\mathbf{\Psi}_h|^T \mathbf{R}_h(\mathbf{Q}_H^h) = \mathbf{R}(\mathbf{\Psi}_h^H)|^T \delta\mathbf{Q}_h \tag{6.25}$$

This shows the *duality* present between the flow and adjoint solvers. It should be noted, however, that this is true only where nonlinear terms are negligible[10].
Defining *high*- and *low*-order interpolation functions, from coarse to fine grids, as Υ and υ, respectively, the error sensor (ϵ) proposed by [9, 19, 57, 61, 64] is:

$$\epsilon_i = \frac{1}{2}\sum_j \left\{ \left| [\mathbf{R}(\upsilon\mathbf{\Psi}_H)_h]_j^T [\Upsilon\mathbf{Q}_H - \upsilon\mathbf{Q}_H]_j \right| + \left| [\Upsilon\mathbf{\Psi}_H - \upsilon\mathbf{\Psi}_H]_j^T [\mathbf{R}(\upsilon\mathbf{Q}_H)_h]_j \right| \right\} \tag{6.26}$$

Where i is the coarse mesh node index and the summation is carried out over all fine mesh nodes contained in the i^{th} coarse control volume. The averaging of the two versions of the *non-computable error correction* aims to include any non-linearities and the different prolongation operators are used to mimic a higher and lower order solution. In particular, the latter replaces the perturbation in the flow and adjoint solutions:

[10] This is due to the previous linearisation:

$$\delta\mathbf{R}_h \approx \frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}_h}\delta\mathbf{Q}_h$$

and

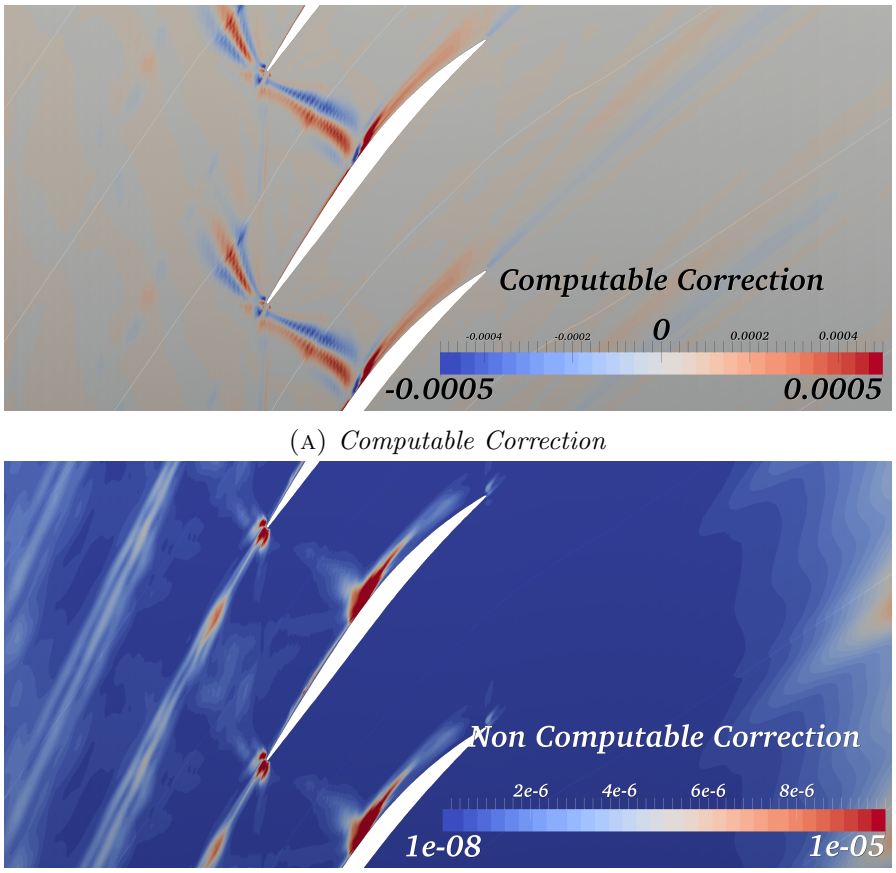
$$\delta f_h \approx \frac{\partial f_h}{\partial \mathbf{Q}_h}\delta\mathbf{Q}_h$$

$$\delta\mathbf{Q}_h = \Upsilon\mathbf{Q}_H - \upsilon\mathbf{Q}_H \tag{6.27}$$

$$\delta\mathbf{\Psi}_h = \Upsilon\mathbf{\Psi}_H - \upsilon\mathbf{\Psi}_H \tag{6.28}$$

To calculate these mesh error estimation techniques, modifications to flow and adjoint solvers are required so their residuals may be written out to file. More importantly, though, linear and quadratic interpolation processes are required. While the *computable* correction needs the latter to be estimated more accurately, the *non-computable* version would require both kinds of prolongation operators so that the deviation from a linear behaviour on the fine grid may be determined. Once the sensor is computed on the fine grid, a restriction operator is required to determine the coarse mesh error map and be able to adapt the relative mesh. The three interpolation operators will be the topic of section 6.2.5.

Examples of the *computable* and *non-computable* error maps are shown in figure 6.6. It is clear that there are similarities between the two. As in the case for the reconstructed *Hessian* matrix in figure 6.4, the shock and its interaction with the boundary layer are highlighted. However, in the *non-computable* case, the shocks are not the highest region of error. In fact, strong emphasis is on the interaction of these with other parts of the flow. In particular, its intersection with the adjoint *reversed wake* and the region where separation occurs have the largest error. Effectively, any flow feature that crosses regions of high sensitivity, as indicated by the adjoint, should be well resolved to improve the functional estimate. It is also interesting to note the low error present on the pressure side of the blade. Both adjoint sensor maps are in agreement, unlike the *Hessian*-based case that does highlight a small layer on the pressure side of the blade wall. A characteristic that has been noted throughout this work relates to the wake. Neither type of adjoint-related quantity will strongly highlight this region of the flow. This is due to the nature of the adjoint convection, as discussed in section 5.5. In fact, from a intuitive point of view, errors caused upstream will be convected downstream by the *NS* relations. These may then be amplified further, but not due to local grid issues. Therefore targeting the upstream inaccuracies in the flow will aid capturing the downstream features. A final note concerns the *computable-correction* in figure 6.6a. In fact, a change is clearly identified in blue/red colour where the error is high. This is due to the nature of *Hydra*. In fact, it is not a *TVD* scheme, and small under- and over-shoots will appear across discontinuities and therefore in the flow residuals that are selected to calculate this sensor map. On the other hand, these are not visible in 6.6b, as in eq. 6.26 as the absolute value of the two components is preferred.



(A) *Computable Correction*

(B) *Non-Computable Correction*

FIGURE 6.6: Adjoint error maps at mid-span of *NASA Rotor* 37 for mass averaged adiabatic efficiency evaluated between inlet and outlet.

6.2.5 Smooth Flow & Adjoint Interpolation

The first point to be noted is that the fine grid will be formed by completely refining the coarse mesh. Therefore it will have approximately 8-times the number of nodes of the original grid, with every new point either splitting a coarse edge, face or cell. The linear interpolation technique is straightforward: for every coarse mesh edge mid-point simply average the end node values. For face and cell centroids, similar averaging is carried out. This process avoids any oscillatory behaviour and is perfectly *monotone*.
On the other hand, the quadratic interpolation is more involved. In fact, the coarse mesh gradients must be computed to include their contribution in the prolongation onto the fine mesh, as shown in figure 6.7. This is done employing the same process as in section 6.2.2, thus allowing a smoother derivative evaluation.
The procedure employed to determine the mid-edge values is an *Hermitian* interpolation, as used by [65, 72]. For what concerns quadrilateral face and hexahedral cell centroids, multidimensional *Lagrangian* elements were adopted ([187]). It should be noted, that in this case the *ghost nodes* procedure was not utilised, as the code is sequential and

increasing further the memory requirements can cause issues with finer grids.

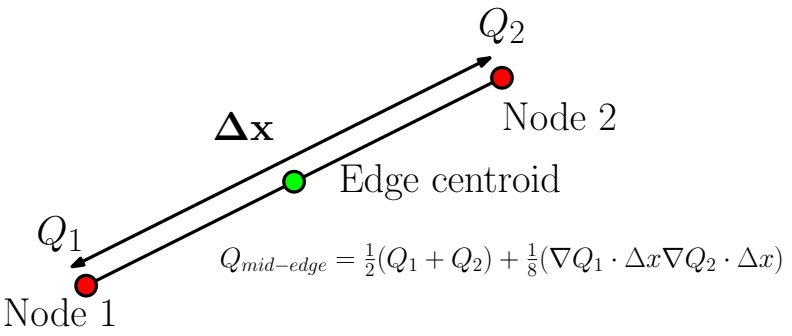


FIGURE 6.7: Quadratic flow and adjoint variable interpolation along an edge.

Unfortunately, though, there is still a degree of noise in the gradients that cause negative density or turbulent viscosity to appear. This is particularly real in the near wall regions where there is a strong curvature, such as the *LE* and *TE* of the blade. To avoid non-physical values appearing, a limiter was inserted in the code. This would check if any of the fine mesh node values was greater or smaller than all the coarse mesh nodes parameters determining them, and if this was the case it would reduce the interpolation to linear. Interestingly, it was noted that in general, the adjoint quadratic interpolation always requires less corrections *w.r.t.* that of the flow. The reason behind this behaviour is thought to be due to the order of magnitude of the adjoint variables: more often than not they are greater than $10^2 \sim 10^3$ unlike the flow parameters and therefore any numerical noise will have lesser impact. Another important observation concerned the gradient computation approach described in section 6.2.2: in fact employing this method rather than *GG* reduced the number of nodes requiring lower order correction, thus indicating the increased smoothness achieved.

Finally, the restriction was carried out by adopting a control volume weighted approach. However, unlike the procedure employed in *Hydra* and discussed in section 4.7, the fine mesh control volume portion contained in the coarse grid one was also taken account of. This means that, for e.g., the contribution of a value belonging to a coarse mesh edge centroid, would be halved before being added to the edge's end points, since its control volume is shared between them.

6.3 Novel Anisotropic & Edgewise Adjoint Grid Error Calculation

6.3.1 A Simplified Adjoint Edgewise Computable & Non-Computable Error Refinement

As described in section 2.3.2, the adjoint adaptation technique is cell-based. In fact, once the error is evaluated nodewise on the fine grid, this is generally restricted to the coarse mesh points (as sketched in figure 6.8). This quantity is then adapted in a cell-based refinement algorithm. While this is not the most efficient grid modification strategy, it does have the advantage of having a smoother error map for the adaptation, as in general restriction will help in this sense.

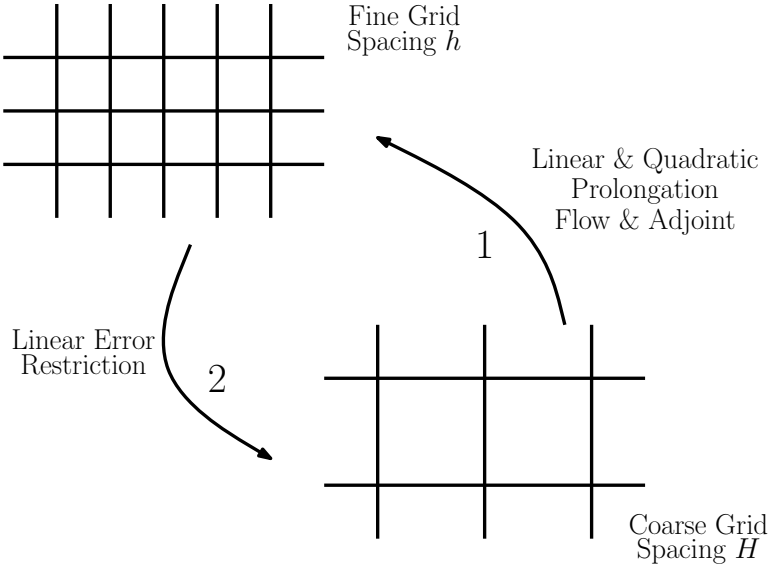


FIGURE 6.8: Original adjoint error estimation process schematic.

As the fine grid is effectively obtained by splitting all coarse grid edges/faces/cells, part of the fine mesh points will match coarse grid edges, as shown in figure 6.9. Therefore, once the error has been evaluated according to eq. 6.20 or 6.26 (depending on which one is chosen for the refinement), each fine mesh node will have a value associated to it. For this reason it is feasible to map the coarse mesh edge ID with the fine grid node index splitting it in half. Consequently, it will be possible to consider the error evaluated on the latter to decide whether or not to refine that edge. This modification to the procedure actually allows to employ an edge-based refinement algorithm to adapt the mesh. Apart from being able to apply an efficient grid enrichment strategy to modify the mesh, this also averts the necessity to compute the actual restriction of the error between grids.

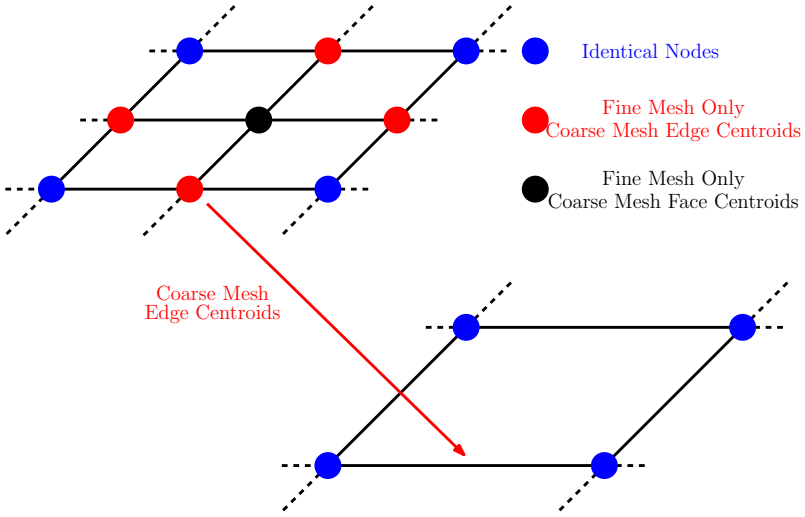


FIGURE 6.9: Matching of the coarse mesh edges with the fine grid nodes.

6.3.2 Efficient Adjoint Edgewise Error Estimation

One of the main issues with adjoint error estimation and adaptation techniques concerns the amount of data and process complexity required to be able to modify the mesh. In fact, for *FVM* flow solvers, a fine grid must be obtained by uniformly refining the initial one and a linear and quadratic prolongation of both the flow and adjoint solutions are needed. Moreover, the *NS* and dual solvers need to be run on the fine grid to determine the linearly and quadratically interpolated flow residuals along with those of linearly estimated adjoint solution. Once these are available, the error may be computed and restricted to the coarse grid where the refinement will take place. The process is particularly problematic for multiple functional error estimation, as separate processes are required for each. To avoid the burden of having to compute the error in the manner described in section 6.2.4, in this work an approach to mimic it without adopting any fine grid was devised.

Firstly, to be able to carry out an efficient process, the error would have to be computed in an edgewise manner and remain smooth. To do so, eq. 6.26 would be replicated by employing edgewise residuals of the coarse mesh flow and adjoint solutions, instead of fine grid nodal values. Additionally, to substitute the difference between linearly and quadratically prolonged flow and adjoint quantities, interpolated values along an edge would be used, that is a process similar to that described in section 6.2.5. Therefore, eq. 6.26 can be re-written as:

$$\epsilon_e = \frac{1}{2}\left\{\left|[\mathbf{R}^e(\mathbf{\Psi}_H)]^T[\Upsilon\mathbf{Q}^e_H - \upsilon\mathbf{Q}^e_H]\right| + \left|[\Upsilon\mathbf{\Psi}^e_H - \upsilon\mathbf{\Psi}^e_H]^T[\mathbf{R}^e(\mathbf{Q}_H)]\right|\right\} \tag{6.29}$$

where:

- ϵ_e is the coarse mesh edgewise error;
- $\mathbf{R}^e$ is the flow or adjoint edgewise residuals operator;
- Υ is the quadratic interpolation along an edge;
- υ is the linear interpolation along an edge;
- $\mathbf{Q}_H$ is the coarse mesh flow solution at each node;
- $\mathbf{\Psi}_H$ is the coarse mesh flow adjoint solution at each node.

An interesting point concerns the way these edgewise quantities are evaluated. For the *NS* case, only an extra variable would be required in the source code to store the residual contribution due to viscous and inviscid quantities along an edge. Effectively, these correspond to edge fluxes evaluated employing the techniques described in sections 4.5.1 (inviscid contribution) and 4.5.2 (viscous quantities). For what concerns the adjoint, things are different. In fact, as discussed in section 5.7, the gather and scatter operations are reversed. Therefore in this case, the edge residual contribution is different for each node:

$$\mathbf{R}(\mathbf{\Psi})_L = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T \mathbf{\Psi}_L \tag{6.30a}$$

$$\mathbf{R}(\mathbf{\Psi})_R = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T \mathbf{\Psi}_R \tag{6.30b}$$

i.e. the right and left nodes of the edge would have a different update for every iteration. For this reason, the edgewise contribution of the adjoint was considered to be the average of the two, thus allowing extra degree of smoothing.

6.3.3 Riemannian Metric for Adjoint Functional Sensitivity to Grid Coordinates

The discrete adjoint solver requires also a *mesh adjoint* to map the grid sensitivities to the actual geometry surface. As previously discussed in detail in section 5.10 this process will compute $\mathbf{\Psi}_g = \frac{df}{d\mathbf{x}}$, i.e. the functional sensitivity to the grid coordinates. This quantity is effectively indicating where in the grid the functional is most sensitive to coordinate perturbations. As part of a mesh adaptation procedure to improve the

estimate of a performance quantity, better estimates can be achieved if nodes are clustered in regions where the most significant variations are present. For this reason, the *mesh adjoint* quantity was employed in the procedure outlined in section 6.2.3. That is, the *Hessian* matrix of second order derivatives was determined by applying the gradient operator to $\frac{df}{d\mathbf{x}}$. This allowed to determine a *Riemannian* metric that could then be employed in the spring-stiffness approach (described in detail in section 6.5.3) to cluster the nodes in most important regions relative to that functional.

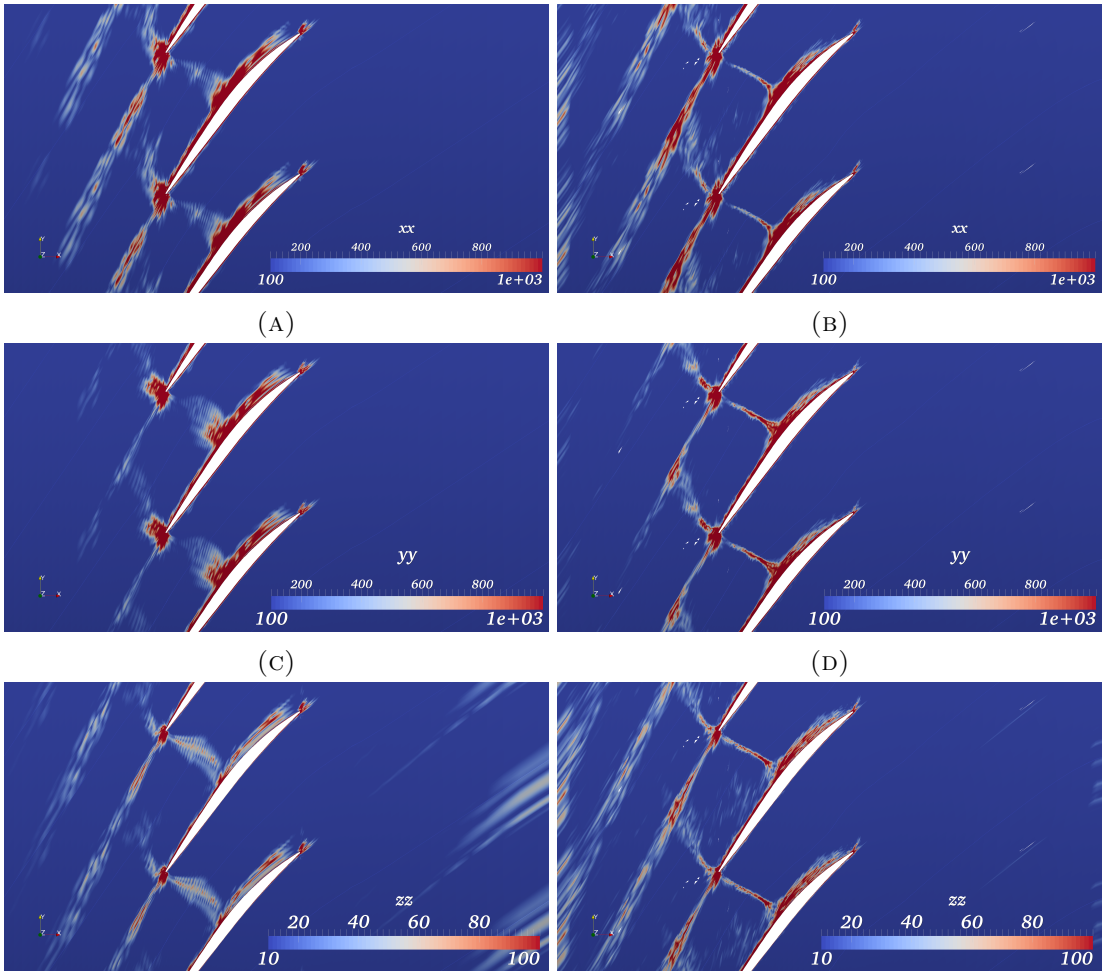
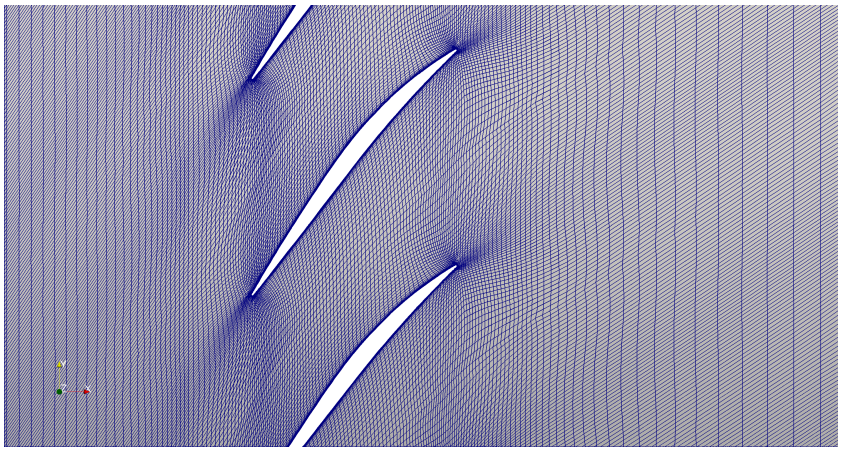
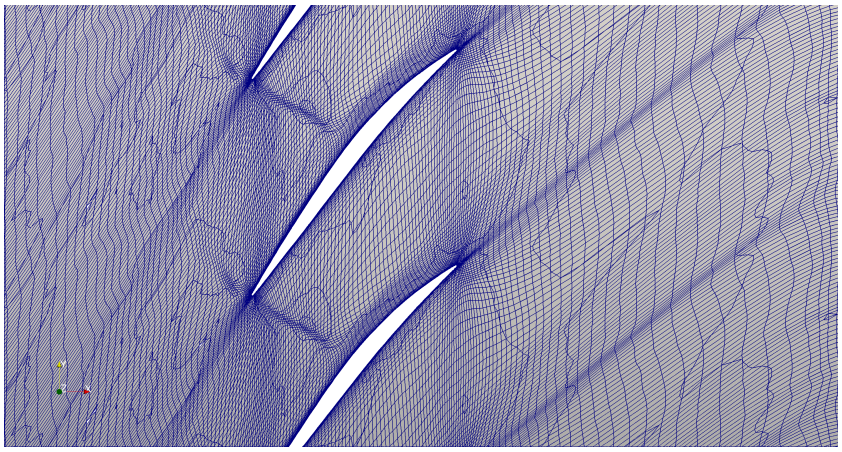


FIGURE 6.10: Reconstructed *Hessian* matrix of $\frac{df}{dx}$ of mass averaged adiabatic efficiency between inlet and outlet obtained using eq. 6.8 for the starting mesh (*LHS*) and the same grid moved 50 times (*RHS*) of *NASA Rotor* 37 at mid-span.

Examples of the reconstructed *Hessian* matrix at mid-span, for adiabatic efficiency evaluated between inlet and outlet, for *NASA Rotor* 37 are shown in figure 6.10. In particular, figures 6.10a, 6.10c and 6.10e represent eq. 6.8 relative to the starting grid shown in 6.11a, while figures 6.10b, 6.10d and 6.10f are the reconstructed *Hessian* matrix diagonal components of the mesh shown in 6.11b, achieved by applying 50 mesh movement iterations. As it can be clearly seen, there are a variety of adjoint and flow features



(A) Starting Grid.



(B) Application of Mesh Movement 50 times.

FIGURE 6.11: Example of adaptation using $\frac{df}{dx}$ of mass averaged adiabatic efficiency between inlet and outlet at mid span of a 670k node *NASA Rotor* 37 mesh.

being highlighted. This is to be expected, as, for example, the efficiency will have a higher sensitivity to the shock resolution. Interestingly, the flow wake is not strongly highlighted: the adjoint effect is that of weighting the upstream regions that cause certain features to form downstream. For this reason, it will target the shock/boundary layer interaction generating separation, rather than the wake itself. As expected, the adjoint *reversed wake* shows high sensitivity. This is indicating that perturbing node coordinates in this region will cause a stronger change in efficiency. Moreover, it can be seen how this high adjoint sensitivity region upstream interacts with the propagation of the shocks. The main part of the domain highlighted by all three components is the *LE*. In fact, at this point the shock actually forms and propagates in both directions. When this hits the suction side of the adjacent blade, it will interact with the boundary layer and cause flow separation. This too is strongly visible in all components of figure 6.10. Comparing the starting grid with the adapted one, it can clearly be seen how the nodes have been relocated towards the parts of the flow showing highest sensitivity. Moreover,

the second derivatives evaluated on the adapted grid show a much sharper capturing of the main flow and adjoint features.

6.4 *JST* Switch Edge-Based Adaptation

As discussed in detail in Chapter 4, *Hydra* is a flow solver employing the *matrix-dissipation* version of the *JST* flux reconstruction scheme. Therefore along each edge a switch is determined in order to blend second and first order solutions to reduce oscillatory behaviour. This quantity does provide a very interesting parameter to be used for an edge-based refinement algorithm. In fact, by storing the edge computation of the switch while running the code, it would be possible to determine along which of these the solution does not have a second order reconstruction. In other words, it allows to identify in a simple manner where part of the flow discontinuities are. According to the formulation in equation 4.22, this is devised to detected features such as shocks, and therefore would be inadequate for turbulent flows as those present in turbomachinery cases. However, it was noted that for the *SA* turbulence model equation, *Hydra* has a different formulation. In this case, it not only contains static pressure differences along the edge, but also relative to the turbulent viscosity. This allows the switch to contain extra information on the location of some of the main discontinuities present in the flow. On the down side, the boundary layer is not captured unless it is turbulent.

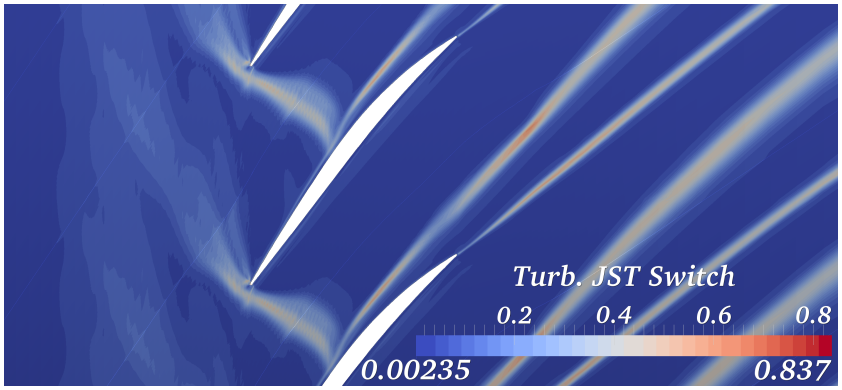


FIGURE 6.12: *SA* turbulence model *JST* switch at mid-span for *NASA Rotor* 37.

An example *SA-JST* switch is shown in figure 6.12. As it can be seen, it highlights important regions of the flow such as the shock along with its propagation and the interface between the wake and the bulk of the flow. While this represents an interesting parameter to employ in a mesh adaptation strategy, it may only be utilised in flow solvers containing this particular flux reconstruction scheme.

6.5 Mesh Adaptation Grid Mechanics

6.5.1 Mesh Refinement

In this work, the main refinement strategy employed is edge-based, apart from the combined feature and adjoint error estimation procedure where a cell marking strategy is used. The reason behind choosing an edge-based refinement is straightforward: it is the most efficient *w.r.t.* cell and face related strategies.
The refinement algorithm employed in *MP* started by splitting all cells into triangular or quadrilateral faces forming them and consequently marking all edges meeting the error threshold. The code would repeatedly loop over faces determining whether edges belonging to them had to be split, and if so it would divide the face according to one of the templates as shown in figure 6.13.

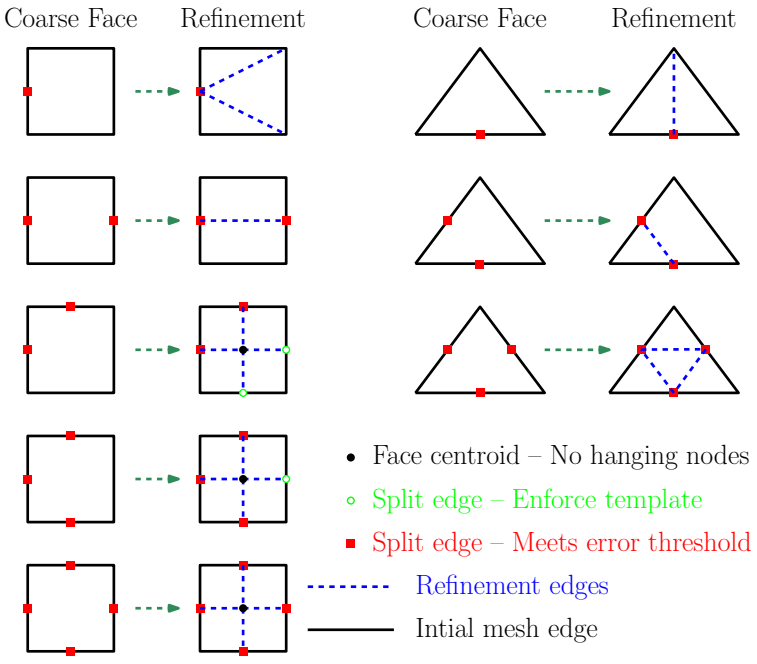


FIGURE 6.13: 2D face refinement templates.

As it can be seen, while triangular faces are more flexible, quadrilateral ones will cause a degree of over-refinement. In fact, when two edges sharing a node or three edges in a face are marked, all four will be split. This is consistent with directional refinement, as in either case the error is indicating that both directions of the face need to be refined. This process is also advantageous in coding terms as it simpler to implement and relatively robust. Finally, the faces of each cell are re-assembled with the new connectivity determined if a matching cell-splitting template is available, otherwise extra edges would be marked. An example of hexahedral edge-based refinement is shown in figure 6.14.

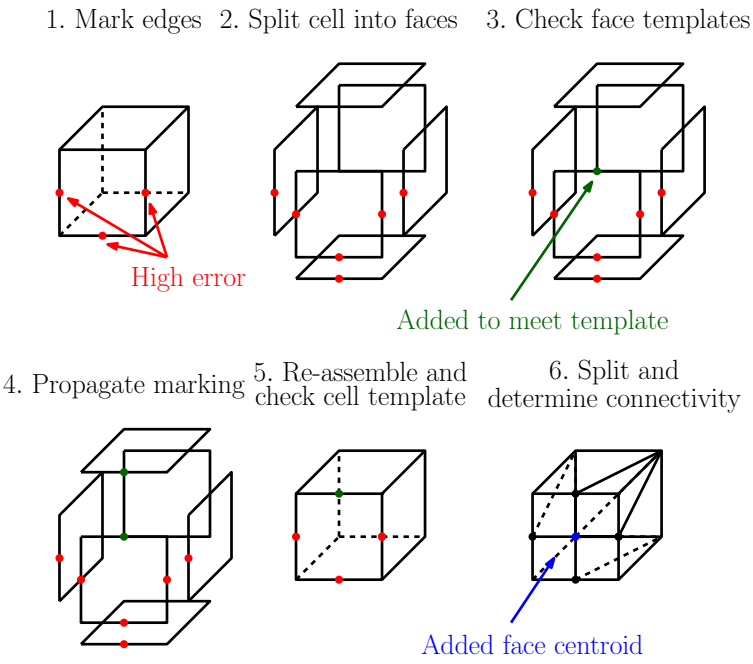


FIGURE 6.14: Example of edge-based cell splitting.

An issue that presented itself when adopting this approach, concerns the over-refinement of hexahedral cells. In fact, due to geometry constraints and many 3D flow features present in turbomachinery, it is very easy to incur in situations where a small amount of marked edges will propagate the nodal addition, unless grid edges are perfectly aligned with the flow *anisotropy*. This problem will mainly present itself in early adaptation stages, as the mesh will be very similar to the user generated one, who will have little knowledge of the flow features exact location. To overcome this issue, the feature/adjoint-based mesh movement can be employed. This will allow grid smoothing and alignment towards complex flow regions.

One of the biggest problems encountered when refining hexahedral grids and avoiding *hanging nodes*, relates to the interface patches between finer and un-refined regions. In fact, to match the connectivity, tetrahedra, pyramids and prisms need to be introduced. These can cause severe problems in the adaptation procedure, in terms of poor convergence, due to abrupt change in cell size and flux directionality, or even negative cells in later refinement steps. The former point is particularly real if the hybrid cells appear in the vicinity of complex flow regions. Hence, the over-refinement issue previously discussed can be helpful in pushing these patches further away from important flow areas.

Finally, a brief description of the cell-based refinement introduced in the combined feature and adjoint adaptation is due. The simplest cell division approach was not adopted in this case. The reason behind this is the poor quality cells that would result. In fact, the simplest and most economical way of splitting an hexahedron is to place a node at its centroid and then form six pyramids, as shown in figure 6.15.

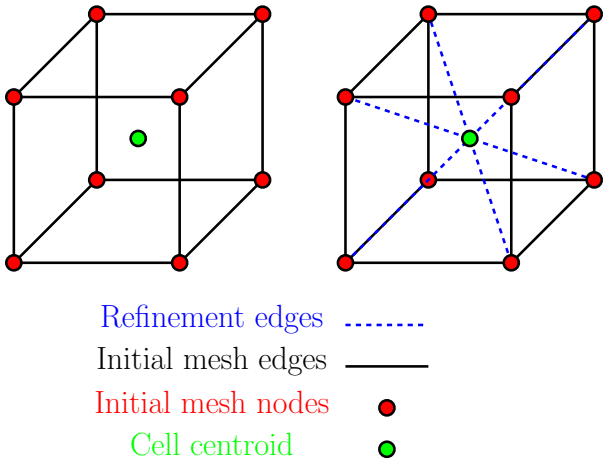


FIGURE 6.15: Example of the simplest form of cell splitting applied to an hexahedron.

In this work, cell splitting was performed by marking all edges belonging to a cell meeting the error threshold, thus employing a subset of the edge-refinement templates. The resulting possible combinations are shown in figure 6.16.

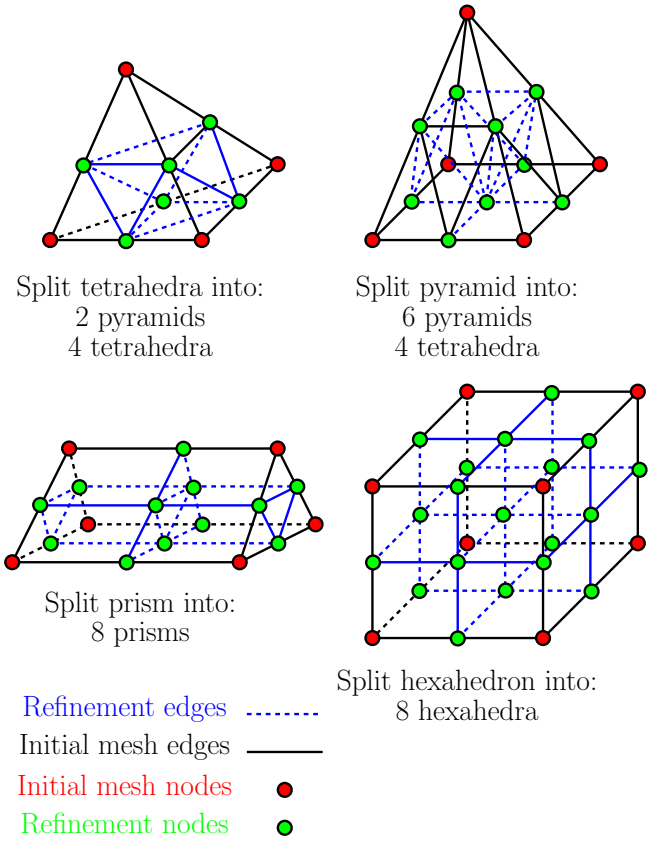


FIGURE 6.16: Cell splitting templates.

The careful reader, will have noted that cells not marked for refinement may have edges needing to be split if one of the neighbouring cells is going to be refined. In these

cases, the previously described algorithm for edge refinement can be employed to form interfaces between the initial coarse mesh and the refined patches.
An additional capability in the code is that of propagating wall-normal refinement by enforcing extra prismatic or hexahedral elements (figure 6.17). The number of off-wall layers forced to prisms or hexahedra is set by the user. Due to the unstructured data format, the code determines the number of off-wall layers requiring special treatment using the cell to face connectivity.

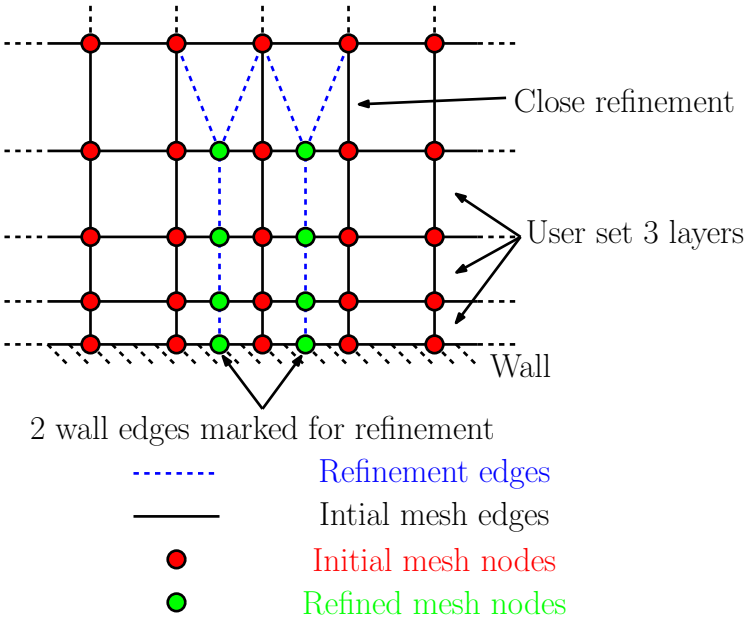


FIGURE 6.17: Wall normal refinement propagation.

6.5.2 Surface Reconstruction

The approach devised in order to improve the geometry approximation after refinement is based on the work of [188]. The first step is to determine the normals of all surface elements, that [188] assumes to be triangular. For such cases, the area vector is determined by using the cross product of two edges, making sure the order is correct so the vector is pointing out of the surface, as shown in figure 6.18.

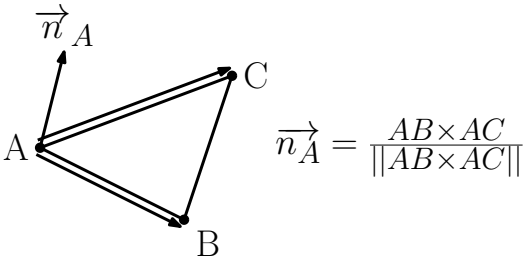


FIGURE 6.18: Triangle norm calculation.

Every node's norm will be determined by accumulating all surface vectors of the triangles they belong to:

$$\overrightarrow{n}_A = \frac{\sum_{i=1}^{N°tri} \overrightarrow{n}_i}{\|\sum_{i=1}^{N°tri} \overrightarrow{n}_i\|} \tag{6.31}$$

An important modification to the area vector determination process is required at ridges. In fact, at these points, multiple norms may have to be stored once they have been identified. As shown in figure 6.19, at the discontinuity between the compressor pressure side and tip, two different norms must be computed to appropriately reconstruct both surfaces.

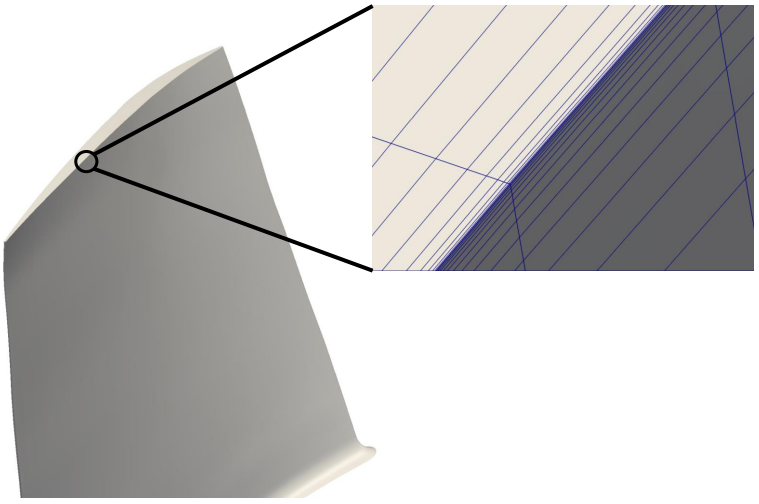


FIGURE 6.19: Ridge example: *NASA Rotor* 37 geometry discontinuity.

To identify locations where extra care must be taken, each surface element norm was compared against the neighbouring one. If the dot product between the two element's vectors was greater than arccos(20°) then the two were marked as being at a discontinuity. The choice of 20° was determined by tests that were carried out during the development process.

In this work, the grid was fully hexahedral to start, with triangular elements appearing only once refinement had taken place. Therefore it would be necessary to determine the norm of quadrilateral elements. These are more complicated than their triangular counterpart, since two surface vectors may be defined. These could be quite different (and possibly wrong) according to the diagonal chosen to split the element into two triangles. An example of such a situation is reported in figure 6.20. To start with, it was assumed that all the surface quadrilaterals were sufficiently smooth to be able to arbitrarily choose the same splitting diagonal for all elements.

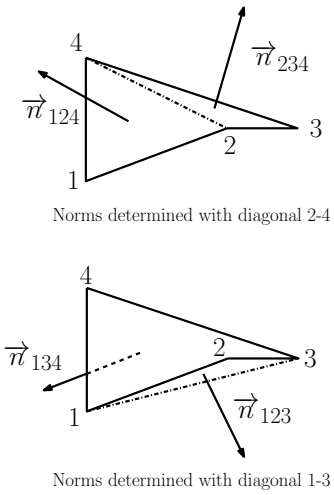


FIGURE 6.20: Surface quads norms ambiguity.

Before moving onto the next step of the geometry reconstruction, an important subtlety has to be mentioned. The periodic boundary will have part of the surface elements connected to a node one each side of it, therefore correct computation of the norm should account for all their contributions, including the effect of the rotation matrix (see section 4.9.2). Additionally, checks for ridges appearing across these boundaries should be carried out.

Finally, the node projection onto the geometry surface can be evaluated. This is done employing the quadratic approach suggested by [188]. First of all the edge curvature is determined by way of (see figure 6.21):

$$\overrightarrow{r}_1 = \|(\overrightarrow{x}_2 - \overrightarrow{x}_1)\| \frac{\overrightarrow{n}_1 \times ((\overrightarrow{x}_2 - \overrightarrow{x}_1) \times \overrightarrow{n}_1)}{\|\overrightarrow{n}_1 \times ((\overrightarrow{x}_2 - \overrightarrow{x}_1) \times \overrightarrow{n}_1)\|} \tag{6.32}$$

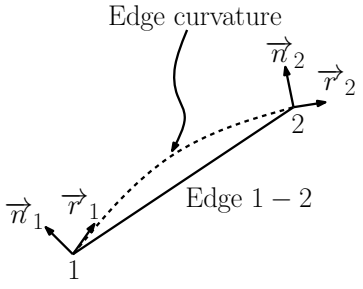


FIGURE 6.21: Edge curvature reconstruction.

The determined vector for each end of the edge is employed in an *Hermitian* polynomial:

$$\overrightarrow{x} = (1-\xi)^2(1+2\xi)\overrightarrow{x}_1 + \xi(1-\xi)^2\overrightarrow{r}_1 + \xi^2(3-2\xi)\overrightarrow{x}_2 - \xi^2(1-\xi)\overrightarrow{r}_2 \tag{6.33}$$

As the refinement will place the point mid-way across the edge $\xi = 0.5$, therefore equation 6.33 becomes:

$$\overrightarrow{x} = \frac{1}{2}(\overrightarrow{x}_1 + \overrightarrow{x}_2) + \frac{1}{8}(\overrightarrow{r}_1 - \overrightarrow{r}_2) \tag{6.34}$$

Once the process has been carried out, negative volume treatment will be necessary as the boundary layer mesh is so thin in the wall normal direction that projected refinement nodes will intersect it. In this case a linear spring mesh perturbation was applied. The perturbation is actually the displacement of the wall node from the mid-edge point to the curvature-approximated position. For this reason, methods devised for optimisation strategies or moving geometries for unsteady simulations may be employed[11]. In this case a linear-spring stiffness approach was adopted, similar to that devised by [189]. As described by [113], this maps the grid to a spring network, where the spring coefficient $\widetilde{\kappa}$, is equal to the inverse of the edge length. Indicating as $\Delta\overrightarrow{x}_i$ the displacement of every node, it is possible to write the force exerted on every grid point as:

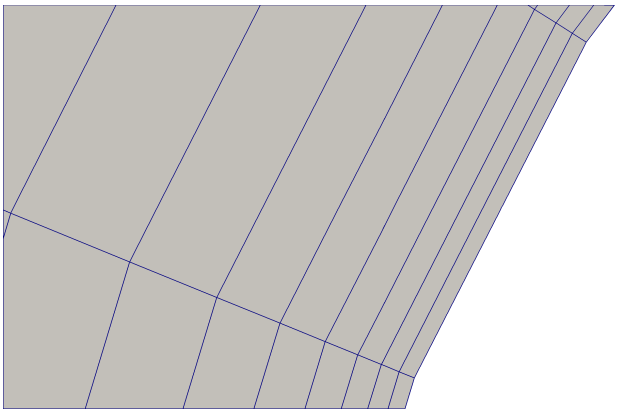
$$\overrightarrow{F}_i = \sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}(\Delta x_j - \Delta x_i) \tag{6.35}$$

Where the summation is carried out over neighbours of node i as determined by the edge connectivity. Consequently, the system is solved in an iterative fashion to achieve static equilibrium at each node (i.e. $\overrightarrow{F}_i = 0 \ \ \forall i$). Therefore, the iterative relation to be solved is:

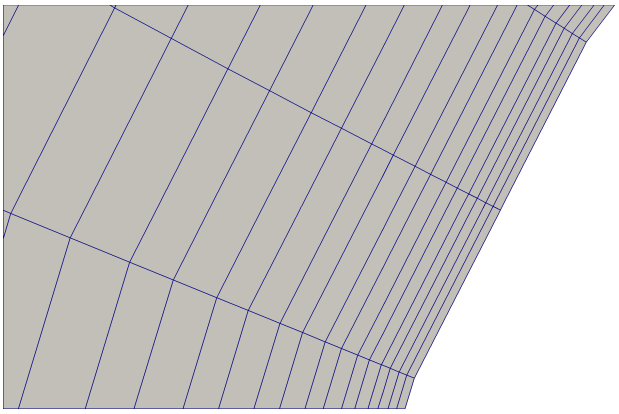
$$\begin{aligned} 0 &= \sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}(\Delta x_j - \Delta x_i) \\ \sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}\Delta x_j &= \sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}\Delta x_i \\ \sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}\Delta x_j &= \Delta x_i \sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij} \\ \Delta x_i &= \frac{\sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}\Delta x_j}{\sum_{j \in \mathbb{E}_i} \widetilde{\kappa}_{ij}} \end{aligned} \tag{6.36}$$

In this work, the functionality to perturb the mesh employed had been initially coded in *Hydra* to be able to determine the *mesh adjoint* required in the full sensitivity chain. The *conjugate gradient* iterative scheme is employed to solve the system.

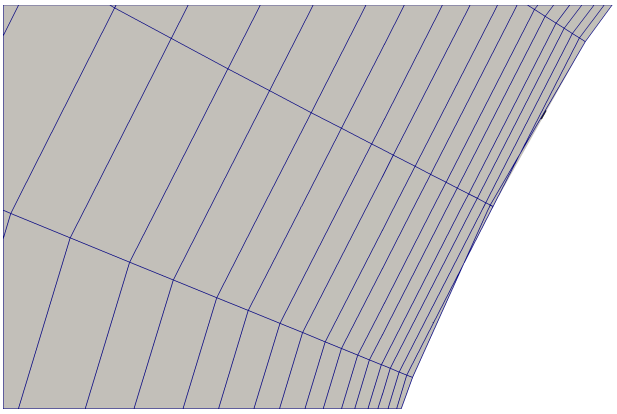
[11] In an optimisation loop the geometry will be perturbed several times. Therefore the grid requires either re-generation or modification in the appropriate regions so it may be simulated again.



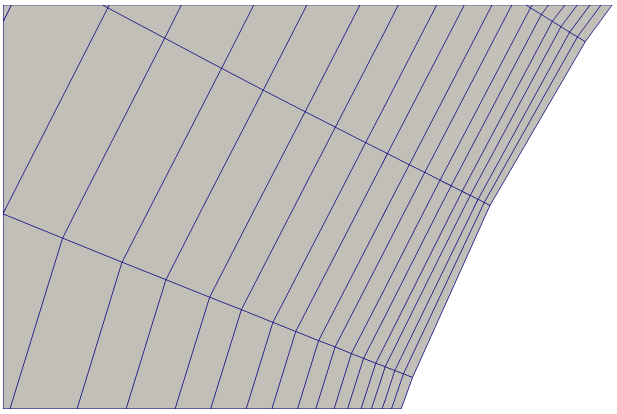
(A) Starting grid.



(B) Refined with no surface reconstruction.



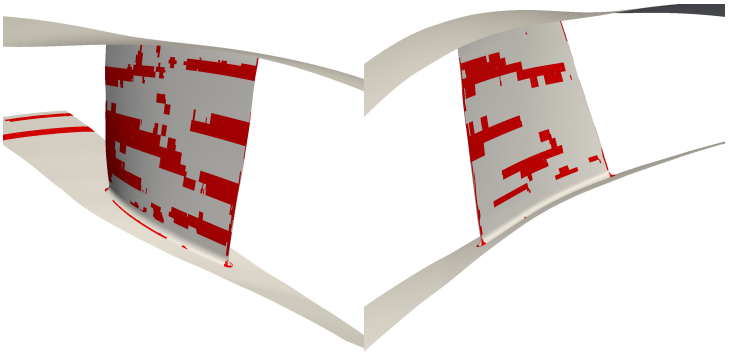
(C) Refined with surface reconstruction.



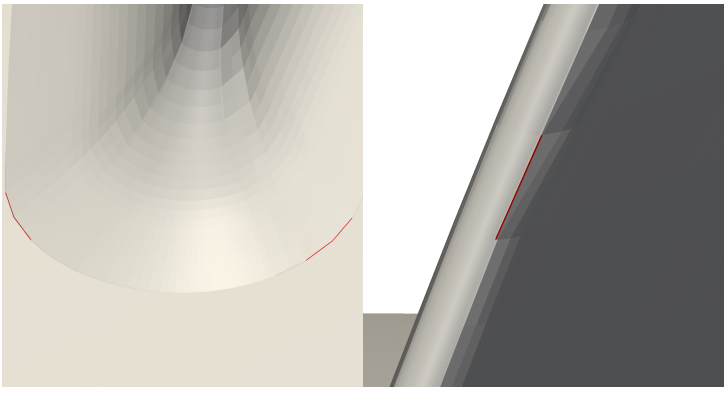
(D) Linear spring mesh expansion about the refined surface with reconstruction.

FIGURE 6.22: Surface reconstruction for refinement at the *LE* of *NASA Rotor* 37.

An example of the overall process is shown in figure 6.22, where the grid around the leading edge of *NASA Rotor* 37 is highlighted. The original mesh refinement strategy (figure 6.22b), places the new node at the edge mid-point (figure 6.22a), thus providing no approximation of the blade's original curvature. By applying the procedure outlined, the point is positioned off the original edge and intersects the boundary layer mesh (figure 6.22c), and thus the resulting grid is invalid. Therefore the linear spring-stiffness approach must be employed. This will perturb the nodes by an amount approximately equal to the relocation vector of the surface node, thus expanding the grid lines (figure 6.22d).



(A) Negative primal volumes after surface reconstruction.



(B) Negative primal volumes after linear spring mesh expansion: *LE* fillet (*LHS*) and *LE* mid-span (*RHS*).

FIGURE 6.23: Negative primal volumes appearing after surface reconstruction.

The linear spring-stiffness effect on the overall grid is significant: in the mesh used in figure 6.22 there were 10862 negative primal volumes after the curvature approximation (displayed in figure 6.23a). After the application of the mesh smoothing this number was reduced to 16. Unfortunately, this still causes the grid to be invalid. Therefore further work is required to solve the issue and allow the software to improve the refinement by replicating the geometry being analysed. In particular, given the location of these problematic volumes (i.e. *LE*/*TE*, fillet/hub interface where the inter-block-boundaries

are), it is possible extra conditions ought to be added in the code. It is believed that area weighting of each face's norm contribution to the nodal wall-perpendicular vector ought to be taken into account. In fact, the possibility of having larger surface elements next to smaller ones was not considered. This issue was particularly obvious between the rotor LE/TE and the suction/pressure side (i.e. where the strong curvature ends and meets the blade side). Additionally, at the hub/fillet interface, this problem was exasperated due to the intersection of the fine near-wall mesh of the hub and that of the fillet. In the particular case of the mesh analysed[12], these had a significantly different spacing, indicating that weighting the surface element's norms by their area could help remove the issue. Once this problem will have been solved, it will be possible to finalise the code by including the curvature reconstruction along the geometric discontinuities such as that shown in figure 6.19.

6.5.3 Mesh Movement

The grid movement strategy is based on the work of [29]. This is a relatively simple explicit spring-stiffness analogy approach as displayed in figure 6.24.

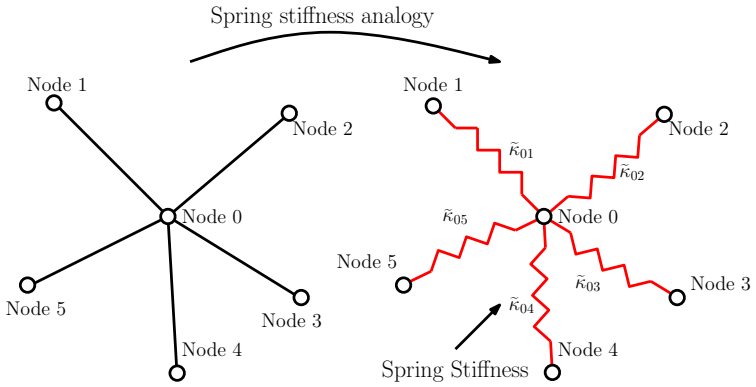


FIGURE 6.24: Grid spring stiffness analogy.

As described in section 6.2.3, the *anisotropic* mesh movement will compute a *Riemannian* edge length d^*_{ij} given the original one and the positive semi-definite *Hessian* matrix along the edge between nodes i and j. The value of d^*_{ij}, i.e. the edge-based error, is thus employed to determine a spring stiffness $\widetilde{\kappa}_{ij}$:

$$\widetilde{\kappa}_{ij} = \frac{d^*_{ij}}{\| \overrightarrow{x}_i - \overrightarrow{x}_j \|} \tag{6.37}$$

[12]This grid was not employed for any of the results presented, it was only used to develop the code.

Where the denominator is original edge's *Euclidian* length. By considering the overall grid as a system of springs, it is possible to formulate the potential energy ($\mathscr{P}_i$) acting on each node as:

$$\mathscr{P}_i = \frac{1}{2}\sum_{j=1}^{N^\circ\mathbb{E}}(\overrightarrow{x}_i - \overrightarrow{x}_j)^2\widetilde{\kappa}_{ij} \tag{6.38}$$

Where the summation is carried out over all the edges connected to node i. It is now possible to minimise this quantity:

$$min\mathscr{P}_i = \frac{\partial\mathscr{P}_i}{\partial\overrightarrow{x}_i} = \sum_{j=1}^{N^\circ\mathbb{E}}(\overrightarrow{x}_i - \overrightarrow{x}_j)\widetilde{\kappa}_{ij} = 0 \tag{6.39}$$

Therefore:

$$\overrightarrow{x}_i = \frac{\sum_{j=1}^{N^\circ\mathbb{E}}(\overrightarrow{x}_j\widetilde{\kappa}_{ij})}{\sum_{j=1}^{N^\circ\mathbb{E}}\widetilde{\kappa}_{ij}} \tag{6.40}$$

However, each grid node has to be smoothly relocated towards the optimum point through a series of iterations to be able to allow other nodes to move without generating invalid elements. Therefore, instead of determining the new location, the difference between the current one and that of the next step may be evaluated:

$$\Delta\overrightarrow{x}_i = \overrightarrow{x}_i^{\text{new}} - \overrightarrow{x}_i^{\text{old}} = \frac{\sum_{j=1}^{N^\circ\mathbb{E}}(\overrightarrow{x}_j^{\text{old}} - \overrightarrow{x}_i^{\text{old}})\widetilde{\kappa}_{ij}}{\sum_{j=1}^{N^\circ\mathbb{E}}\widetilde{\kappa}_{ij}} \tag{6.41}$$

Finally, the updated coordinate is evaluated as:

$$\overrightarrow{x}_i^{\text{new}} = \overrightarrow{x}_i^{\text{old}} + \omega\Delta\overrightarrow{x}_i \tag{6.42}$$

Where a relaxation factor $\omega = [0, 1]$ is included to have a smooth update to each grid coordinate.

An issue that appears with this algorithm is the node order biasing. In fact, the reader can imagine that when the code loops over all nodal indices in the same sequence, the first will have more freedom of movement *w.r.t.* the last. This is even more real if constraints, such as cell volume and skewness, are introduced. Therefore, in this work, the order was changed in groups of four, as shown in the following image:

Iteration	Node ID							
$i \cdot 4$	n1	n2	.....	$\frac{N}{2}-1$	$\frac{N}{2}$	$\frac{N}{2}+1$	.....	N
$(i \cdot 4)+1$	N	.....	$\frac{N}{2}+1$	$\frac{N}{2}$	$\frac{N}{2}-1$	.....	n2	n1
$(i \cdot 4)+2$	$\frac{N}{2}$	$\frac{N}{2}-1$	.....	n2	n1	N	.....	$\frac{N}{2}+1$
$(i \cdot 4)+3$	$\frac{N}{2}+1$	.....	N	n1	n2	.....	$\frac{N}{2}-1$	$\frac{N}{2}$

FIGURE 6.25: Node order change for mesh movement.

Where N is the total number of nodes present in the mesh and i is the iteration.
An example of the resulting node clustering and improved flow feature resolution is shown in figure 6.26. While the initial mesh overestimates the flow feature thickness, the repeated application of this technique overcomes the issue without any node addition.

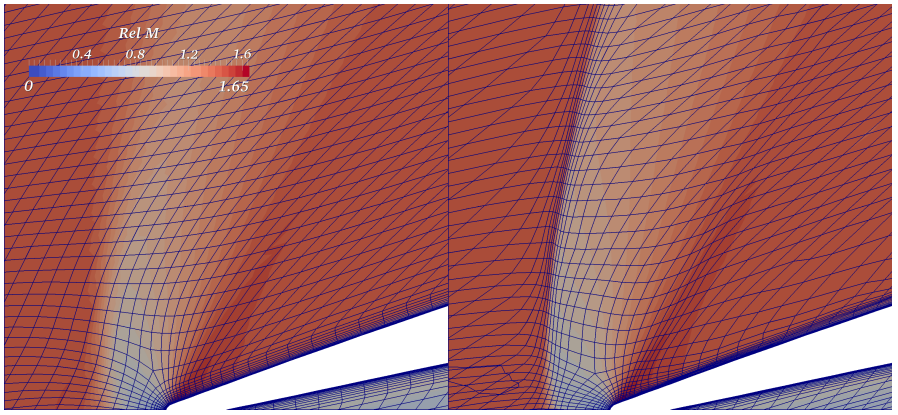
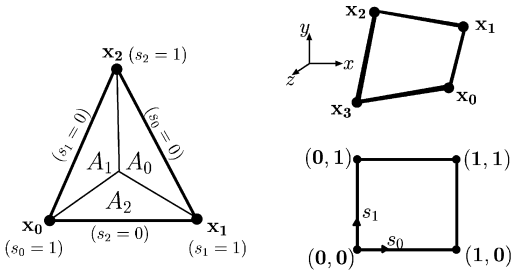


FIGURE 6.26: Example of node relocation around a shock at *LE* for *NASA Rotor* 37 at mid-span.

6.5.4 Surface Grid Relocation

The technique employed to be able to move surface mesh nodes, yet maintaining the geometry definition, is that described by [190]. This adopts a *bilinear* representation of the boundary grid, that may be composed of triangles or quadrilaterals as reproduced in figure 6.27. In this case the nodes were moved according to a geometry optimisation algorithm and then projected onto the closest surface element.



$$A = A_0 + A_1 + A_2$$
$$s_i = A_i/A, \quad 0 \leq s_i \leq 1$$
$$s_0 + s_1 + s_2 = 1$$
$$\mathbf{x} = (x, y, z) = s_0\mathbf{x_0} + s_1\mathbf{x_1} + s_2\mathbf{x_2}$$

$$0 \leq s_0, s_1 \leq 1$$
$$\mathbf{x} = (x, y, z) = (1 - s_0)(1 - s_1)\mathbf{x_0} + s_0(1 - s_1)\mathbf{x_1} + s_0 s_1 \mathbf{x} \; (1 - s_0)s_1\mathbf{x_3}$$

FIGURE 6.27: Bilinear triangular and quadrilateral representation from [190][13].

In this work, however, the node movement had been determined through the spring stiffness approach, driven by *Hessian* eigen-decomposition. Following the unconstrained point relocation, a gradient search to minimise the *Euclidean* distance to the closest starting mesh surface element was employed. This means that it would start searching from the node's original location on the grid and consequently check the surrounding quadrilateral/triangular areas. For points on ridges their movement would either be forced to 0, if they were on corner points, or along a line. Again, as in section 6.5.2, the location of these peculiarities was determined by calculating their norm vectors and checking that the scalar product of adjacent surface normals was smaller than $\cos(20°)$. Examples of successful nodal relocation are shown in figure 6.28, where the red and black edges represent the moved and starting grid, respectively.

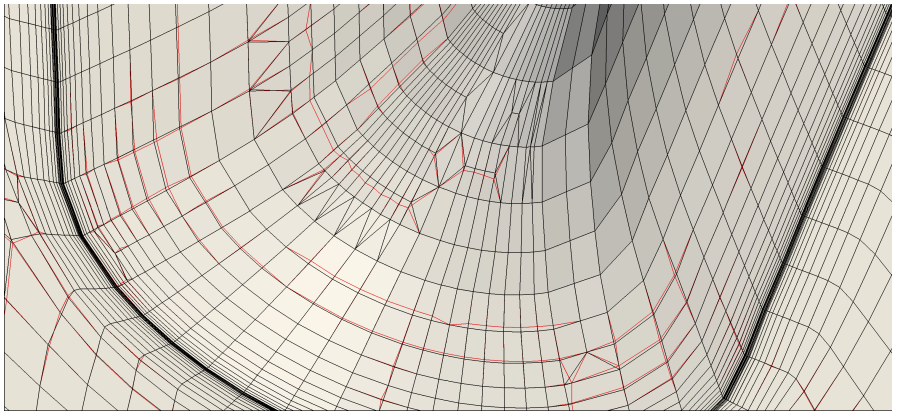


FIGURE 6.28: Surface mesh movement with bilinear reconstruction at the hub/blade interface of *NASA Rotor* 37. Red and black edges represent the moved and starting grid, respectively.

[13]Reproduced with kind permission of *Elsevier*.

It is unfortunate that the surface mesh movement was developed prior a correction to the relocation of periodic boundary nodes, coded by previous developers. In fact, in the original code, the periodic boundary *Hessian* matrix had not been determined correctly (i.e. the rotation described in section 6.2.2 had not been implemented). For this reason, the movement along the periodic sides was constrained to the tangential direction. When the surface relocation was being implemented, the wall nodes shared with a periodic boundary were also enforced to move in the tangential direction. Once the correction to the *Hessian* matrix calculation at these boundaries had been carried out, thus allowing to remove the constrained tangential movement along the periodic sides, inconsistencies appeared in the resulting grid.

Chapter 7

Combination of Feature and Adjoint Grid Adaptation

7.1 Introduction

Within the *CFD* community, it is generally known that adjoint mesh adaptation procedures tend to be more accurate and robust than their feature-based counterparts (see for e.g. [10]). However, they are costly to execute. This because running the dual solver doubles the overall simulation time. However, the time factor can be even more relevant, as for automatically differentiated, discrete, adjoint solvers, the flow *Jacobian* matrix at each computational time step may need to be computed, to avoid storing very large amounts of data, thus lengthening run-time.
The adjoint error estimation approach, has shown in the past that counter-intuitive and unforeseen flow regions may require adaptation to improve the functional of interest. On the other hand, the feature related methods will always tend to ameliorate the resolution of flow complexities, such as shocks, wakes and boundary layers. Careless use of these techniques will cause over-refinement in these parts of the flow, thus resulting in finer grids. Nevertheless, often there is an overlap between the high error regions that they highlight. In fact, good resolution of the main flow complexities can be helpful in determining the correct functional sensitivity. To prove the validity of this assumption, some examples of feature and adjoint error maps are shown in figures 7.2 and 7.4a, where the sensors are shown at mid-span of *NASA Rotor* 37 at design point, for the $0.67M$ grid shown in figure 7.3a[1]. In the first case, the absolute value of the relative *Mach* number *Hessian* diagonal components, highlight the shock, its propagation and interaction with the boundary layer along with the wake. It is clear that, the grid has a relative poor

[1]These should be compared with the 0.67M grid flow solution displayed in figure 7.1.

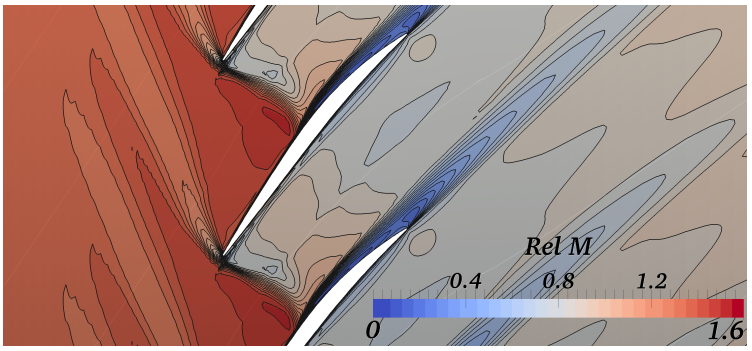
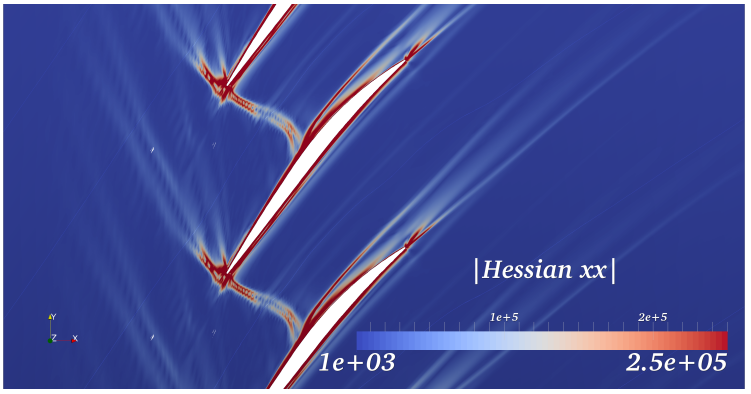
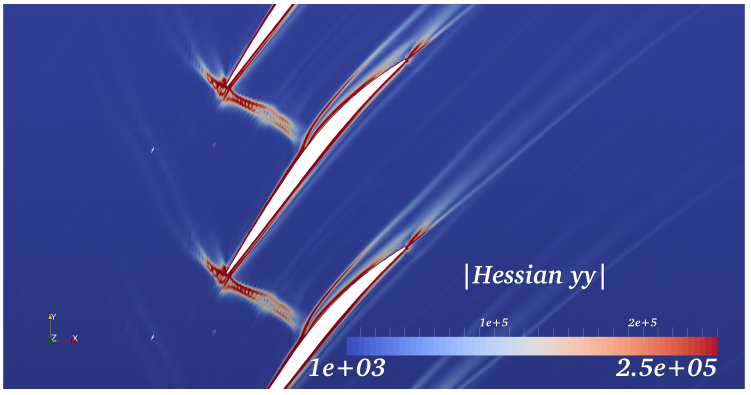


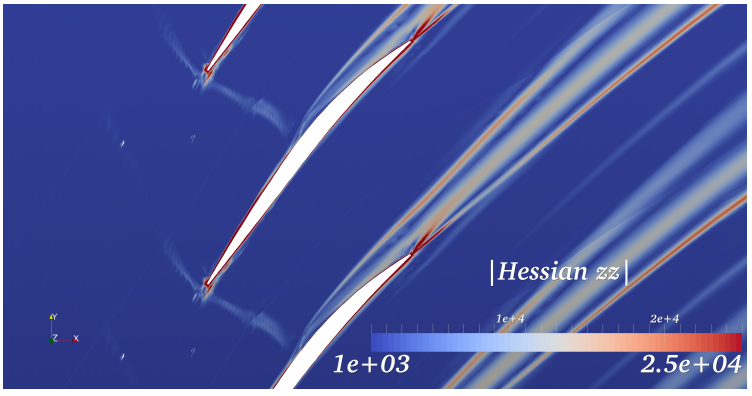
FIGURE 7.1: *NASA Rotor* 37 relative *Mach* number at mid-span.



(A) xx-component.



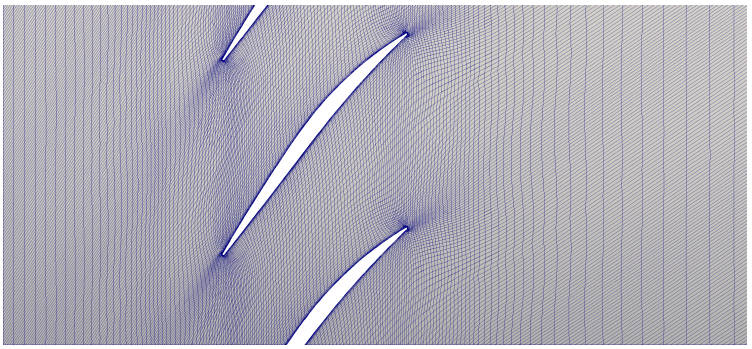
(B) yy-component.



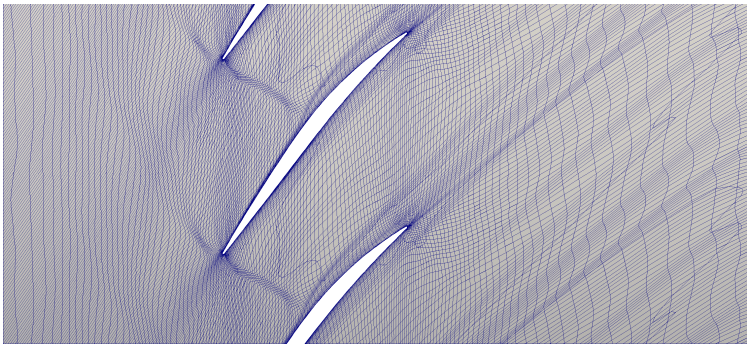
(C) zz-component.

FIGURE 7.2: Diagonal terms of the positive semi-definite *Hessian* of relative *Mach* number at 50% span of *NASA Rotor* 37.

resolution of these flow complexities: in particular the passage shock between the *LE* of a blade and the suction-side of the adjacent one is, qualitatively, very thick.



(A) Starting grid.

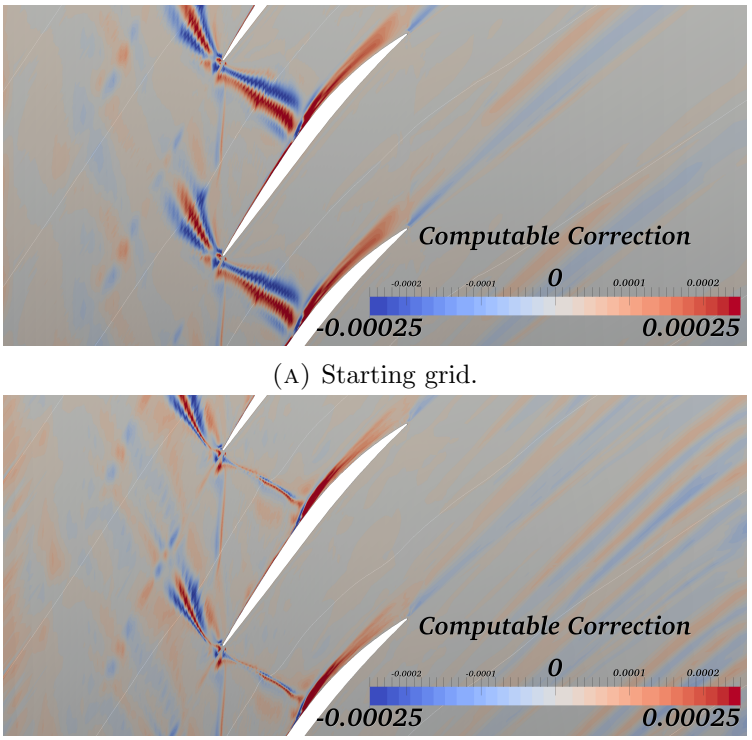


(B) Adapted grid.

FIGURE 7.3: Comparison of the starting *NASA Rotor* 37 0.67M node mesh at mid-span along with the adapted one achieved employing 15 *relative Mach* number mesh movement steps.

On the other hand, the *computable error correction*[2] in figure 7.4a, highlights similar regions of the flow. As expected, due to adjoint nature, upstream parts of the domain will also be accentuated. Nevertheless, the poor resolution of the flow complexities noted in the feature-based approach is also visible in this case. Therefore being able to appropriately capture these should aid limit the regions where adjoint mesh adaptation is required. To prove this point, repeated relative *Mach* number *Hessian*-based mesh movement were applied till the performance values of interest, such as mass averaged *adiabatic efficiency* (η) evaluated between inlet and outlet, stopped changing. The grid employed in figure 7.3a was therefore adapted 15 times, the resulting mesh being shown in figure 7.3b. Once the final grid's adjoint of adiabatic efficiency had been run, the *computable error correction* was recalculated, and is displayed at mid-span in figure 7.4b. It is clear that in comparing this result with that of figure 7.4a, the sensor map is shown to be much more accurate in indicating which regions require further adaptation. In

[2]A subtlety the reader will have noticed that the adjoint *computable correction* maps show, is the intermittent nature of the error. This is due to the flux scheme reconstruction, i.e. *JST* with *matrix dissipation*. This behaviour represents the residuals slight over- and under-shoots around flow complexities, as the flow solver is not *TVD*.



(A) Starting grid.

(B) Adapted grid.

FIGURE 7.4: *Computable correction* at mid-span of *NASA Rotor* 37 for mass averaged adiabatic efficiency evaluated between inlet and outlet.

particular, the shock propagating between the *LE* and suction-side of two neighbouring blades is significantly thinner. This is the result of edge alignment and node clustering towards the main features of interest: they are now significantly better resolved, thus allowing a much more accurate computation of the adjoint-based error.

While it is possible to conclude that feature-based methods may be helpful in better resolving flow complexities that may sharpen the adjoint error map, they do not, necessarily, improve the functional *w.r.t.* which the dual solver has been run. In fact, they tend to improve the flow accuracy globally, rather than being tailored for a single quantity of interest. On the other hand, adjoint techniques weight the residual by $\frac{\partial f}{\partial \mathbf{R}}$. This means that only parts of the flow where the functional is most sensitive and *NS* imbalances are high, will be flagged as problematic. Therefore it is easily understandable that this technique will be far more efficient at targeting the error in a single quantity. Nevertheless, as previously mentioned, employment of adjoint error estimation is expensive due to the extended run-time they require. Additionally, every different functional will need its own adaptation loop, thus if n performance quantities are to be accurately determined, n adaptation runs will be needed. This represents an issue, unless large amounts of *CPU*s and storage are available. On the other hand, feature-based methods are global, therefore only one adaptation procedure is required. The two approaches

may thus be employed in a complimentary fashion: to start with, flow-based grid modification strategies may be applied to improve the grid alignment and feature resolution so that the adjoint error may be able to better estimate the regions requiring modifications. Moreover, to start with a single adaptation process may be run for all values of interest, rather than one for each functional.

7.2 Combined Feature and Adjoint-Based Adaptation

As stated in the previous section, the flow-related grid modification strategy is employed to start with. This consisted in a relative *Mach* number *Hessian*-based node relocation, that is fully *anisotropic*. Starting with r-adaptation, will allow to equidistribute the initial grid error by aligning its edges at no extra cost. Moreover, the author noticed that mesh movement strategies are generally much more robust to employ. In fact, using them will not change the connectivity and thus the cell types, but only their shape and size. This is generally dictated by the flow solver's solution, and if carried out carefully can produce a reliable technique. On the other hand, node enrichment, is much more complex to handle in a repeated mesh adaptation process. In fact, any error estimation will only indicate whether an edge/face/cell ought to be split, and does not carry any information on how any resulting new connectivity in the grid may affect the flow solver. This is best explained with an example. It is often the case that an initial mesh has the shock location incorrectly resolved (albeit by a small amount). Careless refinement in this region will target exactly where the discontinuity has been placed, as no information on whether this is correct or not is available. In the case of conformal solvers, the refinement patch placed around the shock ought to be closed to avoid *hanging nodes*. The next time the flow solver is run, given the increased spatial resolution provided by the refinement, the shock may move onto the region closing the finer patch or in an unrefined zone, causing oscillatory convergence. If further refinement is applied, very distorted elements will appear in the interface patch between the fine and coarse mesh, or even worse, gaps between two finer grid portions. Hence, unless sophisticated refinement algorithms are applied, it is necessary to start all-over again, thus conflicting with the requirement for the process to be automated. On the other hand, mesh movement strategies are better suited to cope with these kind of issues. For this reason, no feature based mesh refinement was applied, it was instead employed when switching to adjoint-based adaptation.

The overall process consisted in repeating the feature-based mesh movement to improve the grid alignment, till the values of interest had converged. Consequently, adjoint-based node enrichment would be carried out. In particular, primal cell refinement based on the *non-computable correction* was applied, where the volume would be marked if the cell's

average nodal error was greater than the selected threshold. The reason behind choosing this sensor, relates to the standard adjoint adaptation process developed by [64], as it has shown that the *non-computable correction* term is better suited for the adaptation *w.r.t.* the *computable correction*. The error formulation consisted of all terms as in equation 6.26. On the other hand, the *computable correction* was adopted to determine the enhanced fine grid functional.

In this work, it was decided to refine a mesh based on the amount of nodes added, and not according to a user-defined threshold. In an industrial setting, many simulations may have to be run and thus only a limited amount of refinement would be possible to reduce the burden in terms of memory consumption and run-time. The reason behind choosing $0.3 \sim 0.4$M as the range of possible added nodes is due to the fact that it represents approximately half of the starting mesh count (i.e. $670{\cdot}056$), and would limit the size of the embedded mesh needed for adjoint-based adaptation steps. A final note concerning the refinement at the wall region is due. As described in section 6.5.1, the adaptation software has the capability of extending the edge splitting in the normal direction to the surface, by a number of layers set by the user. After a wide range of trials, it was concluded that setting the propagation to 10 wall perpendicular layers allowed the best robustness without causing too much over-refinement. Finally, it was decided to apply two adjoint-based refinement steps after the sequence of feature-related mesh movement iterations. This is because no surface reconstruction was employed, and therefore multiple refinement steps may introduce increasingly growing errors due to geometry inaccuracies (this is particularly true when the embedded grid is generated to estimate the adjoint error). Moreover, when employing mesh movement, cells may become significantly distorted. Consequent refinement may then produce negative volumes, particularly in the boundary layer mesh, at the blade's *LE* and *TE*.

Figure 7.5 shows a schematic flow chart representing the procedure taken. This indicates that the adjoint-based process should be carried out till functional or *computable correction* have converged. However, this has been included for completeness and generality, but as previously discussed, in this work, only two refinement steps were allowed.

To the author's knowledge, very few attempts have been made to adapt a 3D jet engine component, including turbulence, utilising adjoint error estimates. Even fewer examples exist of an improvement to the functional of interest using either the *computable correction* ([98]) or the *non-computable* one ([99]). Moreover, adjoint error techniques have been combined with *Hessian*-metric based mesh regeneration to achieve *anisotropy* (for e.g. [64]), but they have not been employed in a sequential fashion to move the mesh prior to the dual error use.

In the following sections the procedure just outlined in detail will be applied to the *NASA Rotor* 37 at aerodynamic design point. Following a description of the test-case, the approach validity will be shown for *adiabatic efficiency* (η), *absolute total pressure*

ratio (Pr), both of which were mass averaged and evaluated between inlet and outlet, and finally mass flow ($\dot{m}$), area averaged and integrated over the exit plane. All values will be compared against a 47M hand-generated mesh.

Additionally, to carry out a more complete analysis, a fully feature-based mesh adaptation process will be used as a comparison. This will consist in repeatedly moving mesh, just as in the combine feature and adjoint process, followed by the *JST*-edgewise switch based refinement. For an in-depth description of this approach and the results, the reader is referred to appendix C.

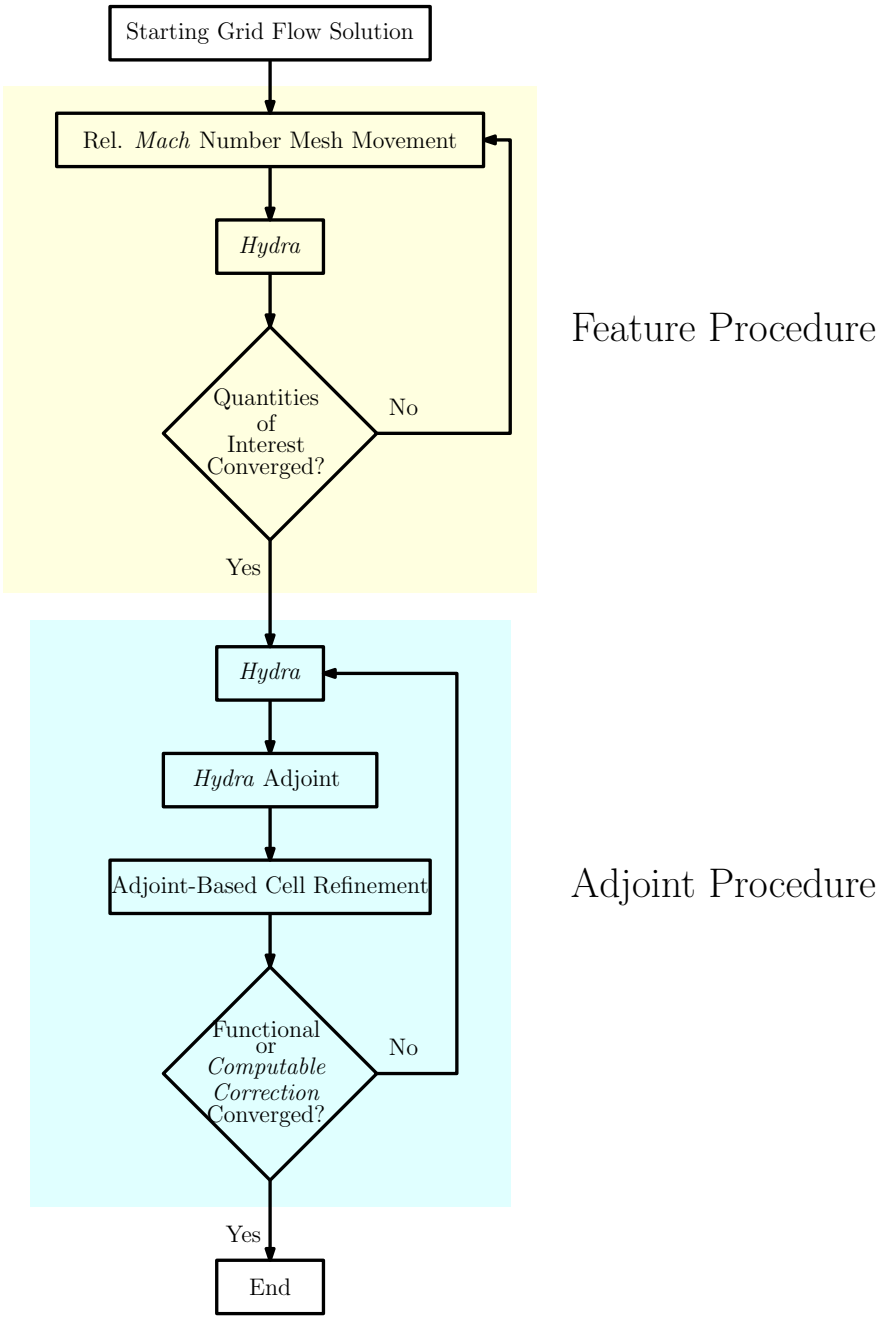


FIGURE 7.5: Combined feature and adjoint adaptation flow chart.

7.3 NASA Rotor 37 Compressor Blade

7.3.1 Physical Aspects

$N°$ Blades	36
Leading-Edge Tip Diameter	$0.5074m$
Leading-Edge Hub Diameter	$0.3576m$
Rotational Speed	$1800.006836\frac{rad}{s}$
Tip Solidity	1.288
Tip-Clearance	$0.356mm$
Tip-Speed	$454.14\frac{m}{s}$
Pressure-Ratio	2.106
Mass-Flow	$20.19\frac{kg}{s}$
Blading	Multiple Circular Arc

TABLE 7.1: *NASA Rotor* 37 characteristics at aerodynamic design point as in [191][3].

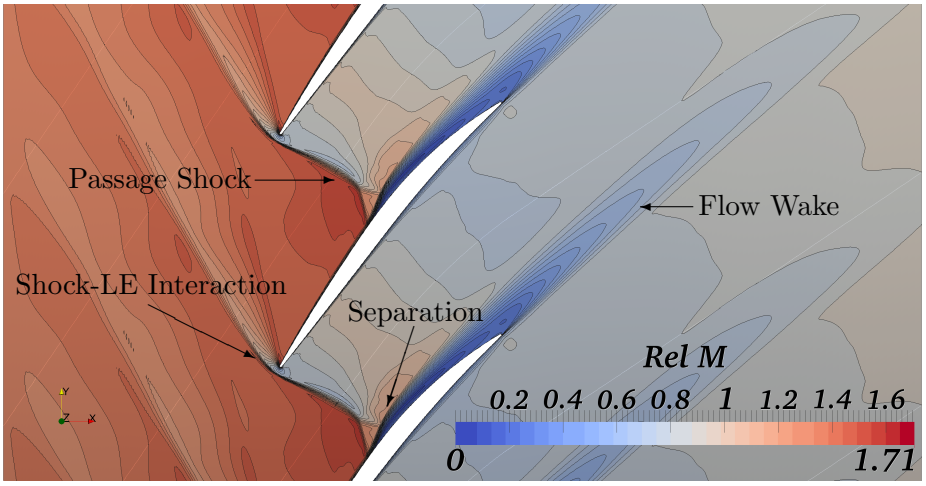
NASA Rotor 37 is an axial transonic compressor blade. It represents the main benchmark for turbomachinery analysis for 3D flow solvers [191], therefore appropriate for testing and validating mesh adaptation techniques. The main characteristics at the aerodynamic design point are given in table 7.1.

In order to achieve an accurate solution to be used as a comparison for coarser grids and determine what kind of flow features needed to be replicated, a $47M$-node mesh was run. Span-wise cuts of the results at 50% and 99% span are displayed in figures 7.6a and 7.6b, respectively. A shock forms just off the blade's *LE* due to its blunt geometry. This propagates in both directions: counter-rotationally, this feature spreads further whilst being smoothed out, till it hits the (computational) inlet plane. In the direction of rotation it impacts the suction-side of the adjacent blade. At this point, the adverse pressure gradient causes flow separation that forms a wake extending beyond the rotor *TE*. The lambda shape of the shock, where it interacts with the boundary layer, had not been predicted by the experimental results displayed by [191, 192]. As stated by [191], the laser anemometer did not measure the flow behaviour near the suction-side of the blade, and was therefore unable to capture the separation as well as the shock-root structure. Furthermore, in the author's experience, the grids required to simulate this behaviour need to be relatively fine ($>$ 3M nodes) if they are not aligned with the shock.

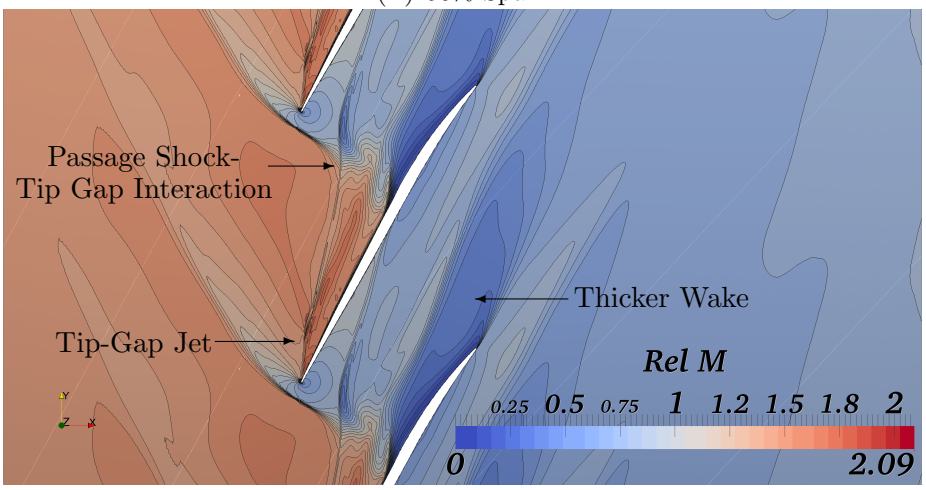
At 99%-span, the flow features get more complex: flow from the pressure side moves across the tip-gap and forms a jet on the suction-side of the blade. This then impacts the shock forming a vortex. The increased boundary-layer thickness at this height can be attributed to the tip-gap flow adding momentum in the circumferential direction.

As hinted by [191], a corner separation forms in the hub vicinity, on the *TE*-suction-side of the blade. Again, unfortunately, the experimental data was not evaluated this close

[3]Reproduced with kind permission of *Springer*.



(A) 50%-Span.



(B) 99%-Span.

FIGURE 7.6: 47M mesh span-wise cuts displaying the relative *Mach* number variation at aerodynamic design point.

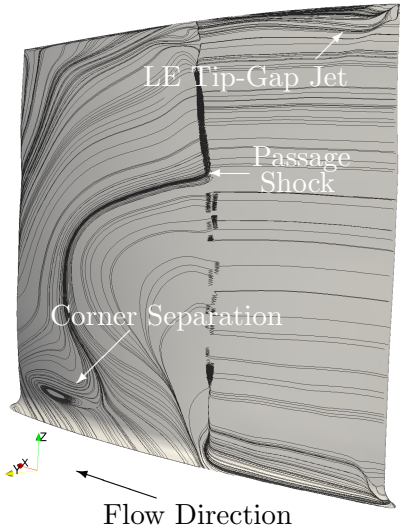


FIGURE 7.7: Suction-side surface streamlines.

to the rotor, and the many simulations reported by [191] were unable to predict this behaviour. On the other hand, *Hydra* is able to capture this phenomenon, as shown in figure 7.7.

In general, in the tests described by [191], the main difficulty of this compressor is presented by the boundary layer capturing and the consequent wake forming, as this is related to the turbulence model.

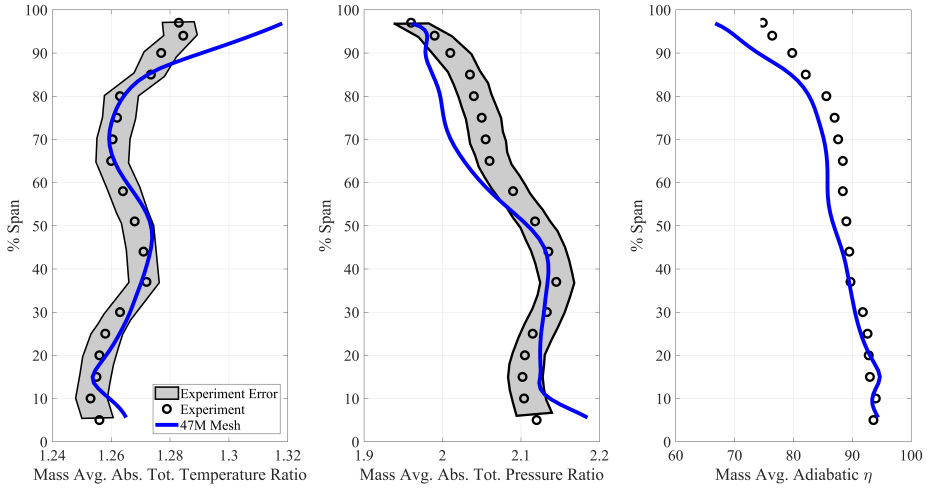


FIGURE 7.8: Comparison between experiment and 47M mesh exit plane radial plot profiles for *NASA Rotor* 37 (experimental data taken from [193]).

Given the availability of experimental data in the form of outlet (*w.r.t.* inlet) radial profiles, a comparison between these results and those of the 47M grid is carried out in figure 7.8. The experimental values were achieved measuring the rotor performance at 98% mass flow *w.r.t.* choke [191–194]. Therefore, in the *CFD* simulations, the first step was to determine the mass flow at choke and then modify (by trial and error) the exit capacity to achieve 98% of the estimated value. It should be noted that the 47M grid calculated choke mass flow matched that of the experiments at 20.93 kg/s. Once the correct operating conditions had been determined, the exit plane (*w.r.t.* inlet) radial profiles of adiabatic efficiency, absolute total pressure and temperature (Tr) ratios could be computed. As stated by [192], mass-flow circumferential averaging of the total pressure and temperature should be used to determine their respective profiles. Adiabatic efficiency is then calculated utilising [194]:

$$\eta = \frac{Pr^{\frac{\gamma-1}{\gamma}} - 1}{Tr - 1} \tag{7.1}$$

As it may be seen from figure 7.8, all three parameters follow the experimental data trend relatively well. Below 10% span, both temperature and pressure ratio show a difference *w.r.t.* the experimental data. This over-prediction is quite common in *CFD* results and

has been attributed to the presence of hub leakage flow upstream of the blade [195]. Another region that is generally poorly resolved by steady-*RANS* simulations is the tip, as the unsteadiness due to tip-leakage-shock interaction is not appropriately predicted [196]. This issue is clearly visible in the total temperature ratio profile. The total pressure ratio is underestimated in the region between 60 to 90% span. This is believed to be caused by the turbulence model under-predicting the velocity profile across the wake. To prove the validity of this assumption, a comparison with the experimental relative *Mach* number at 70% span and 115% chord downstream of the blade is reported in figure 7.9[4].

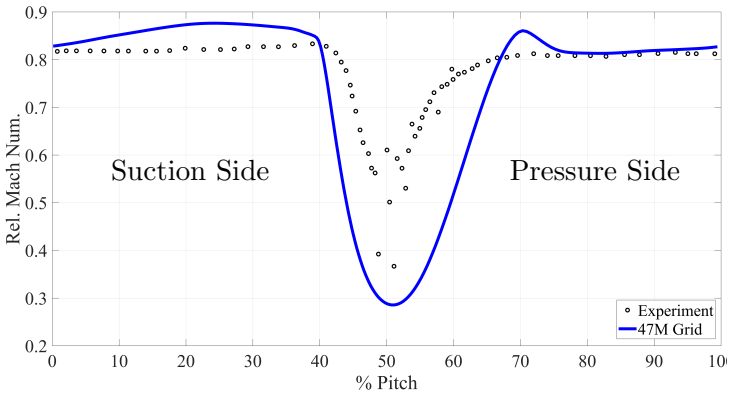


FIGURE 7.9: *NASA Rotor* 37 relative *Mach* number pitch-wise profile at 70% span and 115% chord (experimental data from [197]).

Not only does the simulation produce lower values at the wake core, but its thickness is also over-predicted. This will cause a reduced velocity at the outlet, that will result in a smaller total pressure. An interesting behaviour that may be noticed from figure 7.9 is the presence of an over-shoot on the wake's pressure-side. This is caused by *Hydra* not being a *TVD* solver: if a gradient limiter were used, this feature would disappear.

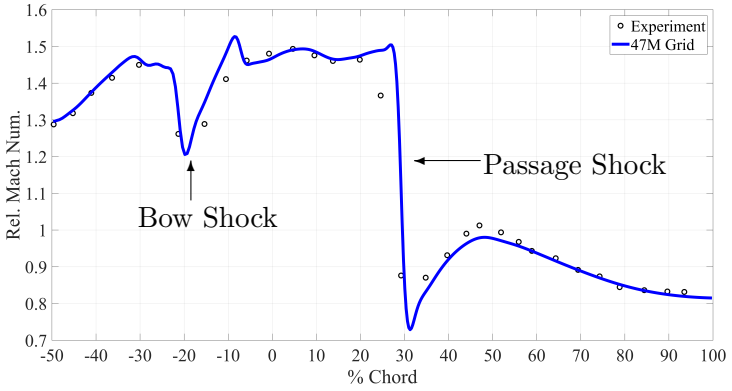


FIGURE 7.10: *NASA Rotor* 37 relative *Mach* number profile between −50% and 100% chord along the mid-pitch line at 70% span (experimental data from [197]).

[4]To appropriately assess the difference, the position of minimum relative *Mach* number of the 47*M* grid has been aligned with that of the experimental data.

Another interesting comparison between *CFD* and experiment is reported in figure 7.10, where the relative *Mach* number profile at 70% span along the mid-pitch line is plotted. The sets of data are clearly very similar, with the 47M grid following the correct trend. Around bow and passage shocks over- and undershoots are again caused by the *Hydra* not being a *TVD* solver.
Following this analysis, it can be concluded that the 47M grid is appropriately capturing the flow features and the data trends within the limits of *CFD* modelling accuracy.

7.3.2 Case Setup

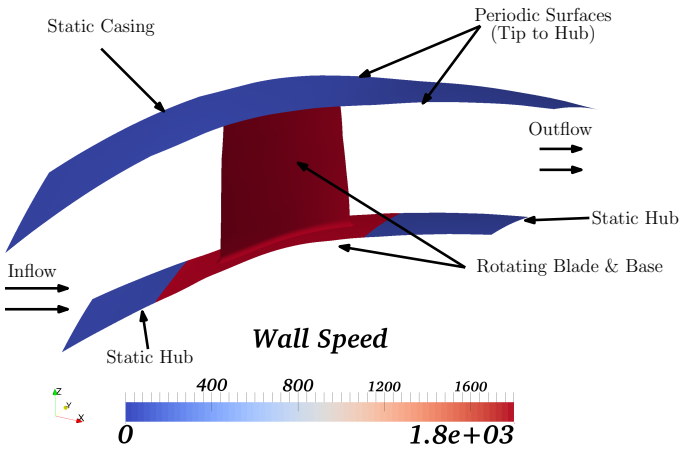


FIGURE 7.11: *NASA Rotor* 37 setup: flow is from left to right.

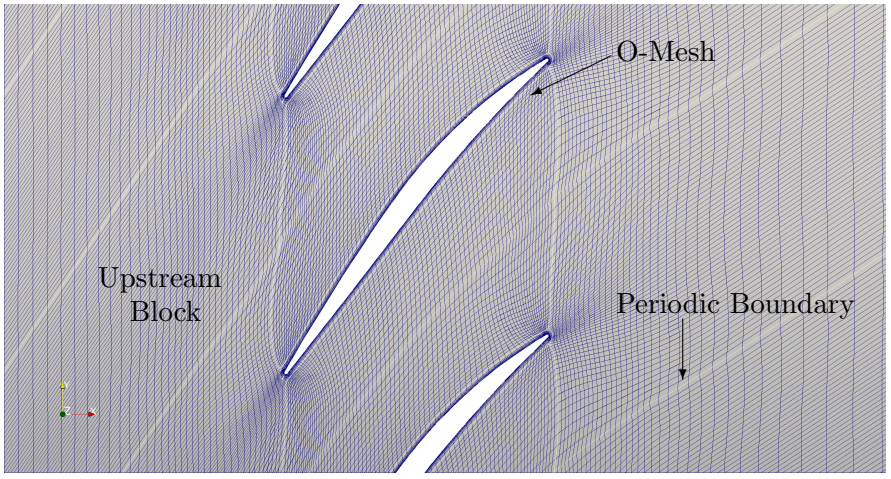
The computational domain is shown in figure 7.11. The surface geometry can be divided into two: rotating parts, i.e. the blade and the hub section immediately beneath it, and static, i.e. the casing and hub up- and downstream sections. While figure 7.11 shows the wall rotational speed in the absolute frame of reference, the flow solver employs the relative frame. For example, this means that the casing will be rotating with a speed of $1800\frac{Rad}{s}$ in the opposite direction *w.r.t.* the real situation. To reduce the computational amount of effort and time required only one blade is simulated by splitting the domain along the passage, at the periodic boundary.
All the starting grids were multi-block structured, generated utilising *PADRAM* (described in section 3.2). The coarsest grid that was adopted in this work had a 0.67M node count and was generated with the desire of having a low number of nodes, while maintaining a good approximation of the blade *LE* and *TE*[5]. Span-wise cuts at 50% and 99%, being shown in 7.12a and 7.12b, respectively. Both have the block boundaries along with the periodic surfaces highlighted. Despite the meshing software being capable of generating good quality grids quickly, the complication lies in aligning the passage

[5]This was particularly important as any consequent refinement step would not be able to appropriately capture any strong curvature.

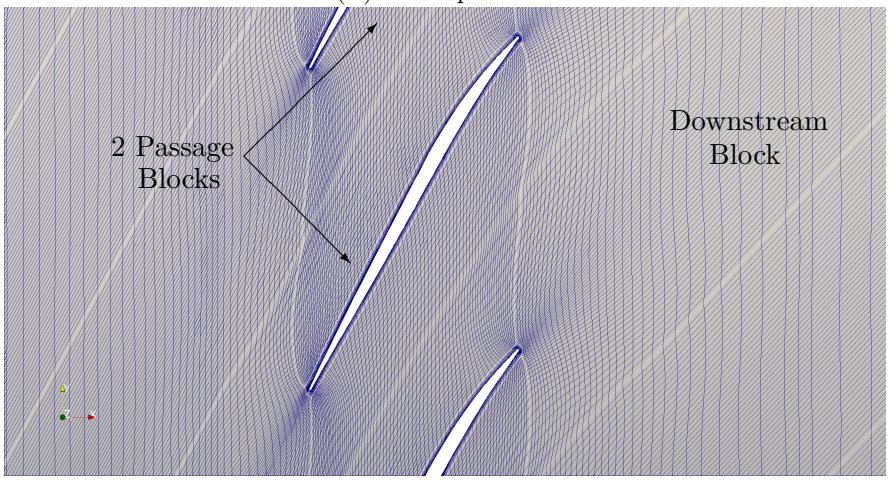
block-boundaries with the surface normal of the rotor. This is due to the presence of the periodic boundary and the changing twist of the blade throughout its height. A crucial point when generating these grids is the tip-gap resolution. In this case a *polar*-mesh was chosen (figure 7.13), as this allows to have a larger number of nodes on the blade *LE* and *TE*, thus capturing the geometry curvature better, while allowing a smooth, coarse grid in the rest of the gap.

The 47M grid was achieved using *PADRAM* by repeatedly (two times) doubling the node count in every direction for each block of the 0.67M case. Additional refinement was applied to be able to achieve a smooth transition between the blocks.

The boundary conditions for all cases were set to subsonic at inflow, while at the outflow subsonic radial equilibrium was imposed. The latter, consists in setting a single static pressure value mid-height of the exit plane and uses the flow's swirl to determine the full outflow profile, as given by the experimental data [198]. In this case, the static pressure value was determined by setting a capacity value of 0.01824 (this corresponds to the aerodynamic design point).



(A) 50%-Span.



(B) 99%-Span.

FIGURE 7.12: 50% & 99%-span cuts of 0.67M grid with block boundaries.

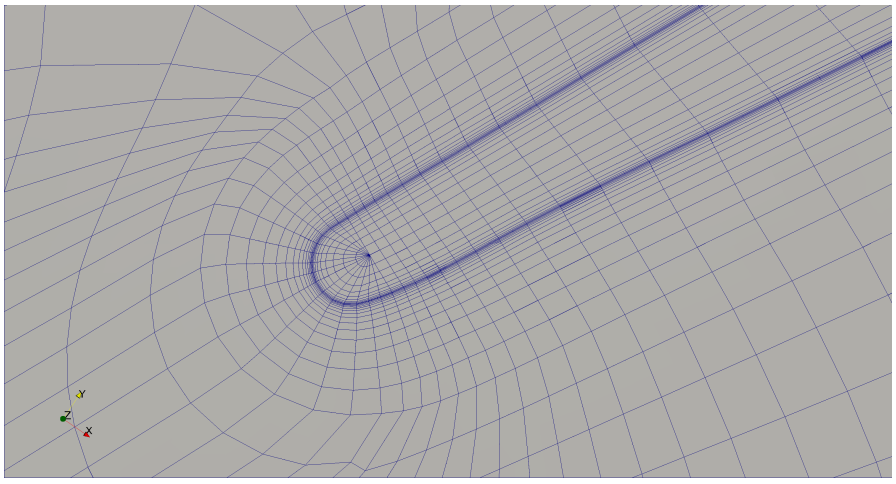


FIGURE 7.13: *NASA Rotor* tip-gap *polar*-mesh.

7.4 Combined Mesh Adaptation

7.4.1 Part I: Feature-Based Adaptation

To start with, the feature-based mesh movement was iteratively run with a fully automated script. The fact that the connectivity was not changed throughout the process meant that the previous step's flow solution could be employed as a starting point for *Hydra*, thus speeding up the overall procedure. The only extra requirement would be the multigrid generation. In fact, not only have nodes changed their positions, but the adaptation software produced a file without any mapping between the multigrid levels. For this reason, it was necessary to regenerate all the grids, which in this case were two[6]. It should be mentioned that a possible future improvement could consist in adapting all the multigrid's levels. This would probably improve the whole adaptation process, as the *anisotropy* would be taken into account throughout the entire multigrid. In fact, as discussed in section 3.3.3, the original edge-collapsing algorithm is *isotropic* and therefore not tailored to coarsen an adapted mesh.
To be able to get an idea of how the parameters of interest changed throughout the adaptation, and check the process robustness, the mesh movement was iteratively applied without any pre-set stopping criteria. In this case it managed to reach 123 iterations, after which it was user-terminated. The performance quantities behaviour is shown in figure 7.14. Apart from the adapted grid's values (blue), the starting (red), target (black) and adaptation process average (green dashed) are plotted for comparison. It is immediately obvious that all three values have closely related behaviour. In fact they set-off by improving (first 3 steps), then steadily decrease to a minimum (between 20^{th} to 30^{th} step) and finally slightly increase and oscillate around the average. Clearly, this

[6]If the fine grid is included, the total is 3.

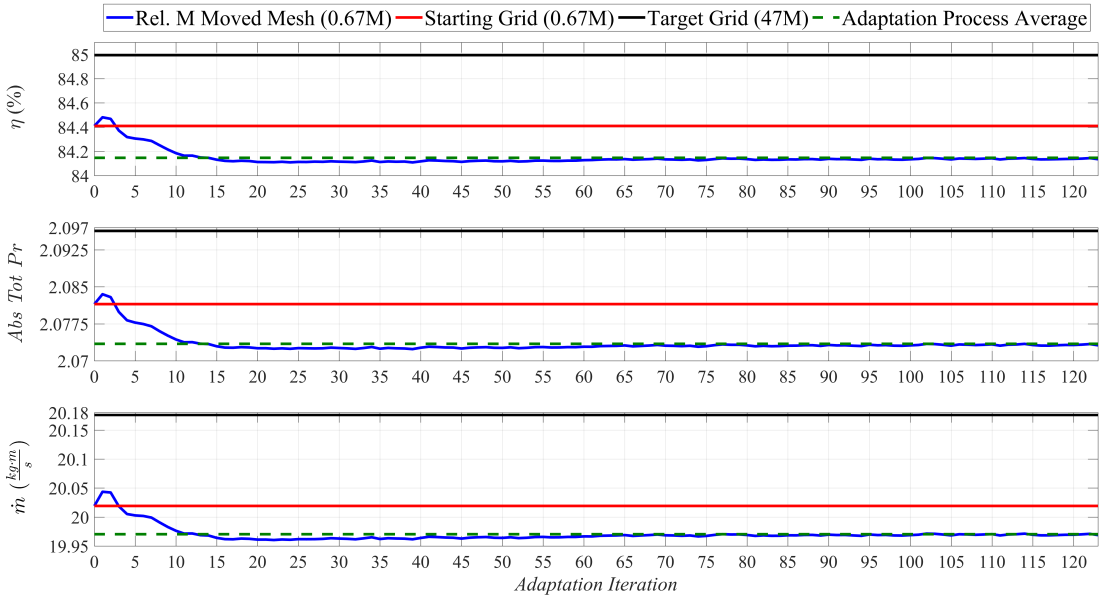


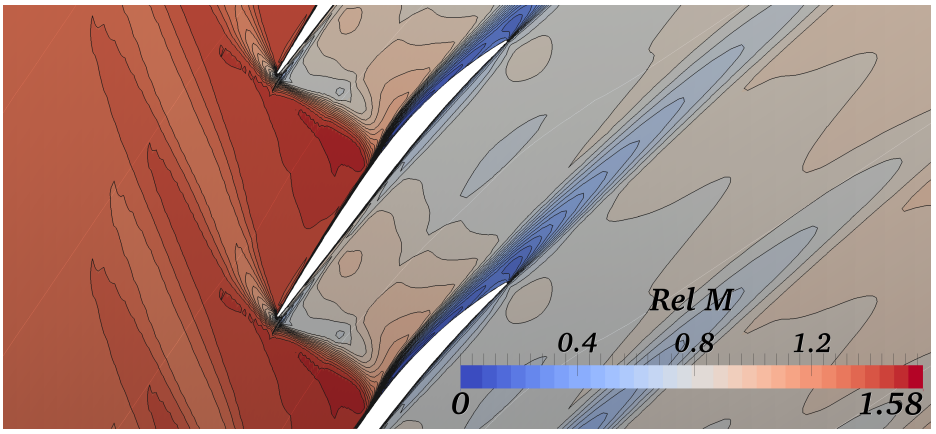
FIGURE 7.14: *NASA Rotor* 37 evolution of η, P_r and $\dot{m}$ throughout the repeated relative *Mach* number mesh movement process.

indicates the strong relationship between the quantities of interest. This reinforces the idea that feature-based mesh adaptation can be helpful in avoiding large amounts of separate adjoint-based grid modifications for each functional of interest. Nevertheless, careful observation shows that after roughly the 15^{th} iteration, variation in the quantities of interest is so small, that it can be considered negligible. Moreover, it would be difficult to claim any possible benefit in choosing one over the other. In this case it was decided to opt for the 15^{th} iteration, as it is the first where all three performance parameters are below the average, with very little difference with the results that follow. To prove this point, the maximum variation in η, Pr and $\dot{m}$ between the value at the 15^{th} step and all the following ones is reported in table 7.2. Clearly, any variation is minuscule, and the only possible gain would be the improved edge alignment and node clustering. However, avoiding the grids produced by later adaptation steps, will reduce the chance of generating negative volumes due to the lower degree of distortion (this is particularly true on the suction-side boundary layer mesh) .

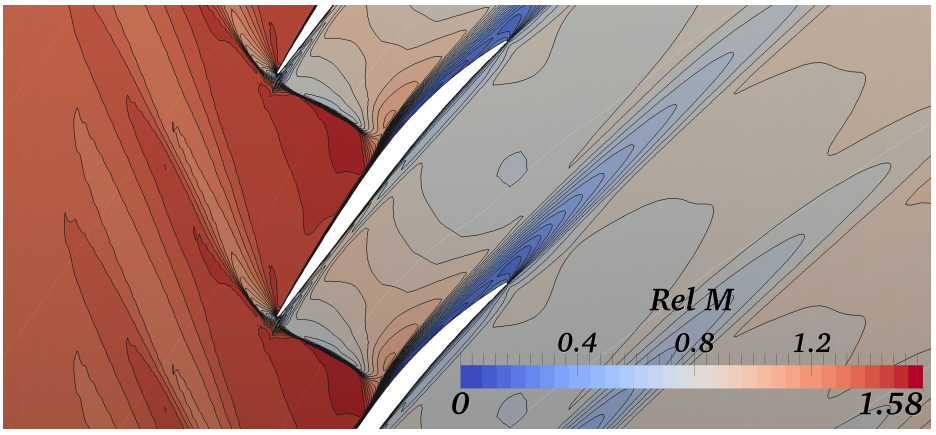
Functional	$max(\Delta)$
η (%)	0.0224
Pr	0.000568
$\dot{m}$ ($\frac{kg \cdot m}{s}$)	0.00368

TABLE 7.2: *NASA Rotor* 37 maximum change in quantities of interest after the 15^{th} feature-based mesh movement step.

A glance at figure 7.14 remarks how the performance quantities are deteriorating as the adaptation is carried out. It can be concluded that the starting grid estimates were closer



(A) Starting Grid.



(B) 15 feature-based mesh movement steps resulting grid performance.

FIGURE 7.15: Comparison of relative *Mach* number at mid-span for the starting and feature-adapted grids of *NASA Rotor* 37.

to that of the target. Therefore one could argue that the overall process is unnecessary or even incorrect. Effectively, this is not the case. In fact, the starting grid is not in *asymptotic range w.r.t.* the target. To this end a qualitative comparison between the 47M mesh (figure 7.6a) and the starting grid (figure 7.15a), is necessary. There are a number of features that are poorly resolved. The main one is the shock: this forms at the blade *LE* and then propagates several times throughout the domain. It is obviously badly captured due to its thickness spanning several cells, but more importantly, its interaction with the boundary layer on the blade suction-side is not correct. Additionally, as discussed earlier, the adjoint reverses flow directionality, indicating high sensitivities in the upstream region. This means that errors generated here will have a greater impact. To this end, it is clear that the upstream propagation of the shock requires much more clustering. Comparing the relative *Mach* number behaviour at mid-span between the starting and adapted grid (figure 7.15), it is obvious that the mesh movement has improved the resolution of the overall shock. In fact, the shock/boundary layer interaction has the correct lambda-shape, the discontinuity thickness across the

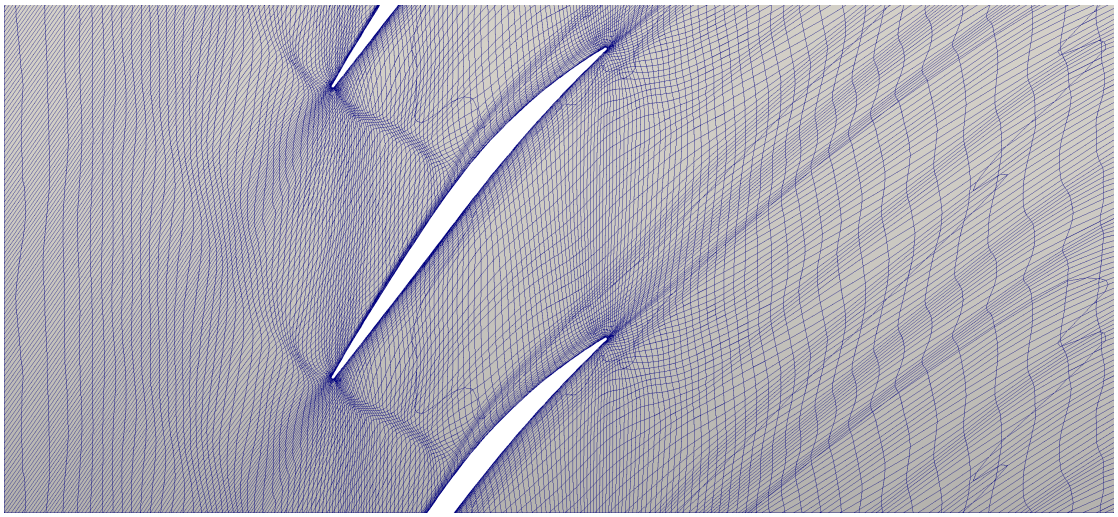
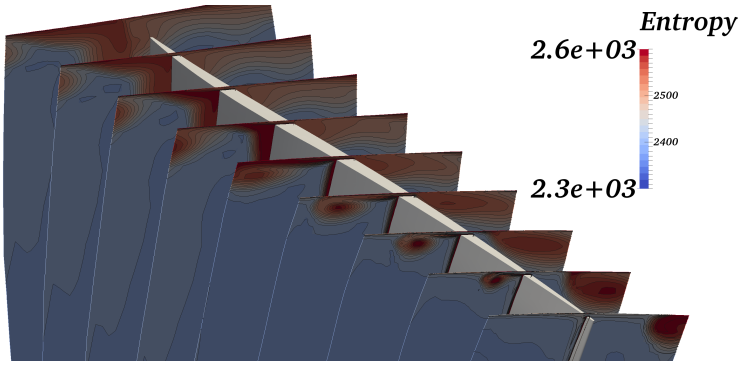


FIGURE 7.16: *NASA Rotor* 37 resulting grid after 15 feature-based mesh movement steps at mid-span.

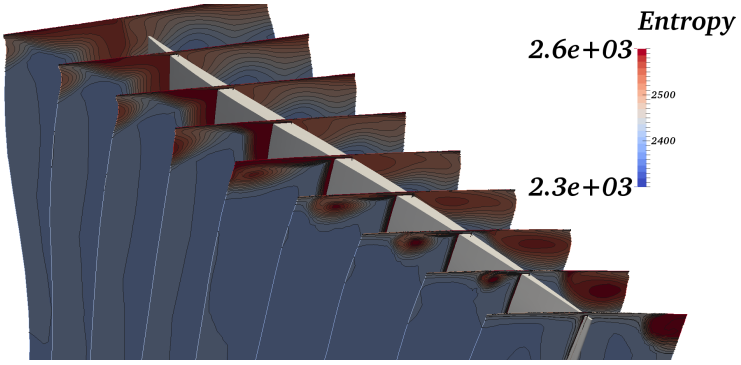
passage is sensibly reduced, and it propagates further in the upstream part of the domain. Interestingly, the wake location and thickness has changed by a small amount. This is probably due the starting grid having a suitable alignment in this region.

As was discussed in section 2.3.1, *anisotropic* adapted grids show the flow features in the mesh lines. This is the case for the resulting grid, displayed in figure 7.16. The shock clustering is visible, particularly along the passage. This spreads out where it interacts with the boundary layer, causing the separation and thus the wake. The interface of the latter feature with the bulk of the flow is also shown by the location of the grid lines. Finally, waviness in the upstream part of the domain proves the nodal relocation towards the shock propagation.

Another interesting characteristic of this test-case is the tip gap flow and its interaction with other flow features. Careful analysis of figure 7.17, shows how the adapted grid's entropy has increased at the vortex core, also larger in size. This appears on the blade suction-side and propagates onto the pressure side. Moreover, on the downstream position of the blade suction-side, where the separation occurs, the boundary layer has thickened, with an entropy increase. Concerning the grid in the tip vicinity, figure 7.18 shows the nodal relocation in this region. Upstream, (figure 7.18a), prior the shock location, the suction-side movement has mainly pushed nodes towards the blade and tip-gap. A greater relocation in the casing direction has occurred on the pressure side due to the tip-gap flow propagation across the periodic boundaries. Further downstream (figure 7.18b), the movement on the suction-side is much more determined. The separation has occurred and the mesh lines have attempted to align with the boundary layer. A similar relocation of the nodes has happened towards the casing along the entire passage mesh.

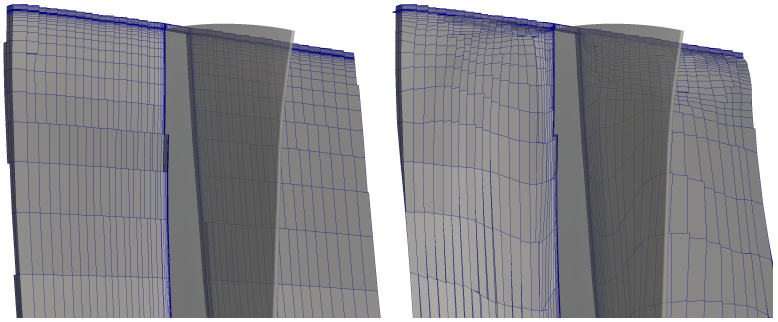


(A) Starting grid.

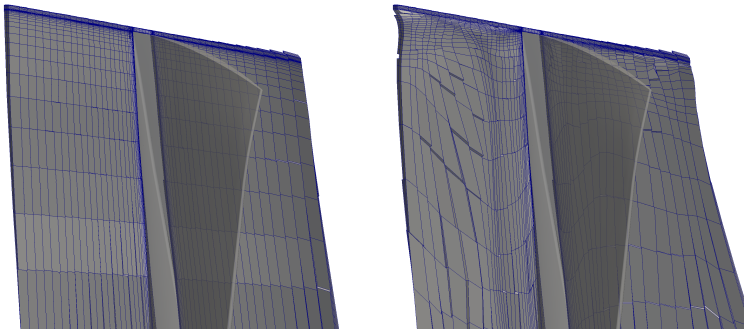


(B) Feature-adapted grid.

FIGURE 7.17: *NASA Rotor* 37 axial cuts for entropy comparison.

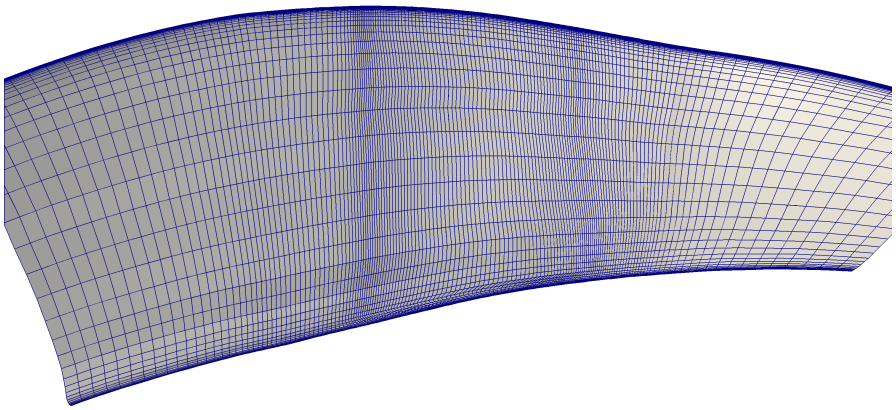


(A) 1^{st} axial cut.

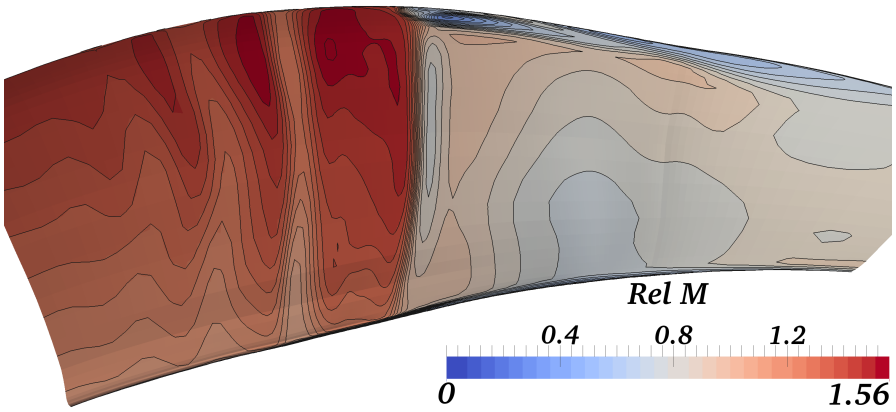


(B) 2^{nd} axial cut.

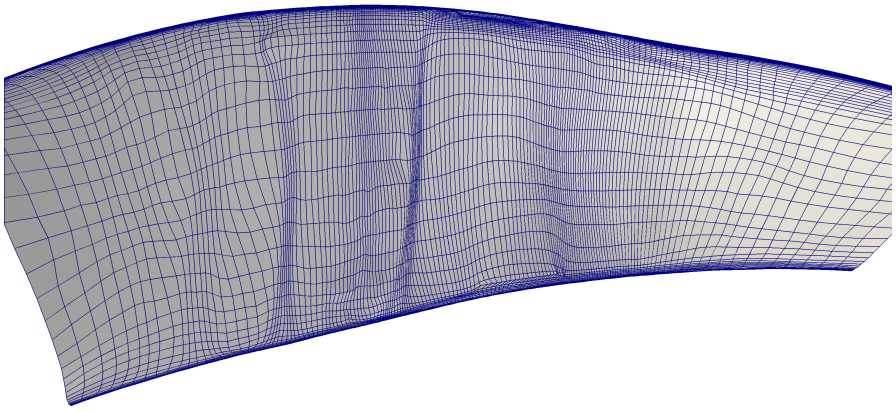
FIGURE 7.18: *NASA Rotor* 37 axial cuts of the starting (*LHS*) and feature-adapted (*RHS*) grids.



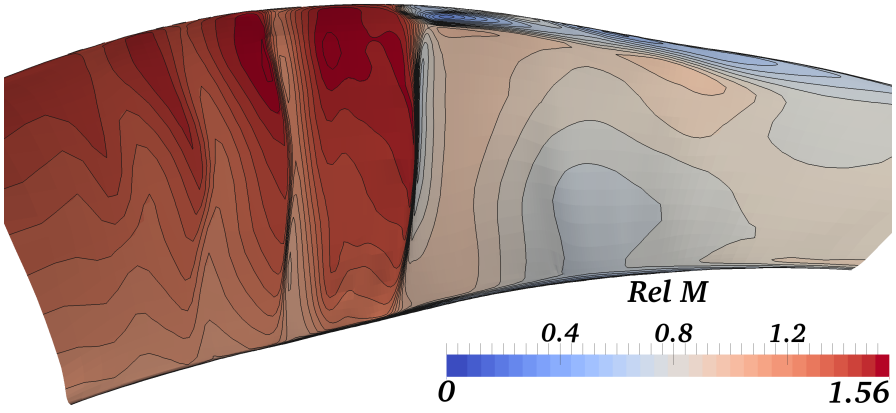
(A) Starting grid.



(B) Starting relative *Mach* number.



(C) Feature-adapted grid.



(D) Feature-adapted relative *Mach* number.

FIGURE 7.19: *NASA Rotor* 37 periodic boundary of the starting and adapted meshes.

A conclusive discussion relative the periodic boundary plane is due at this point. In fact this is a smooth and continuous plane in both initial and moved grids. Comparing the meshes in figures 7.19a and 7.19c, clustering of the mesh lines towards the two strongest shocks cutting the plane is clearly visible. On the other hand, the discontinuity propagation upstream is less aggressively captured by the node clustering. Another region where the mesh density has considerably increased is in the vicinity of the casing, downstream of the passage shock. This is due to the tip-gap propagation. The adapted mesh's resulting flow field is displayed in figure 7.19d. Comparing this to that of the starting grid (figure 7.19b), the shock is stronger, with its resolution having significantly improved. On the other hand, the upstream propagation of this feature is not as crisp, nevertheless bunching up of the contour lines is still visible. Again, despite the significant nodal relocation in the tip-gap, particularly downstream of the passage shock, the resulting flow does not seem to have significantly changed.

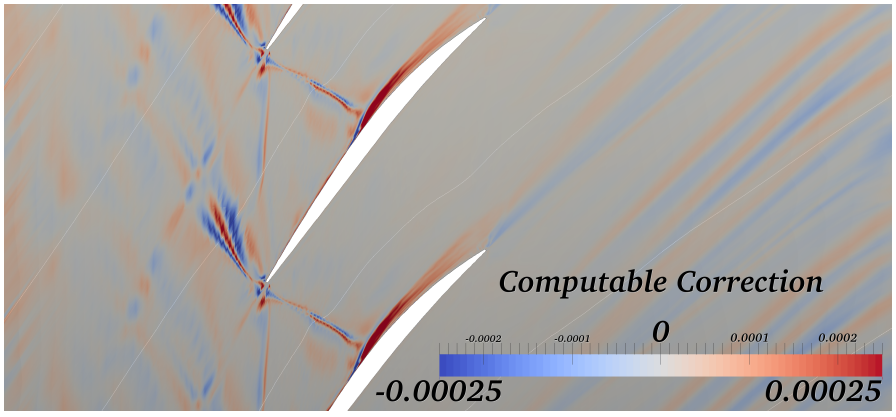
7.4.2 Part II: Adjoint-Based Cell Refinement

Following the repeated feature-based adaptation steps, from then onwards cell-based *non-computable correction* refinement was employed. The *computable correction* was used to improve the functional estimate of the embedded mesh required to evaluate the error maps. Each refinement step added roughly 300 $\sim$ 400k of nodes, and the wall normal refinement propagation was set to 10 layers.

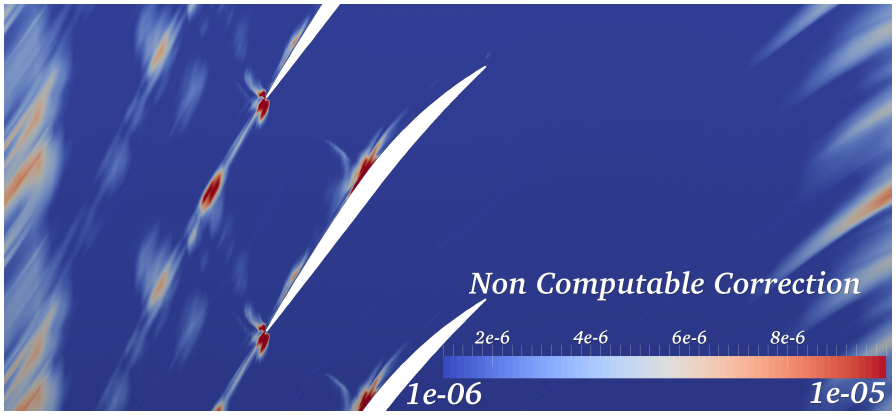
7.4.2.1 Adiabatic Efficiency (Mass Averaged)

Starting from the 15^{th} feature-based adapted grid, both types of adjoint error were determined on this grid. The resulting sensor maps at mid-span are reported in figure 7.20. The two are very different, in fact the *computable correction*, appears to have highlighted all the flow and adjoint features. Shocks, separation and wake are clearly visible, with the parts having been attacked by the previous feature-based adaptation being smaller in size. On the other hand, the usual adjoint upstream weighting is present, and is particularly strong where the *reversed wake* intersects the shock propagation. It is interesting to note that the inter-block boundaries between the upstream and passage blocks are clearly visible in the sensor map. While this error indicates where the functional is linearly affected by the residuals, the *non-computable correction* is indicating where the nonlinear sources of inaccuracies are. In this case the blade *LE*, where the shock forms, the location of the separation and the *reversed wake*. Again, for what concerns the latter, it is highest where it intersects the shock propagating across the periodic boundaries. Interestingly, the surfaces of integration of the quantity of interest seem to be subject

to a considerable measure of error. This is due to the adjoint residuals weighting, and arguably should have been filtered out. However, in this case it was left untouched.
An important point concerns the blade's *LE* error. In fact it is very high in this regions for both types of sensors. Close inspection of this zone of the mesh shows that the inter-block boundary corner point on the pressure side is very stretched causing a significant amount of numerical dissipation (see figure 7.21a). This is then propagated downstream. The feature based approach has not been able to remove this fake feature present in the starting mesh (figure 7.21b). Actually, careful comparison shows that it has possibly enhanced the numerical viscosity in this region. Effectively, feature-based adaptation is unable to detect whether flow complexities, such as this, are real or generated by mesh distortion. The adjoint process, on the other hand, has successfully flagged this on the 1st step (figure 7.21c).



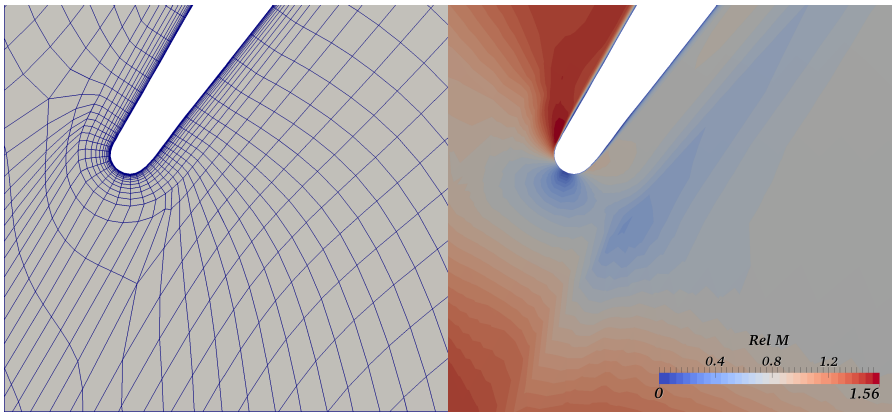
(A) *Computable Correction.*



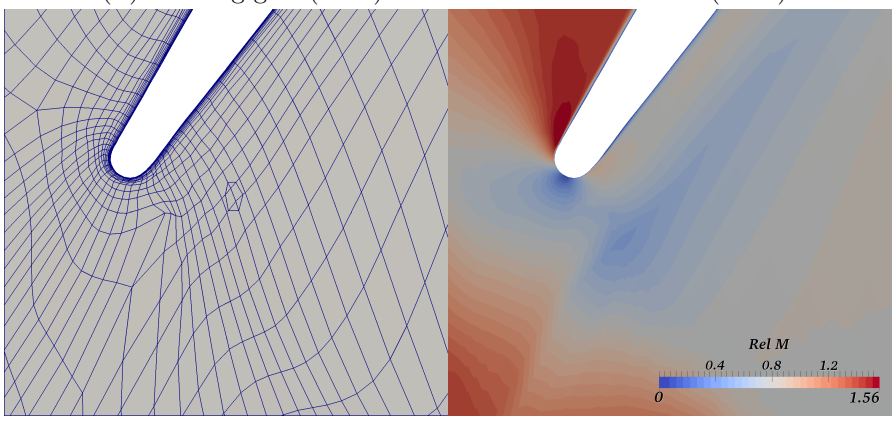
(B) *Non-Computable Correction.*

FIGURE 7.20: Adjoint error at mid-span after repeated feature-adaptation applied to *NASA Rotor* 37 .

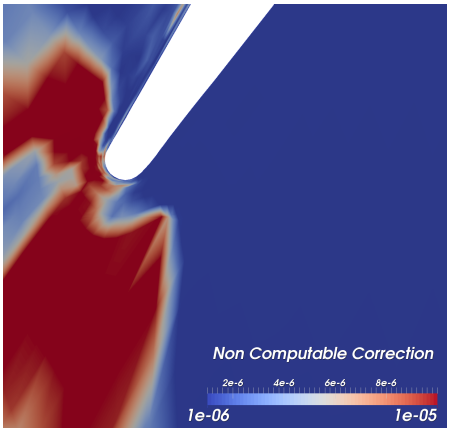
By applying the refinement based on the *non-computable correction*, the grid in figure 7.22 was obtained. The regions previously identified as having larger error have been marked. The suction side of the blade, starting from the *LE* to the initial part of the wake, has been refined. Moreover, it can be seen how it slightly extends along



(A) Starting grid (*LHS*) and relative *Mach* number (*RHS*).



(B) Feature adapted grid (*LHS*) and relative *Mach* number (*RHS*).



(C) Feature adapted mesh *non-computable-correction*.

FIGURE 7.21: *NASA Rotor* 37 *LE* mesh highlighting corner point issue.

the passage shock. The *reversed wake* flagged regions have also been targeted for the adaptation. Slight refinement has occurred at the wake just upstream of the outflow, but more significantly at the inlet plane. It is interesting to note that the blade pressure side and the majority of the wake are left untouched by the process. In the first case, this is indicative of the fact that there is a lower adjoint sensitivity in this regions. In the second case, the starting grid and consequent mesh movement seem to have sufficiently resolved the wake.

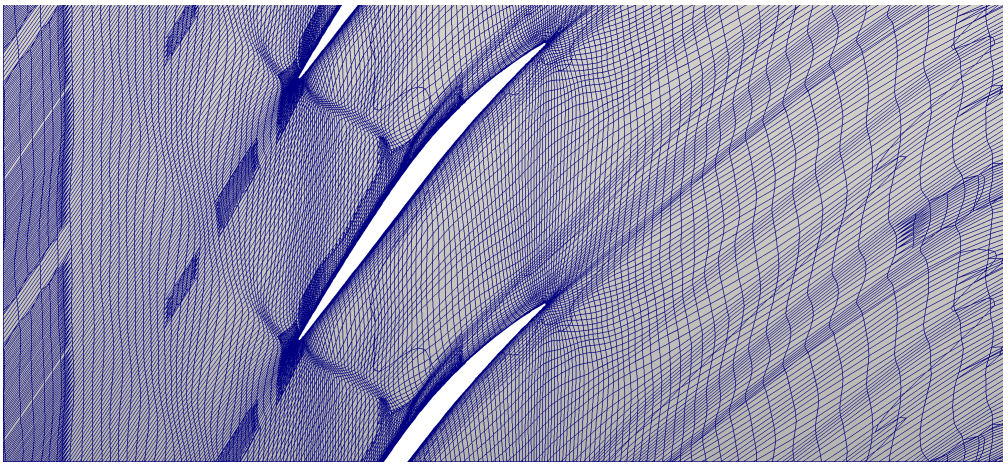


FIGURE 7.22: *NASA Rotor* 37 grid at mid-span after the 1st η-adjoint refinement step.

As mentioned earlier, there is a significant amount of error at the blade's *LE* due to grid stretching. This had been highlighted by the *non-computable correction* map. The resulting grid and flow field in this region are reported in figure 7.23. By comparing the relative *Mach* number on this mesh with that of figure 7.21b, there has been a reduction in this erroneous behaviour. Even so, it is clear that further refinement is necessary to appropriately remove this source of inaccuracies.

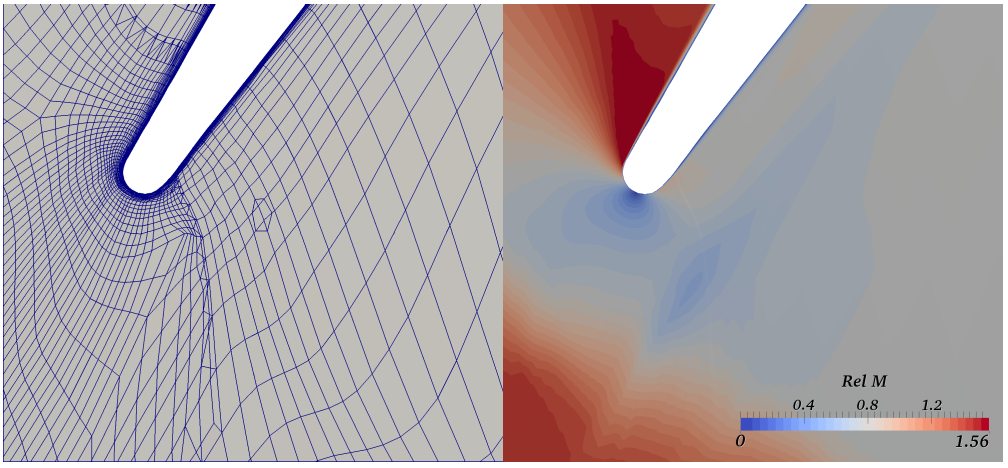
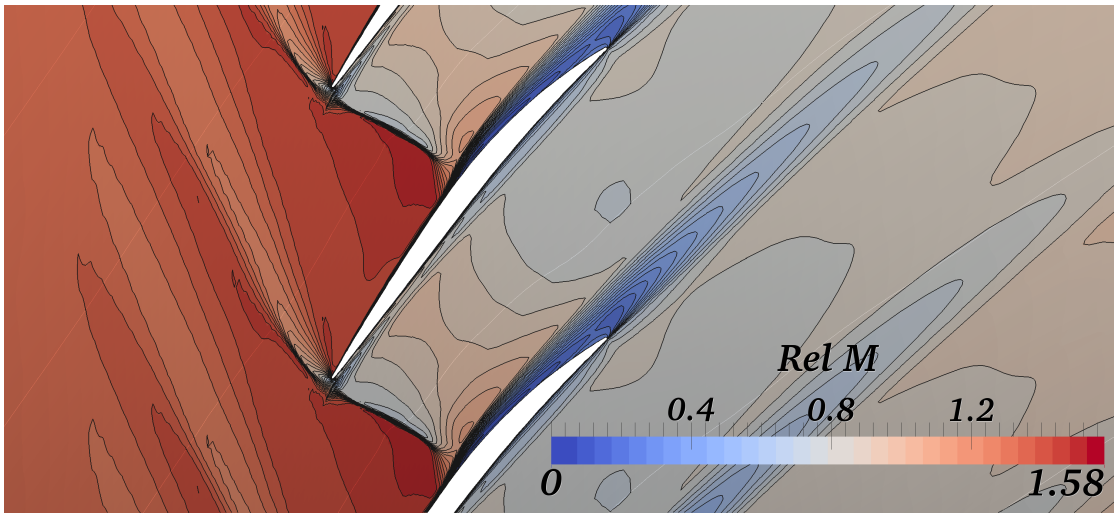


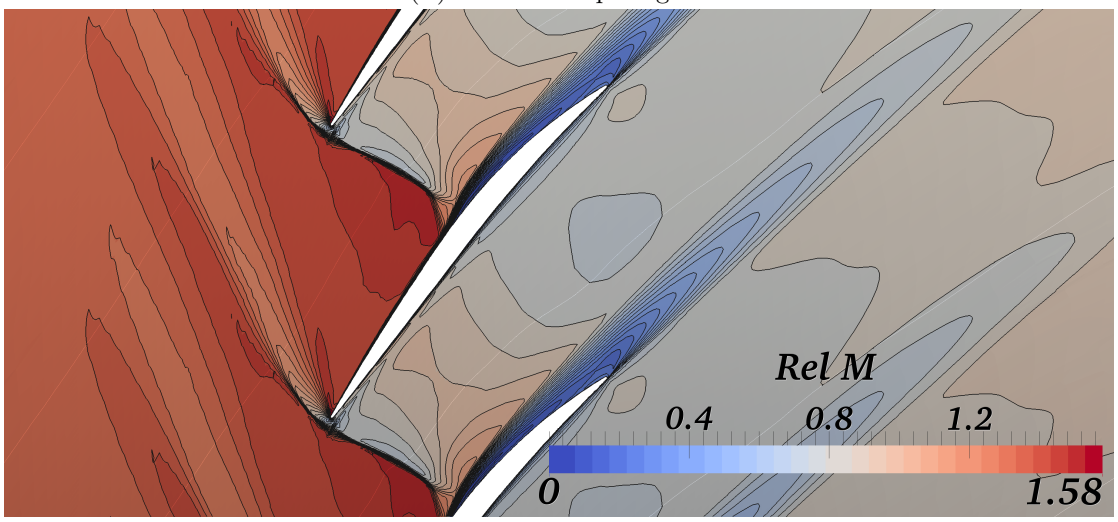
FIGURE 7.23: 1st η-adjoint refinement step grid (*LHS*) and relative *Mach* number (*RHS*).

Finally, for what concerns the 1st adjoint refinement step, a qualitative discussion concerning the flow field is necessary. To this end figure 7.24 is proposed, where 7.24a is the relative *Mach* number at mid-span after the feature adaptation as in figure 7.15b. This has been reproduced a second time to simplify comparison. Figure 7.24b is the relative *Mach* number at the same location for the flow solution achieved on the 1st adjoint-refined grid. Careful inspection shows how the lambda shape of the shock where it interacts with the boundary layer has improved in resolution. Moreover, the flow at the blade *LE* is visibly better captured. Concerning the upstream block of the mesh, it

can be seen how the contour lines around the shocks propagate slightly less *w.r.t.* the final feature-adaptation step.



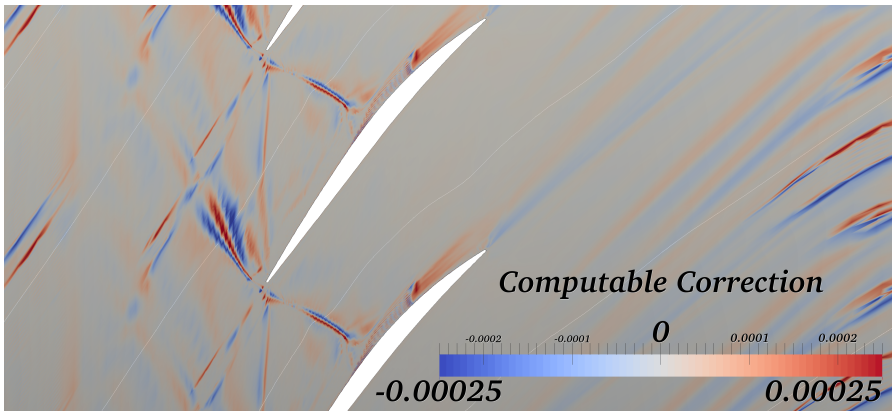
(A) Feature-adapted grid.



(B) 1st η-adjoint-adapted grid.

FIGURE 7.24: Comparison of relative *Mach* number at mid-span between feature- and 1st η-adjoint-adapted grids for *NASA Rotor* 37.

The resulting adjoint error maps are shown in figure 7.25. The first noticeable change, is the significant reduction in error where the refinement patches have been applied. The boundary layer on the blade suction side, *LE* and portions of the reversed wake, have all been subject to a decrease in the inaccuracies. Concerning the parts of the domain that are still highlighted as requiring refinement, a similar discussion to that relating to the previous adjoint error maps can be made. In fact, the *computable correction* will highlight the flow and adjoint features, while on the other hand, the *non-computable* version will flag limited regions of the flow. This is indicative of the fact that there is a reduction of the nonlinearities affecting the functional of interest.



(A) *Computable Correction.*

(B) *Non-Computable Correction.*

FIGURE 7.25: Adjoint error at mid-span after 1st η-adjoint refinement step for *NASA Rotor* 37.

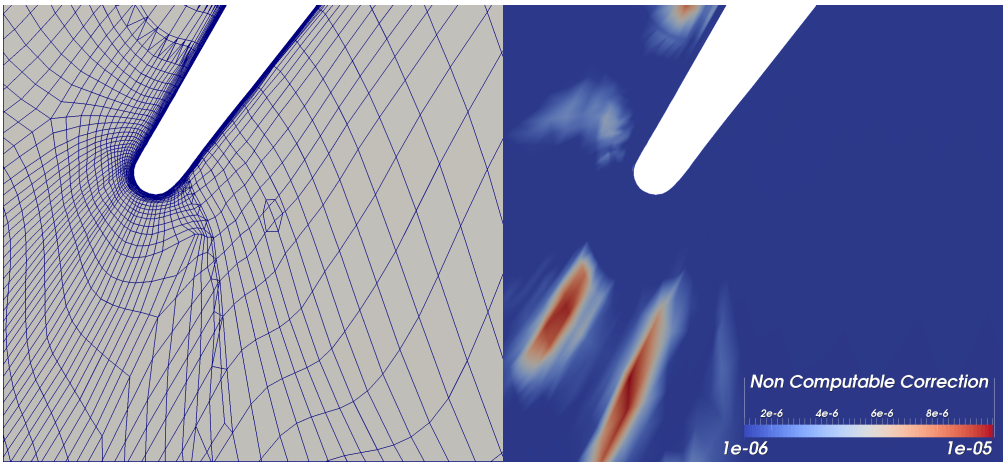


FIGURE 7.26: 1st η-adjoint refinement step grid (*LHS*) and *non-computable correction* (*RHS*).

As discussed, there is still the blade *LE* issue due to the corner point stretching in-between the block boundaries. Figure 7.26 shows how, after refinement, the *non-computable correction* has sensibly reduced in the region where sources of inaccuracies are generated. Nevertheless, there are still two patches that need a finer grid. One of

them corresponds to the corner point, while the other is due to the shock requiring extra resources to be appropriately captured.

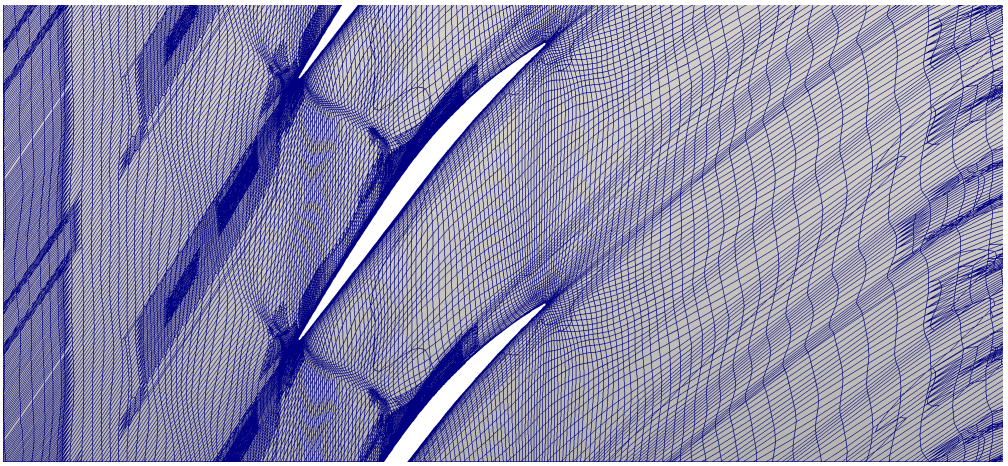


FIGURE 7.27: *NASA Rotor* 37 grid at mid-span after the 2nd η-adjoint refinement step.

The resulting adapted grid is reported in figure 7.27. Clearly, the adaptation has joined all the previous refinement patches in a continuum. In fact the finer patch starts from roughly halfway along the inlet block and then propagates towards the blade *LE*, onto the suction-side, till the first portion of the separation, just downstream of the shock. The cell marking has clearly increased the node density at the interface between shock and boundary layer. This is also the case for the discontinuity propagation at the blade *LE* in the counter-rotational direction. The refinement benefits are clearly visible at the problematic region at the blade *LE* (figure 7.28). The numerical viscosity has now been visibly reduced. The issues related to mesh distortion in this region are now minimal.

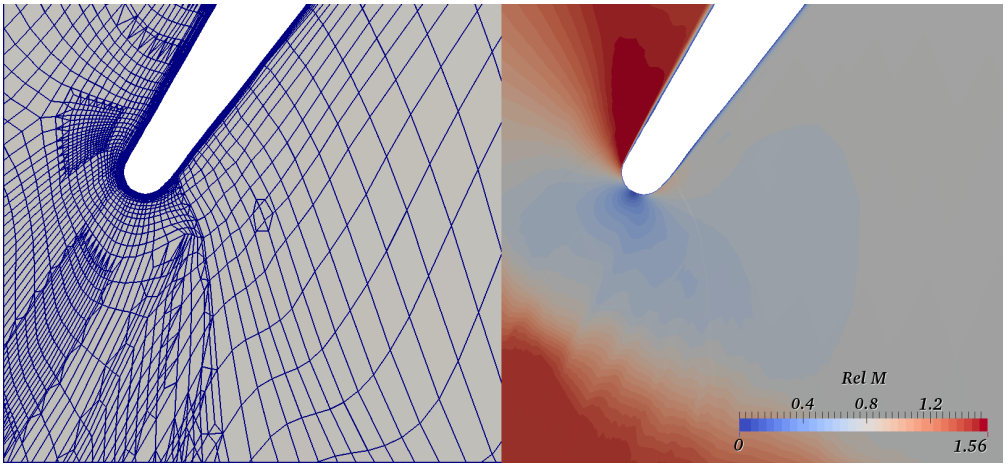
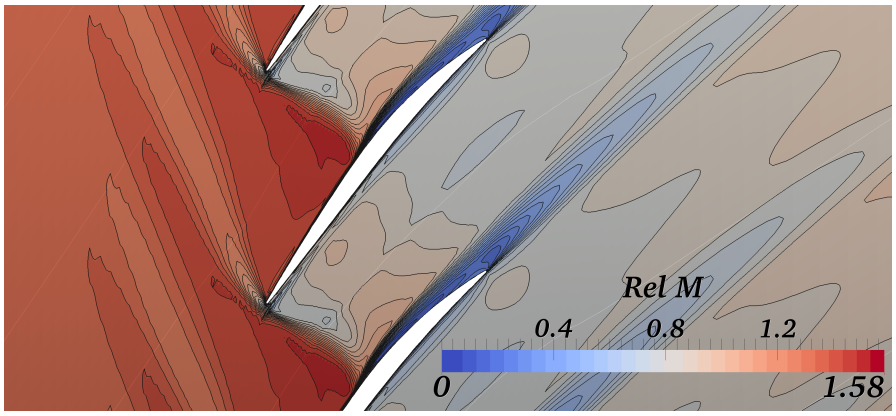
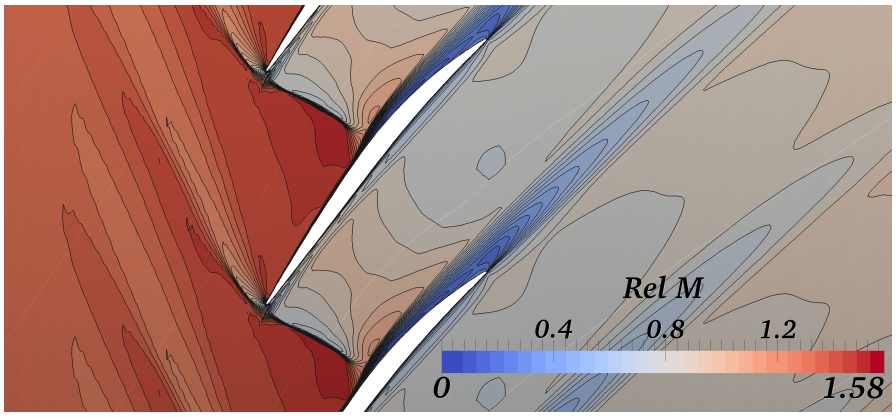


FIGURE 7.28: 2nd η-adjoint refinement step grid (*LHS*) and relative *Mach* number (*RHS*).

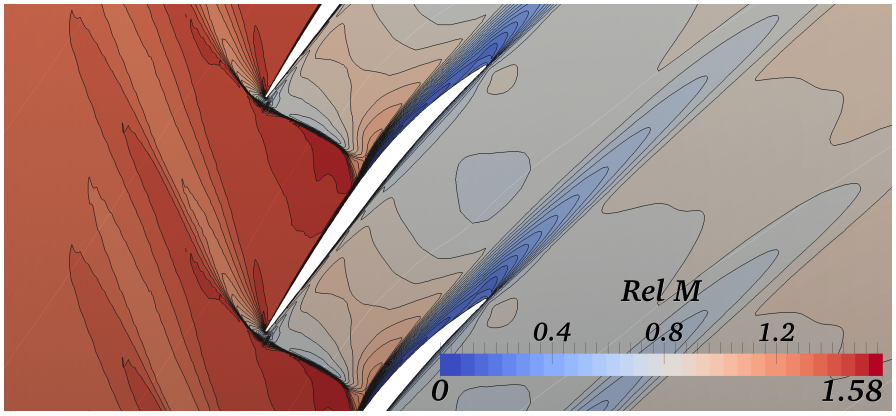
To be able to see the improvements from a qualitative point of view, the flow at mid-span for starting, feature-adapted and both adjoint-refined grids is proposed in figure 7.29.



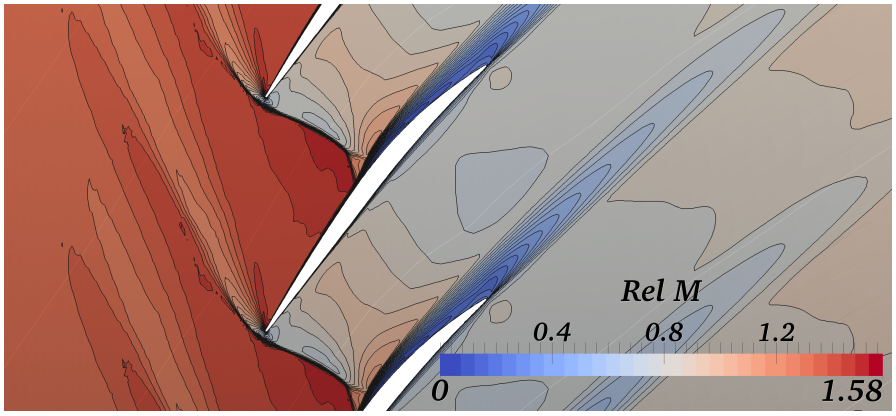
(A) Starting grid.



(B) Final feature-adaptation.



(C) 1st adjoint refinement.



(D) 2nd adjoint refinement.

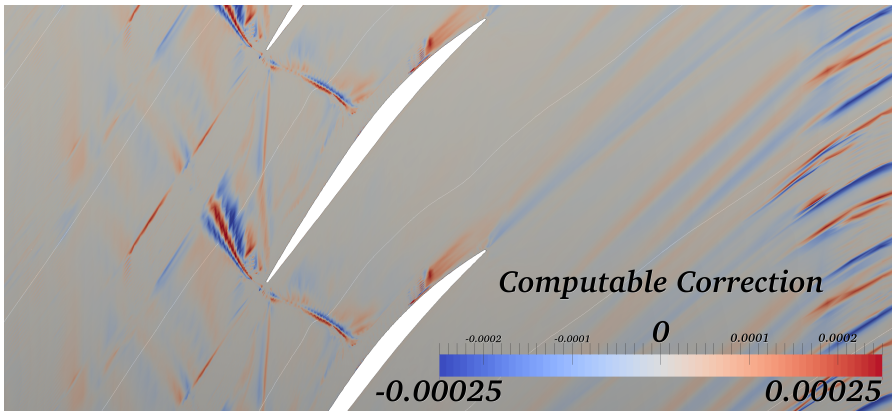
FIGURE 7.29: *NASA Rotor* 37 evolution of the flow field throughout mesh adaptation for η improvement.

The main flow complexities are much better resolved at the final step, with shock propagation contours being tightly bunched-up where it is strongest. An increase in the strength of its propagation into the upstream regions is also visible. More importantly, though, the shock/boundary layer interaction structure is better resolved, with both branches of the discontinuity being visible, unlike in the starting mesh where no sign of this particular conformation is visible. Moreover, the numerical diffusion at the blade *LE* has been significantly reduced and minimised during the last adaptation step. Not much change could be noted at the wake *w.r.t.* the starting grid. As previously mentioned, this is attributable to relatively good resolution/alignment of the starting grid, with any inaccuracy being dealt with by the mesh movement. In fact, careful analysis showed a slight thinning of it after the nodal relocation. This is confirmed by the adjoint error maps that show limited error in this region.

To complete the analysis, the adjoint solver had to be run on this grid to determine the *computable correction* and see where the remaining sources of inaccuracies were. The two adjoint sensors resulting from the latest adapted grid are shown in figure 7.30. The *computable correction* had a larger amount of higher inaccuracies *w.r.t.* the *non-computable* counterpart. This is still the case, however, as in the previous step, where the refinement has occurred, the error has been sensibly reduced. To this end, it is clear how no inaccuracy is flagged on the blade suction-side and *LE*. On the other hand, the other type of sensor map in figure 7.30b, has been clearly decreased with only a few points being visible.

To see how the large issues at the blade *LE* have decreased, a highlight of the mesh and *non-computable correction* is reported in figure 7.31. No significant error can be seen, apart from two straight bands outside one of the refinement patches. By comparing this result with that in figure 7.21c, the improvement is clear.

The most important discussion concerns the adiabatic efficiency computation after each step and at the overall process's end. In table 7.3 the quantity is reported for the starting, 15^{th} feature-based moved, the two adjoint-based refined and the target grids performance up to 4 decimal places in accuracy. Additionally, the value obtained from a fully-feature based approach is included (this is discussed in detail in appendix C). The third column is the η value computed on the mesh by running *Hydra* to convergence. Additionally, the fourth column contains the corrected value. This is achieved by generating an embedded grid and quadratically interpolating the flow and adjoint solutions onto it. Consequently a single flow solver iteration is applied with a *Courant-Friedrichs-Lewy* (*CFL*) number of 0.0 to determine the functional value and the residuals. Once these have been evaluated the *computable correction* can be calculated by multiplying these by the interpolated adjoint. This is then subtracted from the performance quantity computed on this embedded grid.



(A) *Computable Correction.*

(B) *Non-Computable Correction.*

FIGURE 7.30: Adjoint error at mid-span after 2nd η-adjoint refinement step for *NASA Rotor* 37.

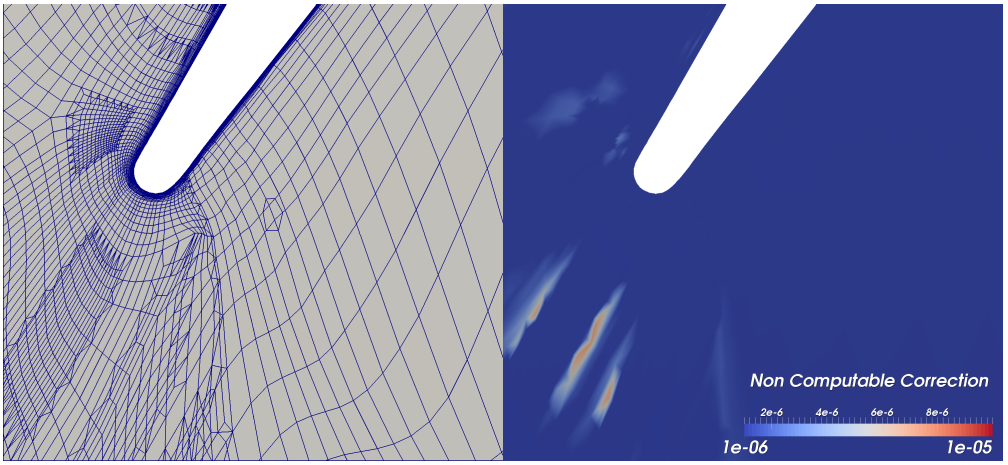


FIGURE 7.31: 2nd η-adjoint refinement step grid (*LHS*) and *non-computable correction* (*RHS*).

As previously discussed, the feature-based mesh movement deteriorated the quantity's value. This was attributed to the starting grid not being in *asymptotic range*, due to the numerous regions in the flow that were inappropriately replicated (e.g. shock resolution and its interaction with the boundary layer), but also by numerical dissipation such as

Grid	$N°Nodes$	η (%)	Corrected η (%)	Pr	$\dot{m}$ ($\frac{kg \cdot m}{s}$)
Start	670056	84.4100	– – – – – – –	2.0815	20.0191
Final Feature Moved	670056	84.1287	84.2834	2.0729	19.9638
1st η-Adjoint Adapt.	1017847	84.6963	84.8354	2.0934	20.1338
2nd η-Adjoint Adapt.	1392622	84.9296	84.9945	2.1004	20.1911
Fully Feature Adapt.	2694775	84.3509	– – – – – – –	2.0805	20.0289
Target	46763130	84.9950	– – – – – – –	2.0963	20.1756

TABLE 7.3: *NASA Rotor* 37 evolution of mass averaged adiabatic efficiency between inlet and outlet throughout the feature and adjoint combined adaptation process.

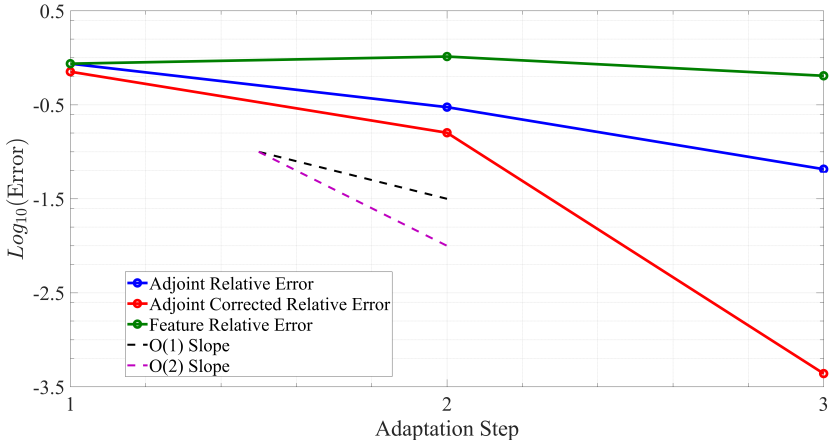


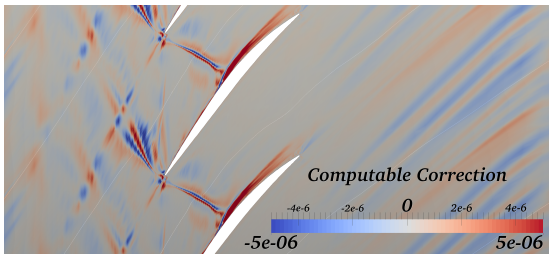
FIGURE 7.32: η error convergence of the final (15th) feature-based moved and two adjoint refined grids along with respective corrected values. As a comparison the result of a fully-feature-based approach has been included.

the inter-block boundary on the blade's pressure-side *LE*. While the former could be improved by the relative *Mach* number mesh movement, the latter was attacked and reduced by the adjoint-based refinement. Moreover, the alignment of the grid's edges with the flow natural *anisotropy* paid-off when the adjoint procedure was employed since the inaccuracies were limited in size. For this reason, the adjoint-refinement is able to remove the effect of nonlinearities significantly improving the functional value. By including the *computable correction* to the embedded grid's estimate, the efficiency sensibly improves and matches that of the target grid up to two decimal positions. As the *computable correction* effectively indicates where the flow residuals linearly influence the functional, the reason behind the corrected value being so accurate, is due to the fact that nonlinearities have been minimised. Nevertheless, even the value on the final adapted grid showed significant improvement and is not far off the target. It should be noted that these results have been achieved starting from a relatively coarse grid ([48] started their adaptation from a 2.8M mesh) and achieved very satisfactory accuracy by doubling the node count. Unfortunately, the fully feature based approach has performed quite poorly. While it was expected to have a worse efficiency estimate, such a large difference, particularly given the large amount of nodes, was unexpected. It is thought,

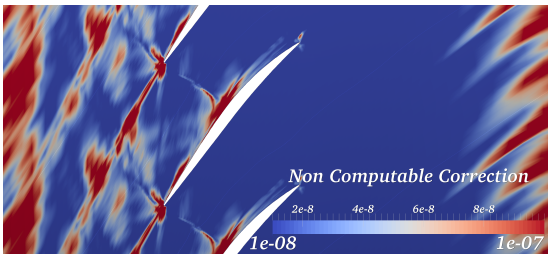
though, that further adaptation steps would significantly reduce the variation.
To be able to see the error rate of convergence the plot in figure 7.32 is provided. The error values are determined by subtracting the adapted or embedded mesh estimate from the target performance. The grid achieved by adjoint-based refinement, has an approximately first order convergence of the functional error per adaptation step. On the other hand, the corrected one has a slightly better initial rate of convergence, that then visibly improves beyond second order. As discussed, the fully feature-based technique is visibly under-performing. A description of the reasons behind this is provided in appendix C.

7.4.2.2 Absolute Total Pressure Ratio (Mass Averaged)

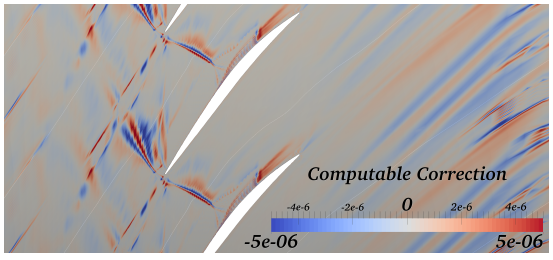
The second functional employed for adjoint-based adaptation is the mass averaged absolute total pressure ratio between inlet and outlet. The evolution of both error maps at mid-span evaluated on the last feature-adapted grid and the two consequent refined meshes are shown in figure 7.33. The reader will have immediately noticed that the there is a significant change in the order of magnitude of the inaccuracies *w.r.t.* that of the efficiency case. The adjoint solver is dimensional, and therefore the dual solution of the efficiency will have dimensions of %, i.e. two orders of magnitude greater than that of the pressure-ratio, that is a dimensionless quantity. Therefore, for the efficiency, errors are simply scaled by a factor of 100. Concerning the starting *computable correction*, this is effectively very close to that of the η-case. In fact both flow and adjoint features are highlighted with a similar weighting. This is not the case for the *non-computable* version. In fact, for the Pr, all zones of inaccuracies are stronger in magnitude. Moreover, there appears to be a slightly larger sensitivity in the passage region. This is the reason behind the extended refinement off the blade suction-side towards the passage, as shown in figure 7.34a. Apart from this characteristic, the previously analysed functional had a very similar refined grid after the 1^{st} adjoint adaptation step.
The error maps evaluated on this grid show a similar behaviour as that of the previous performance quantity. As desired, where the refinement has occurred, the inaccuracies magnitude has significantly decreased, creating a void in the overall sensor field. Nevertheless, while the *computable correction* highlights the strongest parts of shock and wake, the *non-computable* counterpart, shows high inaccuracies are forming at the adjoint *reversed wake*. Moreover, unlike the efficiency case, there are high residuals occurring around the refinement patches, particularly outside that on the suction-side. These can be attributed to the interface between fine and coarse grid regions.



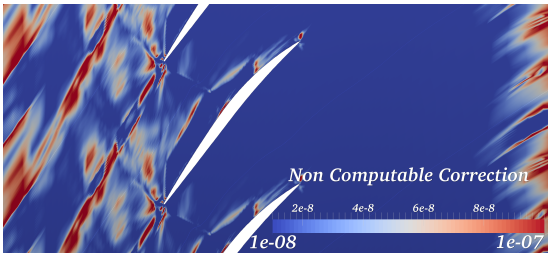
(A) *Computable correction* 15^{th} feature-moved mesh.



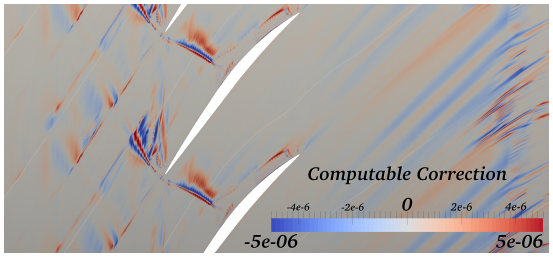
(B) *Non-computable correction* 15^{th} feature-moved mesh.



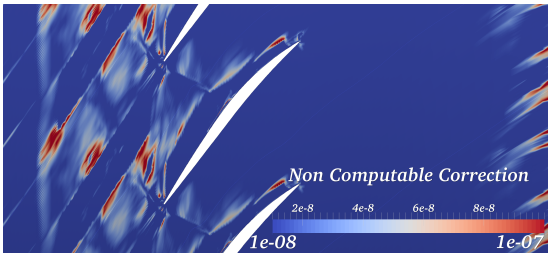
(C) *Computable correction* 1^{st} adjoint-refined mesh.



(D) *Non-computable correction* 1^{st} adjoint-refined mesh.



(E) *Computable correction* 2^{nd} adjoint-refined mesh.



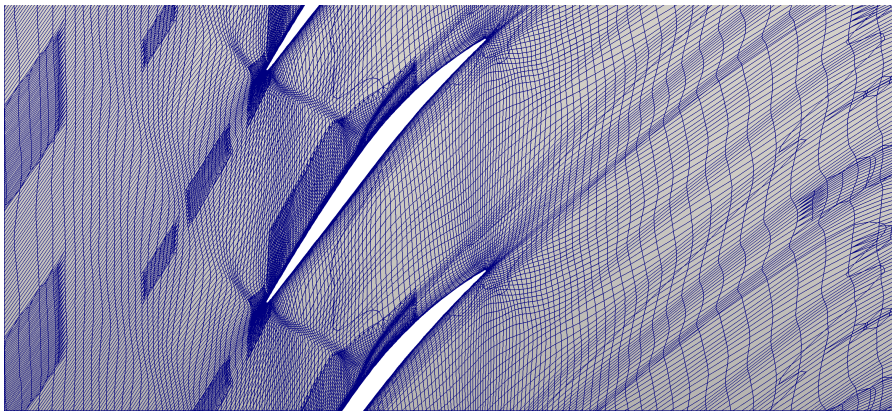
(F) *Non-computable correction* 2^{nd} adjoint-refined mesh.

FIGURE 7.33: Evolution of the mass averaged absolute total pressure-ratio adjoint errors at mid-span of *NASA Rotor* 37 after the feature- and *Pr*-adjoint-based mesh adaptation.

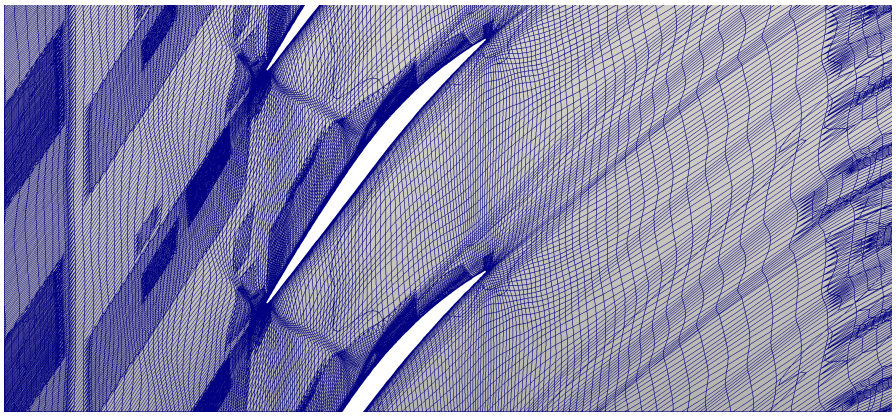
In fact, not only will the control volume size abruptly change, but also the presence of hybrid interfaces to remove *hanging nodes* causes issues, in terms of size, shape and orientation of control surfaces. Unfortunately, there is no way of knowing where interface patches ought to be placed to avoid causing issues in the following flow solution.

In the next adaptation step, the cell marking algorithm has created a straight block of refinement in the upstream parts of the mesh (figure 7.34b). The passage fine block has also been stretched towards the flow wake. Finally the blade *LE* shock has also been targeted. As in the efficiency case, no nodes have been placed on the pressure side of the passage, as well as most of the wake.

Evaluation of the adjoint-based errors on this grid shows how they have been sensibly decreased (figures 7.33e and 7.33f). Again, voids in the maps are clearly visible where the refinement patches are. Moreover, as was previously noted, the *computable correction* shows a larger amount of error, that unfortunately has increased downstream of the shocks. This is also the case in the *non-computable* version. This is probably due to the



(A) 1st step.



(B) 2nd step.

FIGURE 7.34: *NASA Rotor* 37 adjoint-*Pr* refined grids at mid span after the feature- and *Pr*-adjoint-based mesh adaptation.

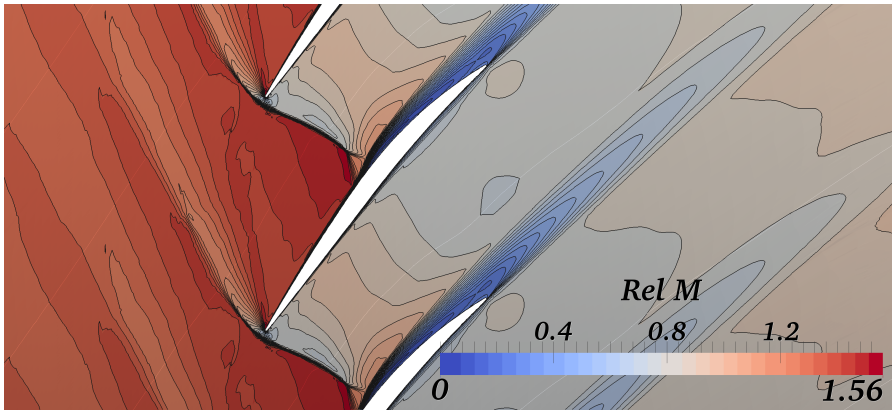
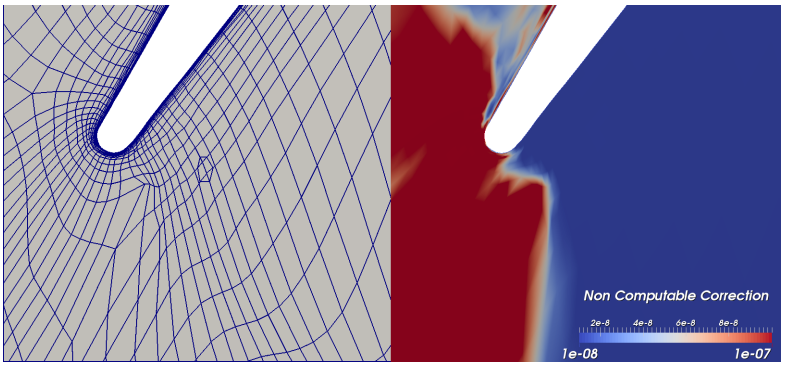


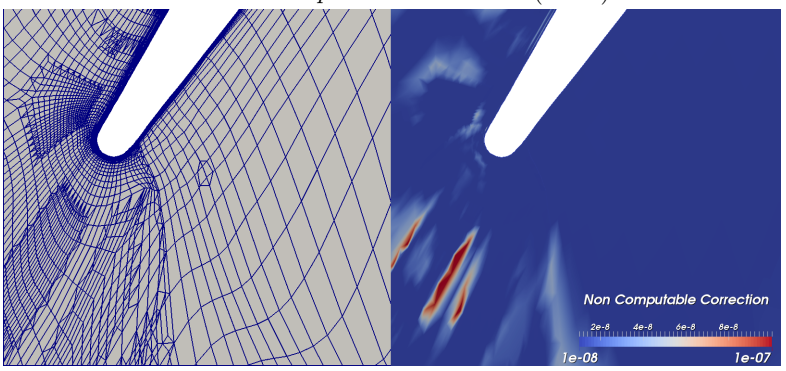
FIGURE 7.35: Relative *Mach* number at mid-span for the 2nd *Pr*-adjoint-adapted grid for *NASA Rotor* 37.

hybrid interface regions in the upstream part of the mesh. In fact, large sources of error may be seen. These are then convected downstream by the flow equations, increasing error, particularly where flow complexities are present. Unfortunately, unlike the previous case, a minimisation of the nonlinearities can not be achieved in two refinement steps, and further adaptation is required. However, the current refinement strategy is

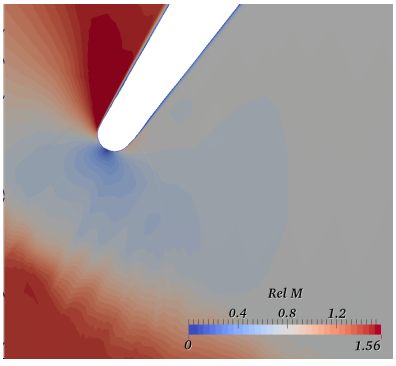
possibly unsuited for the task, as these hybrid elements will be split causing increasingly distorted shapes (negative volumes may also appear). For this reason, the *computable correction* term will not be as accurate as in the previous case.



(A) Grid resulting from last feature-based mesh movement (*LHS*) and *non-computable correction* (*RHS*).



(B) 2nd adjoint-*Pr* refined mesh (*LHS*) and *non-computable correction* (*RHS*).



(C) 2nd adjoint-*Pr* refined mesh relative *Mach* number distribution.

FIGURE 7.36: Evolution of the blade *LE* error and solution during the *Pr*-adjoint adaptation.

Qualitatively, figure 7.35 shows the final grid's solution at mid-span. Comparing with the starting point, i.e. the last result of the feature-based mesh movement in figure 7.15b, improvements are clearly visible. In fact, as in the efficiency case, the refinement has allowed a better resolution of the shock, particularly at the blade *LE*. Moreover, its

Grid	$N°Nodes$	Pr	Corrected Pr	η (%)	$\dot{m}$ ($\frac{kg \cdot m}{s}$)
Start	670056	2.0815	$------$	84.4100	20.0191
Final Feature Moved	670056	2.0729	2.0843	84.1287	19.9638
1st Pr-Adjoint Adapt.	1009019	2.0960	2.0958	84.7770	20.1516
2nd Pr-Adjoint Adapt.	1395880	2.0977	2.0909	84.8468	20.1648
Fully Feature Adapt.	2694775	2.0805	$------$	84.3509	20.0289
Target	46763130	2.0963	$------$	84.9950	20.1756

TABLE 7.4: *NASA Rotor* 37 evolution of mass averaged absolute total pressure ratio between inlet and outlet throughout the feature and adjoint combined adaptation process.

interaction with the boundary layer on the suction-side and the propagation towards the upstream regions is also improved. Another point that was previously mentioned, concerned the stretching of the *LE* inter-block boundary corner point on the blade pressure side. As shown in figure 7.36a, the *non-computable correction* based on the Pr-adjoint solution immediately flags the entire regions as being problematic. As in the previous functional's case, the last refinement step significantly reduces any issues caused in this region, as shown by the error map in figure 7.36b and the resulting flow field shown in image 7.36c.

Finally, the most important part of the analysis concerns the evolution and improvement of the quantity of interest. As shown in table 7.4, the adjoint-refinement step sensibly improves the functional *w.r.t.* the last feature-based movement. Effectively, the quantity is so close to target that the difference between the adapted mesh and the corrected quantity is $2 \cdot 10^{-4}$. Unfortunately, the following refinement step, as discussed, causes errors to appear at the interfaces between refined and coarse mesh regions. These not only reduce the Pr accuracy, but also increase the nonlinearities present in the flow. Therefore the *computable correction* term is not as accurate as expected. Nevertheless, the outcome is again much better than that of the fully-feature based process (see appendix C), with roughly half the amount of nodes.

The error relative to the target performance of the adapted grids is shown in figure 7.37. The 1st point relates to the last feature-based mesh movement iteration. The resulting grid is then employed in both the fully feature (green line) and combined adjoint (blue line) processes. The last curve (red) represents the corrected functional using the embedded grid process. While the values obtained employing the standard adjoint-adaptation procedure visibly reduced the error to start with, as discussed, the second refinement step unfortunately inverted the trend. Nevertheless, it can be concluded that they consistently outperform the feature-based procedure.

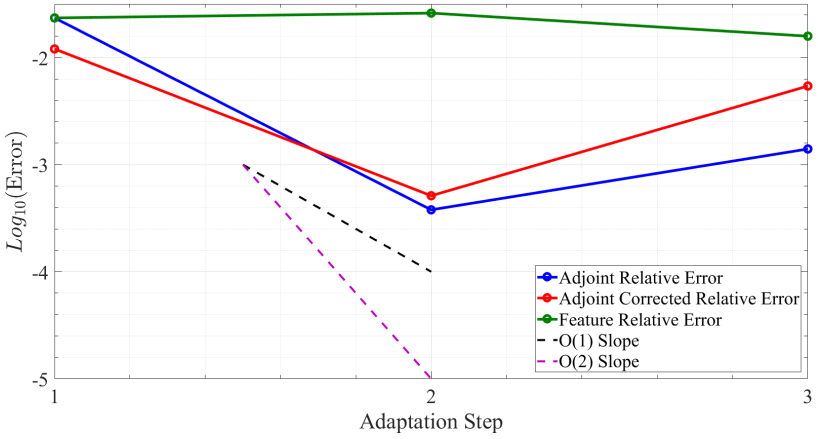


FIGURE 7.37: *Pr* error convergence of the final (15^{th}) feature-based moved and two adjoint refined grids along with respective corrected values. As a comparison the result of a fully-feature-based approach has been included.

7.4.2.3 Mass Flow (Area Averaged)

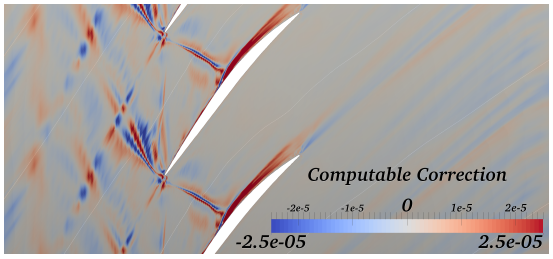
The final quantity of interest was the area averaged mass-flow evaluated on the exit plane. As for efficiency and pressure-ratio, the starting point was the 15^{th} feature-based mesh movement step.

The resulting adjoint-error maps are displayed in figure 7.38. Similarly to previous quantities of interest, the *computable correction* values are representative of the combination of both flow and adjoint features. Moreover, the regions of error are more spread out *w.r.t.* that of the *non-computable* version. In particular, the latter again highlights sources of inaccuracies upstream of the blade, along with the suction-side of it. Comparing the starting error fields in figures 7.38a and 7.38b with those of the pressure-ratio in images 7.33a and 7.33b, the two closely resemble each other (apart from the adjoint dimensions being different). Strong similarities of these with the values computed in the case of the efficiency are also clearly visible (see figures 7.20a and 7.20b).

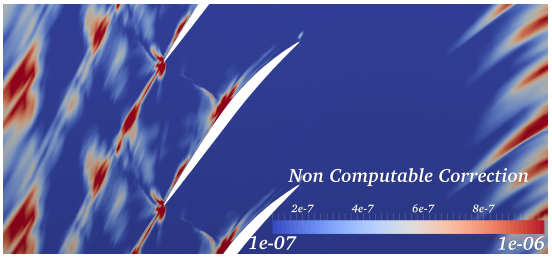
Concerning the 1^{st} refinement step, the grid shown in 7.39a, has the finer mesh patches appearing along the blade suction-side and the high sensitivity regions of the *reversed wake*. As in the case of the pressure-ratio, the refinement has been pushed further into the passage, indicating a higher sensitivity *w.r.t.* the shock. At the blade *LE* there is a continuous refinement patch, unlike efficiency and pressure-ratio cases.

The adjoint errors evaluated on this grid, shown in figures 7.38c and 7.38d, show how the refinement has successfully removed the main sources of inaccuracies, with voids appearing in both fields. Having said that, the issue with the hybrid interface patches seems to have appeared. In fact, looking at the *computable correction*, the error thickness around the strongest parts of the shock seems to have increased. This issue is also present, to a smaller extent, in the *non-computable* version. For this reason, in the following

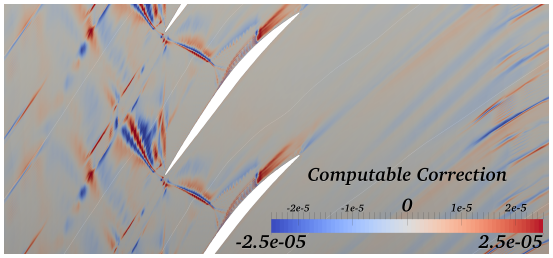
adaptation step (see figure 7.39b), the *LE* discontinuity propagating towards the inlet has been refined. As in previous cases, the upstream adjoint line of high sensitivity, has attracted a considerable amount of nodes. Moreover, as for the pressure-ratio, a small fine patch has appeared downstream of the separation, where the previous refinement ended. The error maps evaluated on this grid are reported in figures 7.38e and 7.38f. Despite removing the error present before any refinement had been carried out, there are some relatively high inaccuracies appearing around the hybrid patches between the fine and coarse mesh. This is particularly true where flow features appear (for instance at the wake).



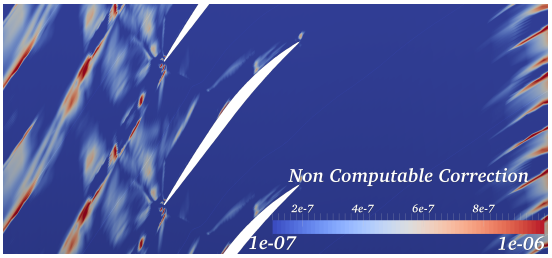
(A) *Computable correction* 15^{th} feature-moved mesh.



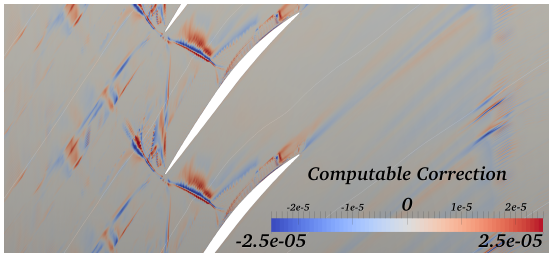
(B) *Non-computable correction* 15^{th} feature-moved mesh.



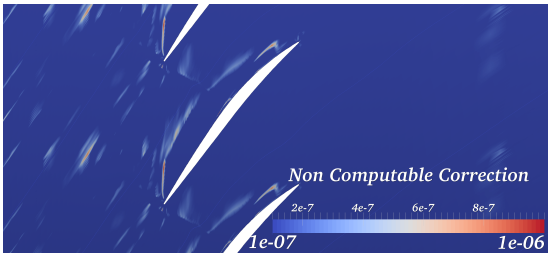
(C) *Computable correction* 1^{st} adjoint-refined mesh.



(D) *Non-computable correction* 1^{st} adjoint-refined mesh.



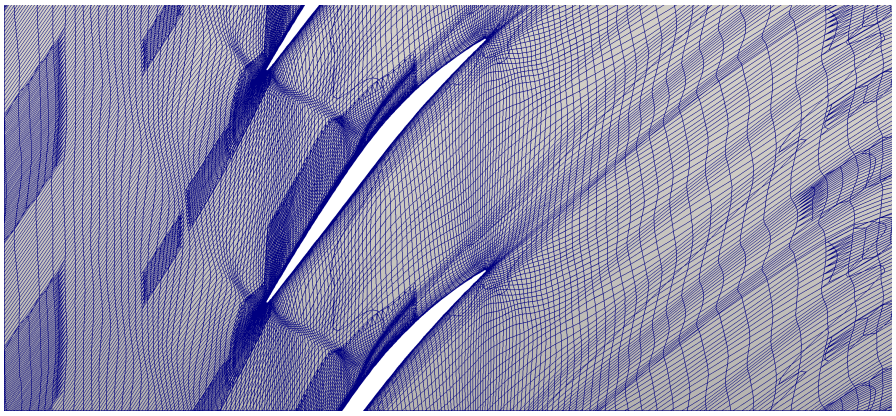
(E) *Computable correction* 2^{nd} adjoint-refined mesh.



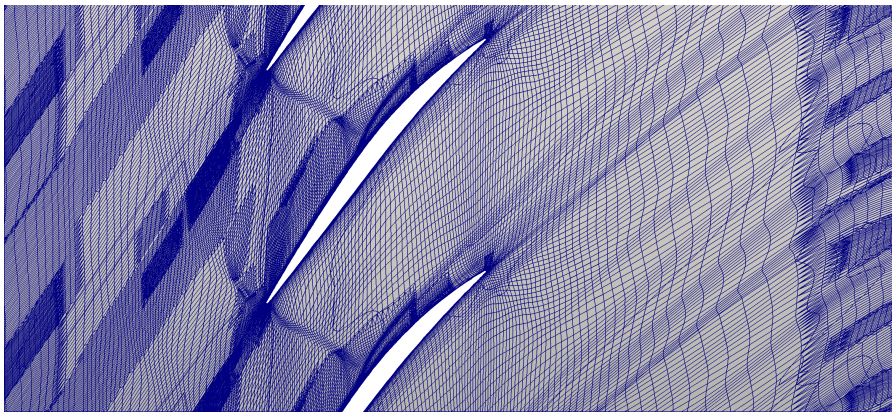
(F) *Non-computable correction* 2^{nd} adjoint-refined mesh.

FIGURE 7.38: Evolution of the area averaged mass-flow adjoint errors at mid-span of *NASA Rotor* 37 after the feature- and $\dot{m}$-adjoint-based mesh adaptation.

To be able to prove that hybrid elements required to maintain a conformal mesh are causing significant issues, *non-computable correction* thresholds of $5 \cdot 10^{-7}$ and $1 \cdot 10^{-7}$ were applied over the entire domain of the final mesh error. The result can be seen in figure 7.40. As expected, the main sources of inaccuracies are in the upstream block and on the suction-side. Moreover, these are concentrated around blade mid-span, not

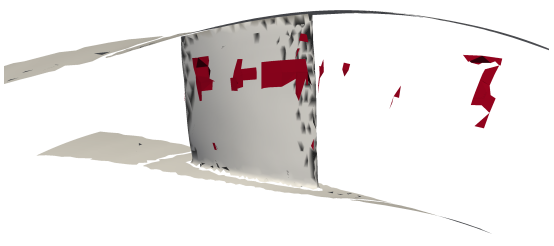


(A) 1st step.

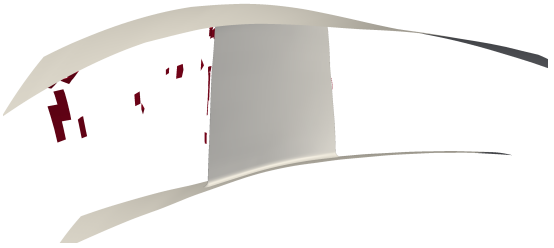


(B) 2nd step.

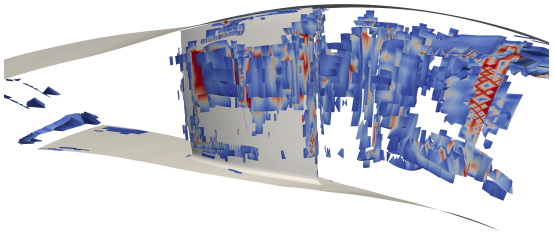
FIGURE 7.39: *NASA Rotor* 37 adjoint-$\dot{m}$ refined grids at mid span after the feature- and $\dot{m}$-adjoint-based mesh adaptation.



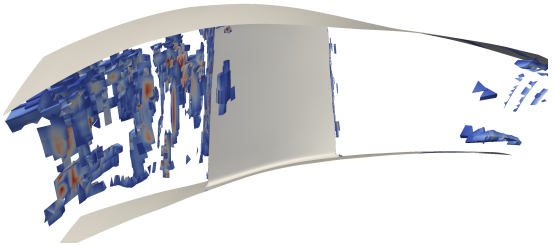
(A) Suction-side error$\geq 5^{-7}$.



(B) Pressure-side error$\geq 5^{-7}$.



(C) Suction-side error$\geq 1^{-7}$.



(D) Pressure-side error$\geq 1^{-7}$.

FIGURE 7.40: Last adapted grid, after feature-based mesh movement and $\dot{m}$-adjoint refinement, cells with high *non-computable correction*.

so much near the casing and even less in the hub vicinity. This is due to the convective terms being stronger further away from these. Extracting the cell types from these figures, it is possible to verify that hybrid elements are sources of errors. The results for a threshold of $5 \cdot 10^{-7}$ are reported in table 7.5, while those for the lower value of $1 \cdot 10^{-7}$, may be found in table 7.6. Both contain the individual contribution of tetrahedra, pyramids and prisms, along with the combined one ("*hybrid*" row). To be able to get a feel for the overall percentage of the various cell types in the final grid, table 7.7 has been included. In the case of the highest sources of error over the entire domain, it is clear that the hybrid cells dominate the contribution with more than 87% of them not being hexahedra. Reducing the threshold five-times, their contribution is still far greater than that of the hexahedra, with the former representing 2/3 of the overall high-error elements. Nevertheless, it should be noted that most of these cells actually neighbour each other, and therefore it is often the case that the error is generated in one and propagated to the other. Given that the hybrid elements are many more, it may be concluded that without these the overall error would be much less. By comparing the quantity of most inaccurate cells *w.r.t.* the overall count, shows that they are actually a minority. In fact, for a threshold of $5 \cdot 10^{-7}$ they represent 0.2%, while in the case of a threshold of $1 \cdot 10^{-7}$, they are only 2.2% of the total number of cells.

	N° cells Error $\geq 5e-7$	% *w.r.t.* Total
Grid total	2682	100
Tetrahedra	1107	41.275
Pyramids	1206	44.966
Prisms	33	1.230
Hexahedra	336	12.528
Hybrid Elements	2346	87.472

TABLE 7.5: Final $\dot{m}$-adjoint refined grid statics for cells with *non-computable correction* value greater than $5 \cdot 10^{-7}$.

	N° cells Error $\geq 1e-7$	% *w.r.t.* Total
Grid total	27009	100
Tetrahedra	8817	32.645
Pyramids	7532	27.887
Prisms	1747	6.468
Hexahedra	8913	33.000
Hybrid Elements	18096	66.999

TABLE 7.6: Final $\dot{m}$-adjoint refined grid statics for cells with *non-computable correction* value greater than 10^{-7}.

	N° cells	% *w.r.t.* Total
Grid total	1221333	100
Tetrahedra	109044	7.365
Pyramids	106328	7.180
Prisms	44161	2.982
Hexahedra	1221333	82.474
Hybrid Elements	259533	17.526

TABLE 7.7: Final $\dot{m}$-adjoint refined grid cell type statistics.

To conclude the discussion, the mass flow quantity evolvement throughout the process is reported in table 7.8. The error relative to the target quantity of the adapted, embedded and fully-feature based grids is shown in figure 7.41. As for the case of the pressure-ratio, the functional sensibly improves once the adjoint-refinement is employed. Unfortunately, the accuracy deteriorates slightly in the second adaptation step. According to the author, on the basis of the detailed analysis provided, the issue is due to the presence of hybrid elements causing an increase of the error quantities. Unfortunately, further refinement would not bring justice to the process, as continuous splitting of these elements in a bid to reduce the inaccuracies, would cause others to appear. As a remedy to this, the refinement strategies proposed by [116, 118] are suggested. These exploit a technique that would remove any hybrid elements requiring refinement and replace them with refined versions of the original grid's hexahedra.

Grid	$N^\circ Nodes$	$\dot{m}$ $(\frac{kg \cdot m}{s})$	Corrected $\dot{m}$ $(\frac{kg \cdot m}{s})$	η (%)	Pr
Start	670056	20.0191	– – – – – – – –	84.4100	2.0815
Final Feature Moved	670056	19.9638	20.0469	84.1287	2.0729
1^{st} $\dot{m}$-Adjoint Adapt.	999530	20.1455	20.1589	84.7572	2.0951
2^{nd} $\dot{m}$-Adjoint Adapt.	1337205	20.1339	20.1509	84.7216	2.0928
Fully Feature Adapt.	2694775	20.0289	– – – – – – – –	84.3509	2.0805
Target	46763130	20.1756	– – – – – – – –	84.9950	2.0963

TABLE 7.8: *NASA Rotor* 37 evolution of outlet area averaged mass-flow throughout the feature and adjoint combined adaptation process.

7.4.2.4 Comparison of other performance quantities

The evolution of quantities other than the one central to the adaptation were included in tables 7.3, 7.4 and 7.8. As it can be seen, the three parameters are improved regardless of which one is subject to optimisation. This is effectively consistent with the error maps shown for each one. In fact by comparing their both their *computable* and *non-computable* error maps they all highlight very similar regions (neglecting the order of magnitude difference). This resulted in similar adapted meshes with small differences

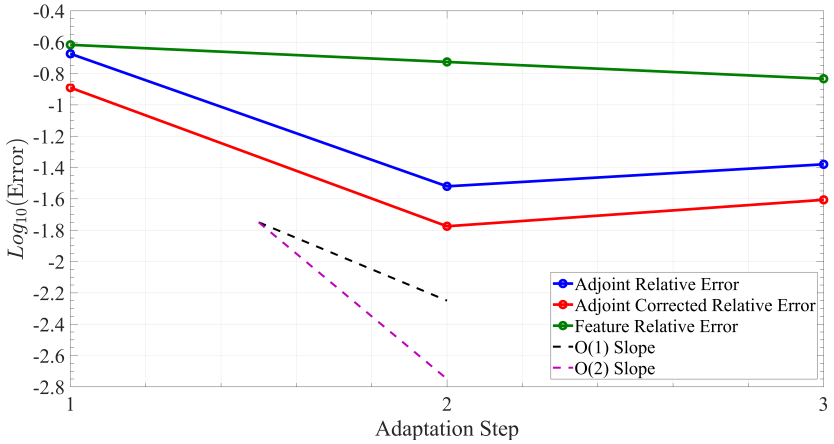


FIGURE 7.41: $\dot{m}$ error convergence of the final (15^{th}) feature-based moved and two adjoint refined grids along with respective corrected values. As a comparison the result of a fully-feature-based approach has been included.

concerning the propagation of the refinement into the passage. One of the reasons for this resemblance relates to surface of integration being the same for all cases (i.e. outflow). Additionally, both adiabatic efficiency and total pressure ratio are mass averaged, meaning that this ought to be correctly predicted in order to improve their accuracy. Finally, the adiabatic efficiency value is computed employing equation 7.1 and therefore total pressure and temperature ratios, as well as mass flow, must be accurate in order to reduce its error.

7.4.3 Process Time Consumption

The adaptation process speed is compared with that of the 47M in table 7.9.

Case	Minutes	Seconds
15 Mesh Movement Adapt.	145	51
2 Adjoint Adapt.	6150	13
Complete Adapt.	6295	4
47M Grid	7233	1

TABLE 7.9: Raw *CPU* timings for the combined feature and adjoint adaptation process.

It is clear that there is a gain in time, but unfortunately, this is not as significant as hoped. This is primarily due to the adjoint adaptation process. In fact, as may be calculated from the table, the feature based mesh movement requires only 2.31% of the overall adaptation time to run. One of the issues with the adjoint-based part relates to the perfect convergence that ought to be achieved to compute the errors accurately: in fact, the consequent interpolation to an embedded grid will amplify the flow and adjoint

residuals and thus hinder the calculation of the correction factor. This is not the case for the mesh movement iterations, as the best convergence is only required in the last step. A second issue that appears only in the adjoint-part of the procedure, relates to the refinement reducing the convergence slope. This problem is even more accentuated in the final grid, having been modified by 2 refinement steps. As was mentioned in chapter 4 when discussing the flux reconstruction, [149] states that *centred schemes*, such as that implemented in *Hydra*, can have extra stability problems when employed with hybrid grids generated by the refinement. Therefore, the reduced convergence rate can be expected. It should be noted that stopping the process after a single adjoint-adaptation step would reduce time consumption by 64%. The main issue that slows down the procedure, relates to the adjoint code not storing the flow solution's *Jacobian* matrix, that ought to be computed at every step, and the application of *AD* software, that does not produce an optimised code, thus causing a significant time increase in the process.

To decrease the wall clock requirements of the overall adaptation, tests would be required to determine what is the maximum convergence level needed to reliably compute the adjoint errors. In fact, if the *computable correction* were not required, it is possible that higher residual levels would not affect the *non-computable correction* evaluation, as for this to be employed in a refinement algorithm, only the relative weight of each cell in the overall value should be sufficient to determine whether to add nodes or not. Finally, there would also be the possibility of choosing the *Generalised Minimum Residual* (*GMRES*) adjoint solver that is available in *Hydra*, as this could help reduce the time consumption. A final comment that should be made concerns achieving the solution on the 47M grid: apart from the mesh generation, case pre-processing and *CPU* time consumption, it is very time consuming to determine which flow solver settings manage to achieve reliable convergence. In fact, in general it is more difficult to minimise the residuals with finer grids as any unsteadiness in the flow will start to affect the convergence. Added onto these issue is the amount of memory required to store the data and time needed for the post-processing. Moreover, unlike the adaptation process, the values presented in table 7.9 do not consider the time required to achieve the 47M grid adjoint solution.

7.5 Conclusions

Summarising, the first adjoint-based mesh adaptation process has been applied to turbomachinery test-cases. It has been shown that by employing a repeated, feature-based, mesh movement strategy, the grid significantly improved the alignment, thus yielding a better estimate of the *computable* and *non-computable corrections*. While other feature and adjoint combined procedures employed them simultaneously ([19]), in this case they

are used in a sequential manner. Therefore, before any adjoint solver simulation, the grid has already been improved to better capture flow features that will affect the adjoint error. Two refinement steps based on the *non-computable* error were performed. These produced excellent results for what concerns the adiabatic efficiency evaluated between inlet and outlet. In fact, the node enrichment process was able to reduce the nonlinearities affecting the functional, therefore allowing the *computable* counterpart to produce a much improved estimate. For the absolute total pressure ratio between inlet and outlet and the mass flow evaluated at the exit plane, significant benefits were obtained in the first adaptation step with a decrease in error. Unfortunately, in the following one, the presence of hybrid elements closing the refinement patches proved to be too problematic. In fact, despite reducing the error previously present, new sources appeared caused by the diverse types of cells. As a remedy to the issue, and have the ability to reliably reduce the overall sources of inaccuracies, the author recommends refinement techniques that will remove any hybrid cell marked to be split, and replace it with a refined version of the original grid's hexahedra.

Chapter 8

Fully Anisotropic & Edgewise Adjoint Grid Adaptation

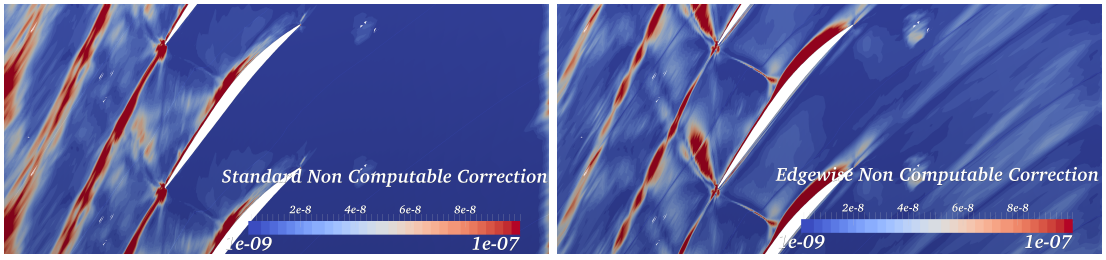
8.1 Introduction

Adjoint mesh adaptation techniques have been employed either on their own, or in a combination with feature based processes. In the latter case, this is to capture flow *anisotropy*, i.e. be able to cluster nodes in the flow direction, thus aiding adjoint error estimation, as shown in chapter 7. However, no technique to improve the grid to capture the adjoint *anisotropy* has ever been applied. In fact, if it were possible to modify the mesh to be able to align the grid edges with the regions of high sensitivity of the functional, the process would be more efficient. This would be feasible if there were a functional-related quantity to be adopted to determine a *Riemannian* metric that could be employed in the mesh movement process outlined in section 6.5.3. Moreover, having an edge-based error estimation for the refinement, would allow to add to the effectiveness of the overall procedure.

Such techniques have been devised in this work, and will be applied to a typical fan case with a splitter found on a modern jet engine and *NASA Rotor* 37. It will be shown how the mesh relocation employing the mesh adjoint quantity is capable of clustering and aligning the grid not only towards the important flow features for the functional of interest, but also to its high sensitivity regions. Comparison with feature-based mesh movement will prove the novel technique's benefit, in terms of functional estimation and convergence properties.

With regards to the refinement, the first edge-based grid enrichment strategy, was designed to completely remove the need to create an embedded grid and thus any extra

flow/adjoint run that may be required, as described in section 6.3.2. This allows a simpler process, not only more efficient in terms of time consumption, but also achieves a huge reduction in memory usage. In the *non-computable correction* term, differences between the quadratically and linearly prolonged flow and adjoint solutions are needed. This requires 4 different quantities to be evaluated and then stored. These are substituted by the subtraction of interpolated quantities along a coarse mesh edge, thus avoiding the need to create a fine grid. Moreover, if the error is computed by the adaptation software as in this case, they do not need to be stored. In the sensor devised by [64], the embedded mesh flow and adjoint residuals are required for the linearly interpolated quantities. Thus, extra runs of the respective solvers are needed, along with storage of the nodal residuals. To avoid this, the fine grid *NS* and adjoint imbalances were substituted by those along each edge of the coarse mesh. Although this procedure does not allow to accurately calculate the *non-computable correction*, it does produce a smoother error map that mimics its behaviour on the embedded mesh employing the prolonged quantities. Additionally, the error will be fully edge-based. Therefore the requirement of an efficient refinement process has been achieved.



(A) Standard calculation. (B) Novel edgewise calculation.

FIGURE 8.1: Mass averaged absolute total pressure ratio *non-computable correction* evaluated at mid-span of *NASA Rotor* 37: comparison of standard and new techniques.

A comparison between the *non-computable correction* calculated as in chapter 7 with that using the modification just outlined, is reported in figure 8.1. As it can be seen, very similar regions are highlighted by the two techniques, the main difference being the intensity with which they are flagged. In fact, the standard calculation of figure 8.1a, shows a strong weighting of the adjoint. All the upstream, suction side and *reversed wake* are clearly the strongest sources of inaccuracies. On the other hand, the shock propagation, along with the wake have a lower level of error. In chapter 7, it was shown that, as the adaptation progressed, the refinement mainly targeted the upstream block. On the other hand, this new formulation of the error, does not have the same rigorous mathematical formulation. However, it is clear that it highlights the same regions as error sources, but more importantly, it manages to weigh equally the flow and adjoint quantities. This can be helpful in improving the adaptation process, as the refinement will not only add nodes in the upstream regions.

Concerning the second grid-enrichment technique, this is based on a simple adjustment employing the *non-computable error* devised by [64], and described in section 6.3.1. Firstly, it was noted that once the flow and adjoint quantities had been interpolated to an embedded grid to evaluate the error, this is also calculated for each coarse mesh edge. In fact, in the fine grid, there is a node splitting each edge of the parent mesh. Employing this quantity to determine whether to refine or not the coarse mesh edge, allows to achieve a fully edge-based adaptation algorithm. Another benefit resulting from this modification, is the elimination of error restriction from the embedded to the parent grid.

8.2 Combined Adjoint-Based Movement & Refinement

Just as in chapter 7, the adaptation process will start with mesh movement. However, in this case the node relocation will be driven by the functional sensitivity to grid coordinates. This will be repeated until the functional of interest has converged. This first part will allow grid clustering in regions where the performance quantity is most affected. The second part will consist of repeated mesh refinement. Once the grid has been aligned with the features affecting the functional, adjoint-error calculations will be more accurate, thus allowing a more targeted and efficient refinement. Two different processes will be run: the first will add nodes based on the *non-computable correction* as determined employing the edge residuals on the coarse mesh. The second procedure will use the original adjoint-error calculation on the embedded grid, but will mark the coarse mesh edges, rather than the entire cell. In both cases, the refinement will be applied to the grid achieved through repeated mesh movement based on the functional sensitivity to the node coordinates. The overall approach is summarised in the flow chart of figure 8.2.

In this work, as no geometry reconstruction was carried out, the refinement would have to be limited. Moreover, given the experience accumulated with the results outlined in chapter 7, it was decided to apply a single refinement step to minimise possible issues caused by hybrid elements appearing. In fact, these could partially mask benefits of the novel approaches.

A final mention should be made concerning the refinement settings. In fact, as in chapter 7, the number of nodes added was approximately half that of the starting grid. This also included the wall normal refinement propagation, that was set to 10 layers.

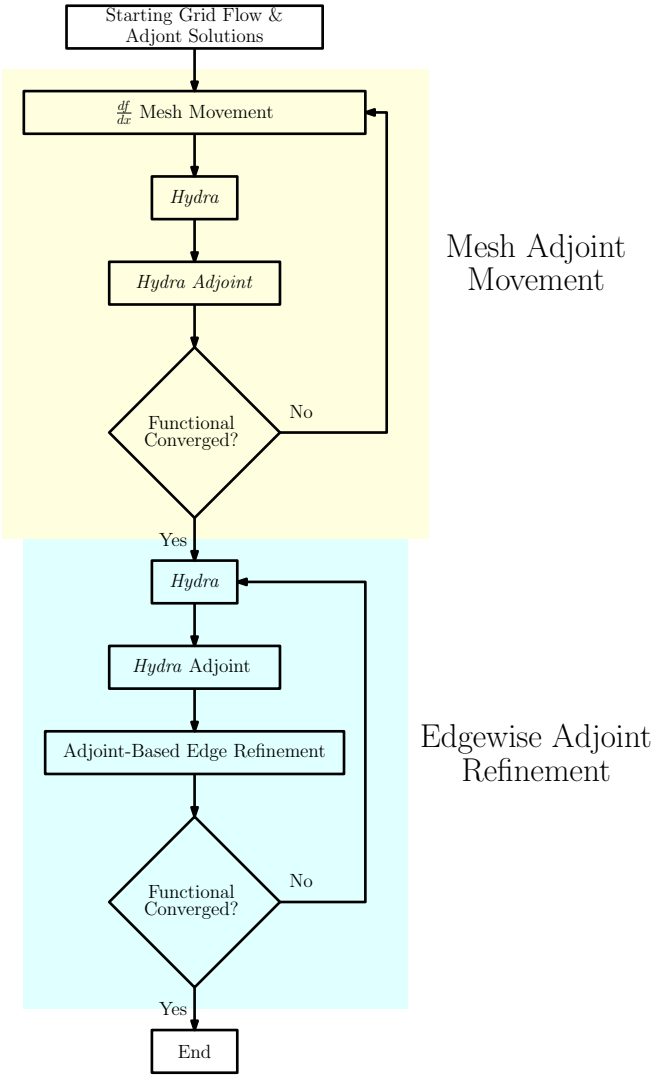


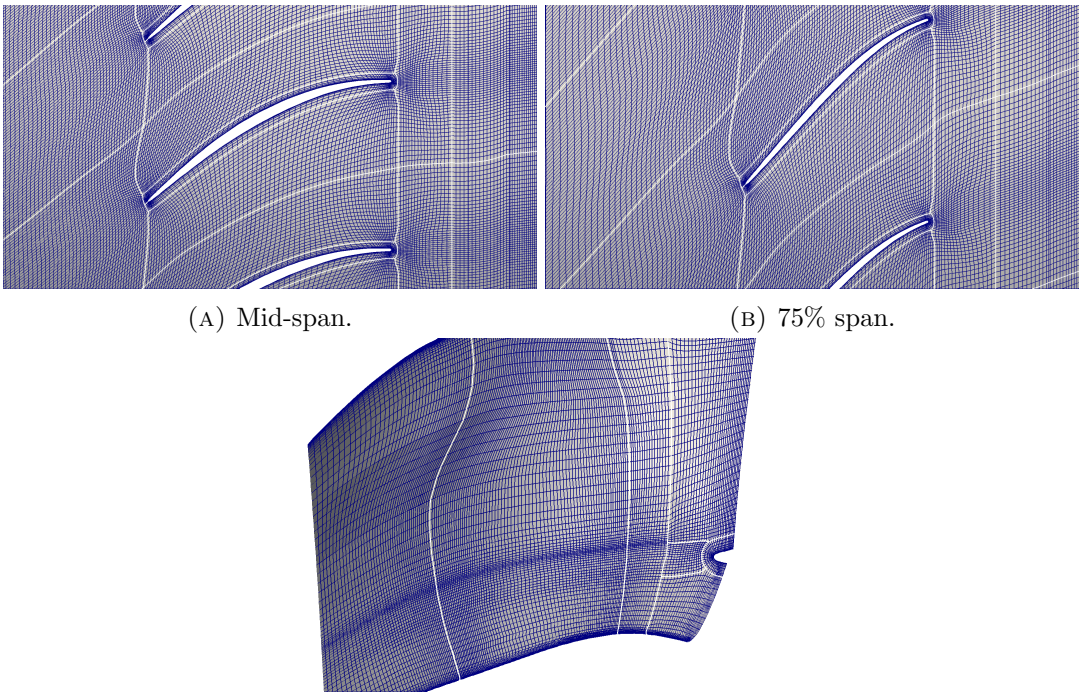
FIGURE 8.2: *Anisotropic* edgewise adjoint adaptation flow chart.

8.3 Fan Blade

8.3.1 Physical Aspects & Computational Setup

Close to the operating point, the fan blade of interest proved to produce a very stable flow with limited features. At choke, on the other hand, more intricate behaviour was observed. For this reason, it was decided to run the flow and adjoint solver at these conditions. As in the case of *NASA Rotor* 37, the target performance was provided by a very fine mesh of approximately 87M nodes achieved by refining two times in every direction the starting grid of 1.3M nodes. Extra points were added in-between the block boundaries to provide satisfactory grid quality. As for the *NASA Rotor* 37 test-case of the previous chapter, the starting and ultra fine grids were generated by way of *PADRAM*, described in section 3.2. These proved to be high quality structured

multi-block meshes. An example of the coarse 1.3M mesh at mid-span is reported in figure 8.3a. The inter-block boundaries have been highlighted with white lines. In figures 8.3a and 8.3b a similar block structure as for the *NASA Rotor* 37 can be seen, i.e. *O-mesh* around the blade, two passage blocks split between pressure and suction side and inlet/outlet components. The only difference is the downstream block divided into two. The periodic cut on the pressure side in figure 8.3c shows the block composition around the splitter.



(A) Mid-span. (B) 75% span.

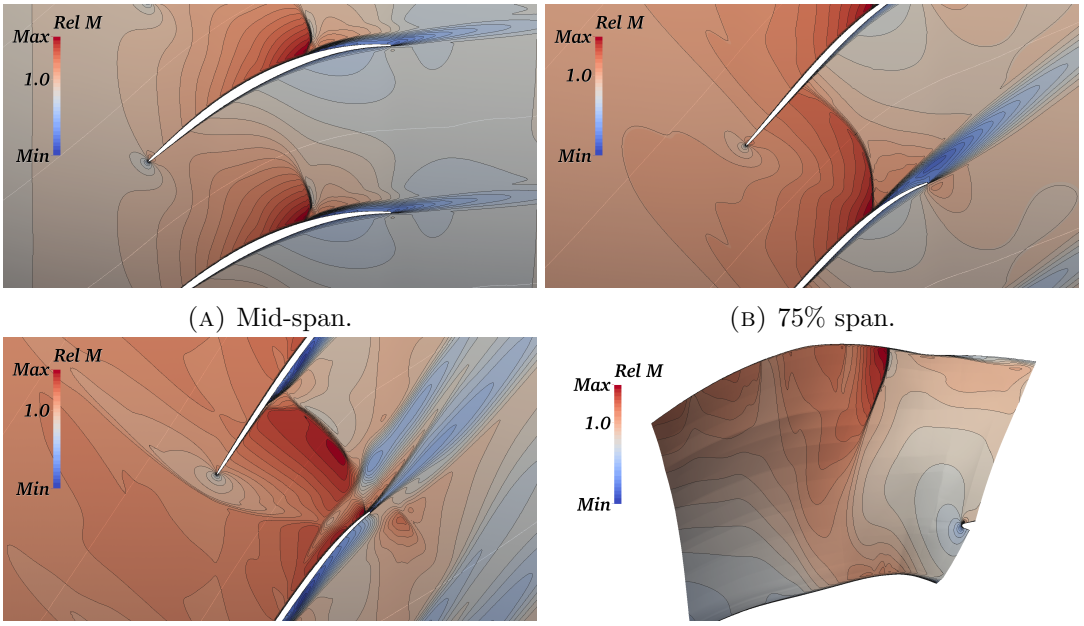
(C) Periodic boundary.

FIGURE 8.3: Fan blade 1.3M starting grid.

The described flow will be relative to the 87M grid solution. As it can be seen from the domain cuts at 50, 75 and 99% span (figure 8.4), there is a strong shock forming just before the blade *TE*. This causes a small degree of separation to occur, as shown by the wake in figure 8.4a. Moving along the blade height, it also increases in strength, until it hits the pressure side of the adjacent blade and due to the adverse pressure gradient, separation occurs. This, then propagates over the blade tip onto the suction-side where it interacts with the root of the shock (see figure 8.4c). The discontinuity intensity change can also be noted along the periodic boundary in figure 8.4d. To be noted is the limited flow complexity near the splitter.

Concerning inlet, bypass and engine outlet, these were set to subsonic inflow and radial equilibrium subsonic outflow, respectively. To enforce choke conditions, the bypass exit boundary setting was changed to a static pressure outlet. In the absolute frame of reference, the blade and upstream part of the hub were rotating with the same speed. The splitter was static, as were the casing and the downstream hub section.

For the adjoint mesh adaptation the quantity of interest was the mass averaged *adiabatic efficiency* evaluated between inlet and bypass (η_{BP}) and inlet and engine outlet (η_{EN}).



(A) Mid-span. (B) 75% span.

(C) 99% span. (D) Periodic boundary.

FIGURE 8.4: Fan blade relative *Mach* number distribution on the 87M grid.

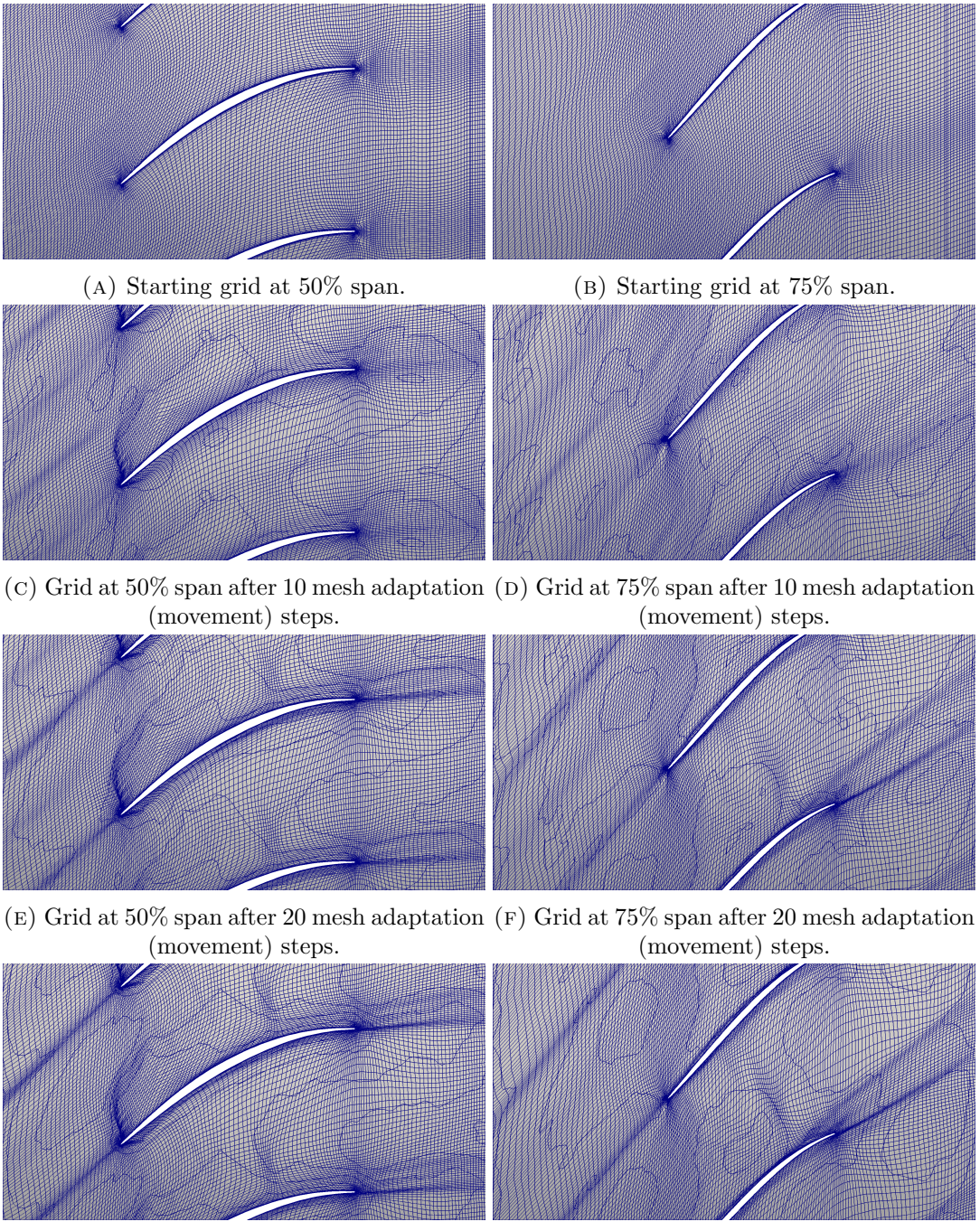
8.3.2 Part I: Adjoint Riemannian Metric Mesh Adaptation

To start with, only the $\frac{df}{dx}$ mesh adjoint quantity will be repeatedly employed to adapt the mesh for two quantities mentioned in the previous section. This will be used to compute a *Riemannian* metric indicating the *anisotropy* of the functional sensitivity. This will be employed to drive the mesh movement algorithm described in section 6.5.3. The overall process will be repeated until the functional estimation stalls to a fixed value. To be able to show the mesh adjoint $\frac{df}{dx}$-based node movement validity, this will be compared with the results obtained by a relative *Mach* number mesh relocation strategy (described in appendix D).

8.3.2.1 Adiabatic Efficiency between Inlet & Bypass Exit (Mass Averaged)

One of the main quantities of interest to a fan designer, is the adiabatic efficiency evaluated between inlet and bypass outlet. Comparing the target performance and that of the starting, 1.3M grid, at choke, it was seen that the latter over-estimated it by over half a percent. Therefore, it would be interesting to see if applying the procedure described in section 6.3.3 along with the spring stiffness mesh movement would improve the functional estimation (no refinement being carried out at this point). The overall

number of adaptation steps was 45, with highlights of the evolving grid shown at mid- and 75% span reported in figure 8.5. The magnitude of the mesh adjoint $\frac{df}{dx}$ is shown in figure 8.6.



(A) Starting grid at 50% span.

(B) Starting grid at 75% span.

(C) Grid at 50% span after 10 mesh adaptation (movement) steps.

(D) Grid at 75% span after 10 mesh adaptation (movement) steps.

(E) Grid at 50% span after 20 mesh adaptation (movement) steps.

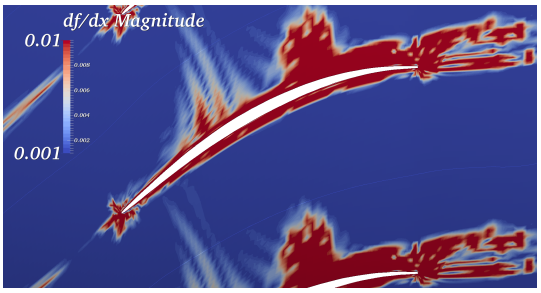
(F) Grid at 75% span after 20 mesh adaptation (movement) steps.

(G) Grid at 50% span after 30 mesh adaptation (movement) steps.

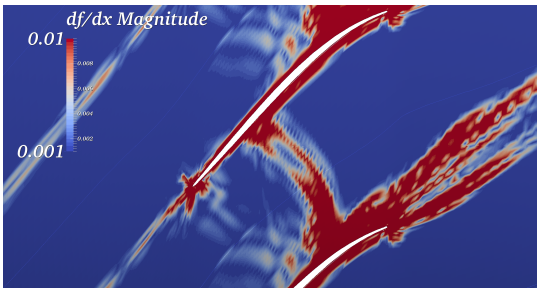
(H) Grid at 75% span after 30 mesh adaptation (movement) steps.

FIGURE 8.5: Evolution of the grid throughout the mesh adaptation process for η_{BP}.

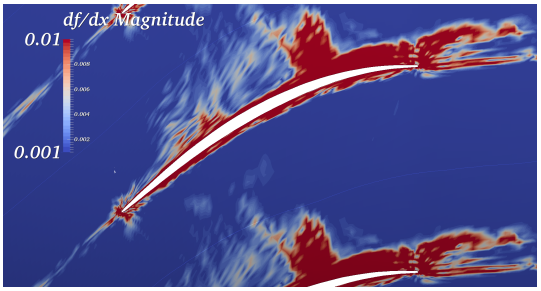
By comparing the latter with the flow features seen in figure 8.4, it is clear that the shocks and flow wake are high sensitivity regions for the functional. However, as in the case of *NASA Rotor* 37, a *reversed wake* of relatively high magnitude forms in the



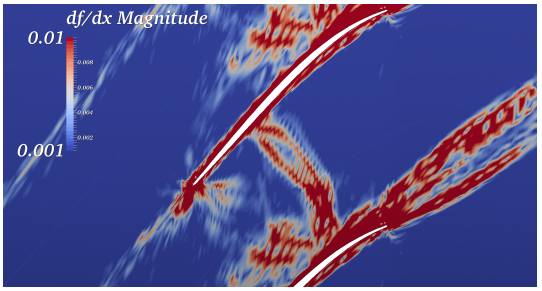
(A) Starting grid mid-span.



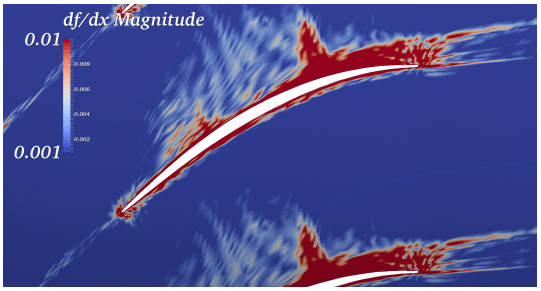
(B) Starting grid 75% span.



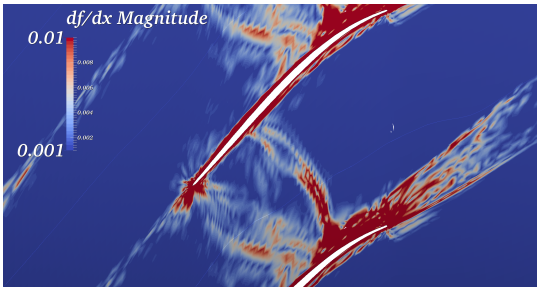
(C) After 10 mesh adaptation/movement steps at mid-span.



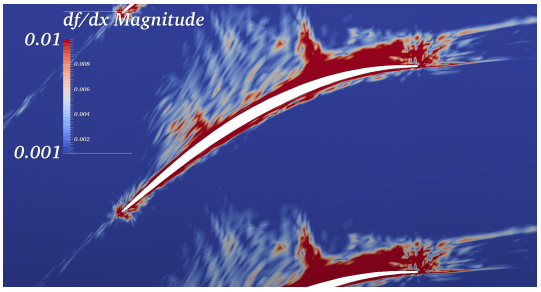
(D) After 10 mesh adaptation/movement steps at 75% span.



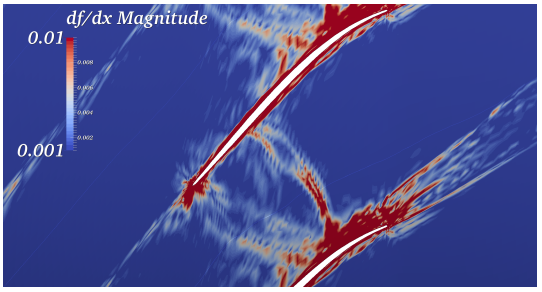
(E) After 20 mesh adaptation/movement steps at mid-span.



(F) After 20 mesh adaptation/movement steps at 75% span.



(G) After 30 mesh adaptation/movement steps at mid-span.

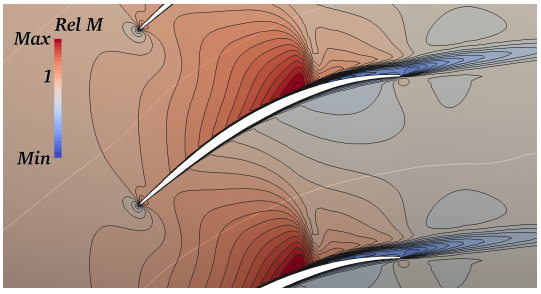


(H) After 30 mesh adaptation/movement steps at 75% span.

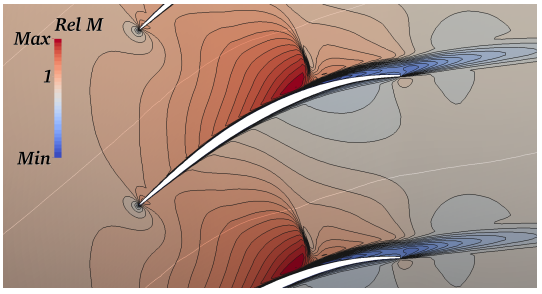
FIGURE 8.6: Evolution of the mesh adjoint $\frac{df}{dx}$ magnitude throughout the mesh adaptation process for η_{BP}.

upstream block. As the shock propagation is bounded within the passage, the upstream sensitivity is far less *w.r.t.* the compressor case in chapter 7. This allows a simpler mesh adaptation procedure.

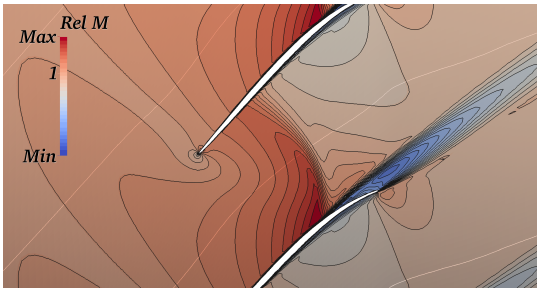
As it can be seen from the grid evolution, as the adaptation steps progress, the clustering and alignment improve with the sensitivity features, regardless of whether they appear in the flow solution or not. As one would expect, after a while the mesh movement starts to reduce its influence. For this reason no grid is shown after the 30^{th} iteration step. However, by looking at figures 8.5g and 8.5h, both at mid- and 75% span, the flow wake and the shock locations are clearly visible in the grid lines. This is also the case for the dual *reversed wake*. It is interesting to note that at mid-span, a strong curvature of the grid lines occurs on the blade *LE* on the suction side. By analysing the functional sensitivity in this region, it is clear that there is a relatively large difference between the passage values and those immediately upstream (the legend's threshold is too high to see it). For this reason, the arc at the *LE* has materialised. As the mesh adaptation progresses, it can be seen that the sensitivity maps change. In fact, while the shock region appeared to be quite thick for the starting mesh, at the end it significantly reduces in size. Also, the wake is not entirely highlighted as a region of high sensitivity, only its interface with the bulk of the flow. The evolution of $\frac{df}{dx}$ is not only important to achieve an improved grid for the functional of interest, but it is a crucial parameter in the discrete adjoint optimisation process. Therefore achieving a more accurate estimation of this quantity can play a significant role in appropriately improving the test-case performance.



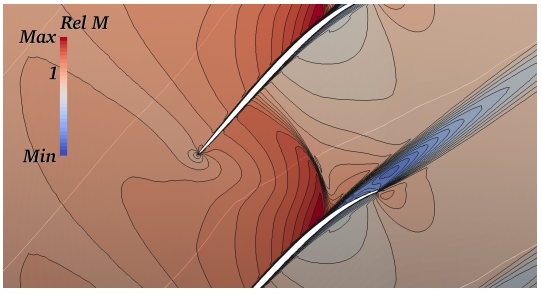
(A) Relative *Mach* number at mid-span for the starting 1.3M mesh.



(B) Relative *Mach* number at mid-span after 30 mesh adaptation/movement steps.



(C) Relative *Mach* number at 75% span for the starting 1.3M mesh.



(D) Relative *Mach* number at 75% span after 30 mesh adaptation/movement steps.

FIGURE 8.7: Comparison of the starting mesh flow with that obtained after 30 mesh adaptation steps for η_{BP}.

Figure 8.7 shows how the flow resolution has changed between the starting grid and the

one achieved through 30 mesh adaptation/movement steps. While the wake has had minimal change, the shocks mid- and 75% span are clearly better resolved. The reason for the minimal change in the downstream wake is caused by the initial mesh appropriate alignment in this region. It should be noted that no nodal addition has been carried out, therefore the final adapted grid still has the same node count as the starting coarse mesh. Nevertheless, by carefully analysing the 87M node mesh in figure 8.4, there are still some differences. In fact, at both span-wise locations reproduced in images 8.7b and 8.7d, the root of the shock on the suction-side is still not appropriately captured. Moreover, at 75% span, the discontinuity is still, qualitatively speaking, quite thick.
It is well known that the main attraction towards adjoint-related mesh adaptation strategies is their capability to minimise errors in the grid related to the functional of interest. Therefore, for this case it is essential to be able to show improvement of the η_{BP}, as otherwise it will loose any real significance as a grid modification strategy. To this end, figure 8.8 has been included, showing the variation of absolute difference between the adapted mesh and the target value achieved with the 87M grid.

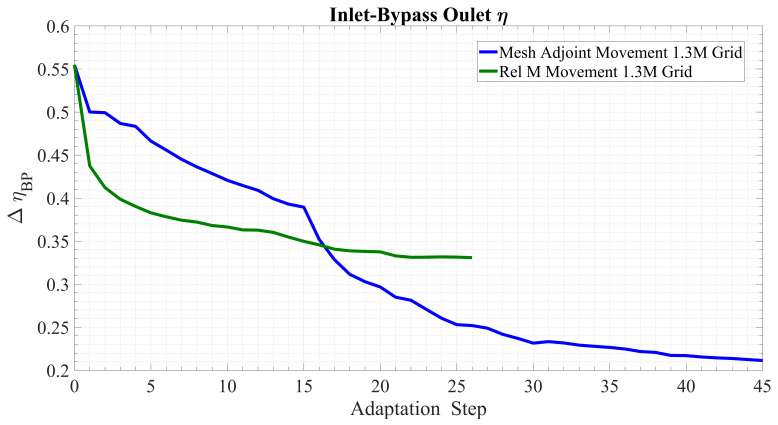


FIGURE 8.8: Variation of the difference of η_{BP} between adapted and target grids during the adaptation process.

Clearly, the mesh adjoint node movement outperformed the feature based approach[1] in terms of estimation of the functional and number of adaptation steps. In fact, the relative *Mach* number related process, stopped converging to pre-defined user level after the 26^{th} adaptation step. Moreover, the flow solver settings required changing throughout the process. On the other hand, the use of $\frac{df}{dx}$ calculated by the mesh adjoint is far more robust. In fact, no variation to the solver settings were required for either the flow or adjoint codes. Furthermore, it reached the 45^{th} iteration converging to the same level, and would have been able to continue, but the very small improvement in the functional negated any real benefit. The reason behind this robustness is believed to be related to the formulation of the mesh adjoint relation. By looking at equation 5.70, it is possible

[1]The reader is referred to appendix D for an overview of the resulting grid and flow obtained using this process.

to see that there the $\frac{\partial R}{\partial x}$ term is present. This, multiplied by the flow adjoint produces a contribution to $\frac{df}{dx}$, that is then outputted as the overall mesh adjoint quantity and used in the adaptation procedure. Despite being a linearisation of the flux functions *w.r.t.* the mesh coordinates, it still has the effect of including the residual behaviour in the overall sensitivity. Moreover, as discussed by [96], adjoint-based mesh adaptation techniques actually include the effect of the flux *Jacobian*-matrix, and therefore contain system stability information.

In figure 8.8, the reader will have noted that the feature process was performing better *w.r.t.* its adjoint counterpart, then around the 15^{th} step, things changed with the dual process rapidly reducing the functional estimation towards the target performance. During various tests conducted by the author, it was found that sometimes the $\frac{df}{dx}$ quantity can be slightly noisy in regions of very low sensitivity. The *Hessian* matrix calculation procedure relies on a GG operator applied to the mesh adjoint output. During this process, it is possible for the noise to be amplified, and therefore cause a degree of erratic movement in the grid. For this reason, in the process employed above, edge-length weighted *Laplacian* smoothing was applied to the sensitivity magnitude to even out the overall field. This was done for the first 15 adaptation steps causing a visible reduction of the rate of change of the efficiency. However, in this case, it was found that after this number of iterations, the smoothing was not required anymore. This is the reason behind the abrupt change in the convergence history of the objective.

8.3.2.2 Adiabatic Efficiency between Inlet & Engine Outlet (Mass Averaged)

In parallel to the adaptation process for mass averaged adiabatic efficiency calculated between inlet and bypass exit, the same value across inlet to engine outlet was also applied in a separate process. This to be able to see if changing part of the integration surface would cause any differences. Moreover, it was found that while the coarse mesh was over-estimating η_{BP} by roughly half a percent point, η_{EN} was underestimated by a fifth of percent point. Therefore, it was interesting to see if the mesh movement employing η_{EN}'s mesh adjoint would be able to increase the value and, given the limited difference, stall in the vicinity of it.

Figure 8.9 shows how the functional absolute difference between adapted and fine grids varies using the feature-based approach and the mesh adjoint technique. Clearly, the latter is not only able to correctly increase the value, but stalls by approximately 0.025% relative to the result of the much finer grid, thus indicating the procedure reliability. Moreover, it outmatches the relative *Mach* number mesh movement, as this tends to decrease the functional as a result of the adaptation. Given that the main sources of

error for the feature-technique are in the upper half of the span, the regions causing inaccuracies at lower radius have not been appropriately targeted. This is where η_{EN} was seen to have the highest sensitivity *w.r.t.* the grid coordinates. To check the validity of this assumption, the *non-computable* error as in eq. 6.26, was computed. The maximum value of this over the entire domain of the starting grid is shown in figure 8.10.

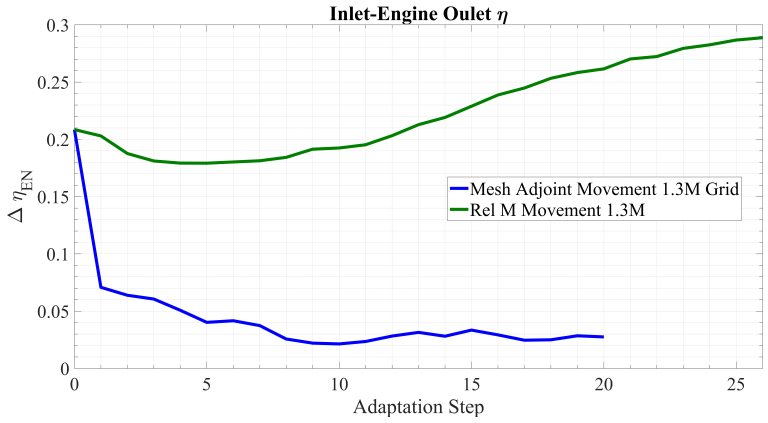


FIGURE 8.9: Variation of the absolute difference of η_{EN} between adapted and 87M grids during the adaptation process.

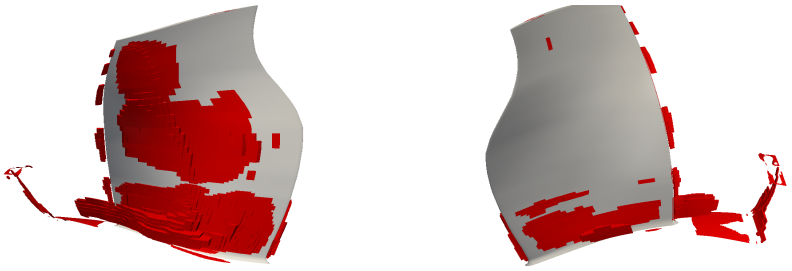


FIGURE 8.10: Maximum *non-computable correction* for η_{EN} over entire domain of the starting mesh (suction-side *LHS*, pressure-side *RHS*).

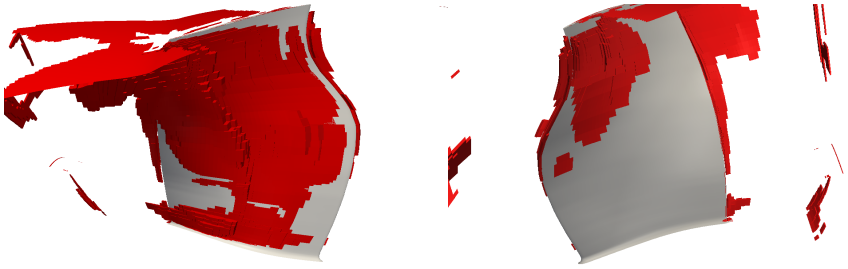


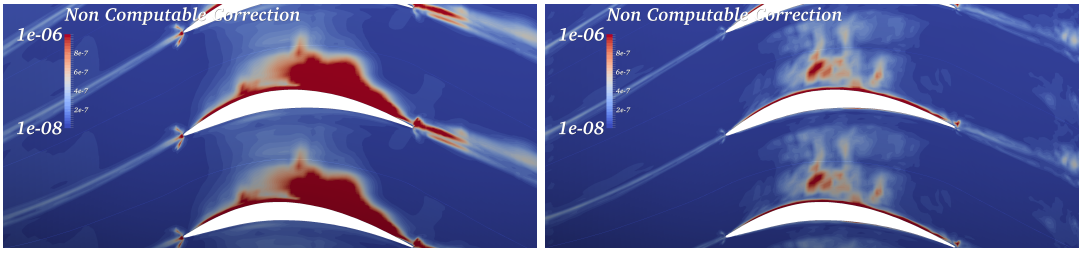
FIGURE 8.11: Maximum *non-computable correction* for η_{BP} over entire domain of the starting mesh (suction-side *LHS*, pressure-side *RHS*).

Clearly, the functional is most sensitive around the blade surface, particularly on the suction-side, where the estimated shock location is[2]. Additional high sensitivity regions are the wake, but mainly the lower span region. The latter is to be expected, as the

[2]The word *estimated* is used as the starting mesh does not have an accurate resolution of the shock.

performance quantity integration surface is below the splitter, at the outflow towards the engine. Comparing the error map with that of the η_{BP} for the starting mesh (figure 8.11), it can clearly be seen that the two are almost complementary, with only a minimal superposition of high inaccuracy regions. Therefore, the feature-based approach is able to improve the value of η_{BP}, but not that of η_{EN}, due to the fact that flow features at higher span are much stronger and are better resolved. Additionally, these have a greater impact on the functional evaluated over the bypass exit rather than closer to the hub.

As η_{EN} appeared to stall, with very small oscillations, after the 10^{th} adaptation iteration, the *non-computable correction* (calculated with eq. 6.26) map at this step is compared to that of the starting grid in figure 8.12. As it can be seen, the error has decreased throughout the adaptation process. The magnitude has clearly been reduced on the blade suction side, but also at both *reversed* and flow wakes.



(A) *Non-computable correction* at 10% span for the starting 1.3M mesh.

(B) *Non-computable correction* at 10% span after 10 mesh adaptation/movement steps.

FIGURE 8.12: Comparison of the starting mesh η_{EN} *non-computable correction* (calculated as in chapter 7) with that obtained after 10 mesh adaptation steps for η_{EN}.

A final note concerns the smoothing. As previously mentioned, this was adopted for the η_{BP} case to smoothen out the sensitivity field for the first 15 adaptation iterations. This was the case for η_{EN} as well. Clearly, the smoothing has not impeded the process from sensibly improving the functional and relative *non-computable* error.

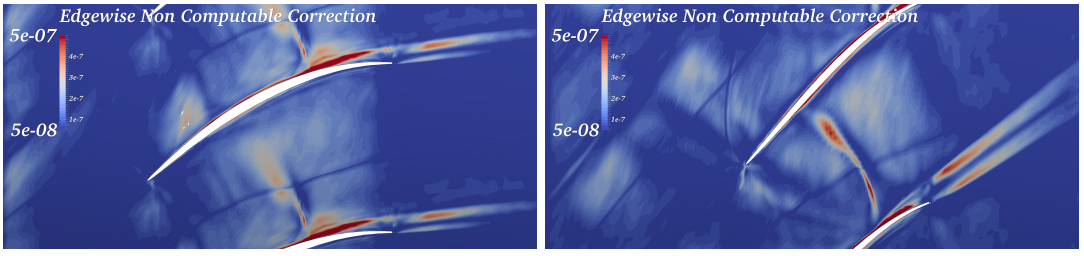
8.3.3 Part IIA: Efficient Edgewise Adjoint Mesh Refinement

8.3.3.1 Adiabatic Efficiency between Inlet & Bypass Exit (Mass Averaged)

It was shown in section 8.3.2.1 that the mesh movement based on the mesh adjoint quantity $\frac{df}{dx}$ significantly and reliably helped improve the estimate of the functional of interest. This was proved for the mass averaged *adiabatic efficiency* evaluated between the inlet and the two outlets (bypass and engine) separately. Concerning η_{EN}, the final result was close to the target performance of an 87M and therefore no refinement was considered as this would increase the cost for very little gain in accuracy. On

the other hand, for η_{BP}, the mesh movement was able to significantly reduce the gap between coarse and fine mesh, but the final difference was still approximately half that of the staring one. For this reason, it was decided to apply the refinement to the 30^{th} adaptation step mesh achieved in section 8.3.2.1. This was done utilising the modified adjoint *non-computable correction* devised in section 6.3.2. Once the flow and adjoint solutions had reached the desired level of convergence, the edgewise residuals/fluxes for each were appended to their respective files by the solvers. As for the *NASA Rotor* 37 case analysed in chapter 7, it was decided to add approximately half the starting mesh nodes.

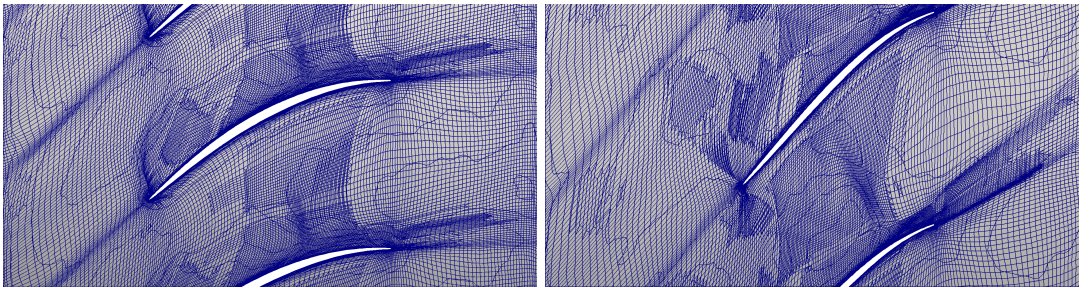
The resulting error employed for the refinement is displayed in figure 8.13 at mid and 75% span. The main sources of inaccuracies are bounded in the passage, with the only exception being the interfaces between the wake and the bulk of the flow in the downstream region. Interestingly, no upstream adjoint weighting is visible, possibly indicating that the mesh movement clustering towards the *reversed wake* has achieved the desired resolution. The highest error, is at the shock and the separation, just before the blade *TE*. At 75% span, due to the reduced movement towards the shock in the vicinity of the pressure side, a wider high error band is present.



(A) *Edgewise non-computable correction* at 50% span for the 30^{th} adapted mesh.

(B) *Edgewise non-computable correction* at 75% span for the 30^{th} adapted mesh.

FIGURE 8.13: η_{BP} *edgewise non-computable correction* after 30 mesh movement adaptation steps.



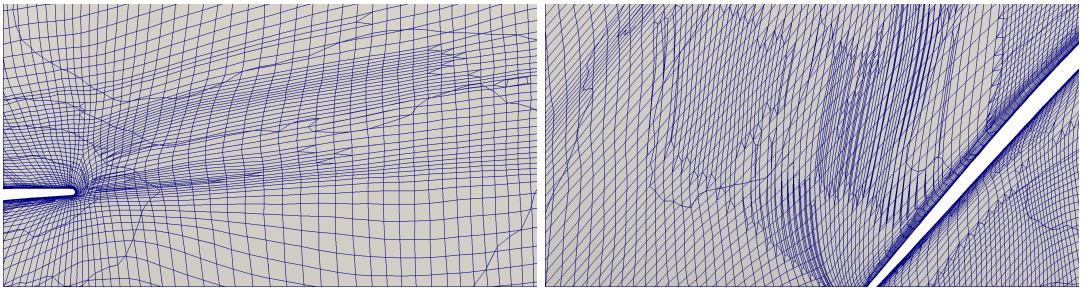
(A) Refined mesh at 50% span.

(B) Refined mesh at 75% span.

FIGURE 8.14: Resulting adapted grid after the *edgewise non-computable* error adaptation.

By applying the refinement according to the error maps just discussed, the grid in figure 8.14 was obtained. Due to the inaccuracies being spread out over the passage without

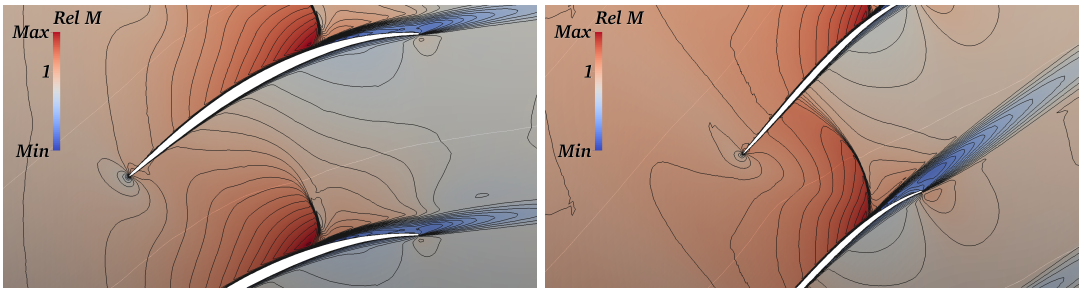
any real directionality, nodes are added in an *isotropic* manner. This is not the case at the wake. In fact, as its error is purely *anisotropic*, the procedure is able to detect this behaviour and only split edges in the normal direction to the wake itself, as shown in figure 8.15a. At 75% span, the refinement directionality is visible in the vicinity of the blade *LE* with propagation across the periodic boundary.



(A) Downstream wake refinement at 50% span.

(B) *LE*/periodic-boundary refinement at 75% span.

FIGURE 8.15: Resulting *anisotropy* after refinement using the *edgewise non-computable* error adaptation.



(A) 50% span.

(B) 75% span.

FIGURE 8.16: Relative *Mach* number after the *edgewise non-computable* error-based refinement.

To be able to see if the process has had any improvement on the flow feature resolution, figures 8.16a and 8.16b are provided. By comparing these with figures 8.7b and 8.7d, the only noticeable difference is the bunching up of the contour lines at the shocks.

Case	# Nodes	$\Delta\eta_{BP}$ (%)	$Log_{10}(\Delta\eta_{BP})$	$\Delta\eta_{EN}$ (%)
Starting Grid	1.3M	0.5548	−0.2559	0.2085
30^{th} move iter.	1.3M	0.2315	−0.6354	0.0931
Refined 30^{th} move iter.	1.9M	0.0589	−1.2300	0.2408

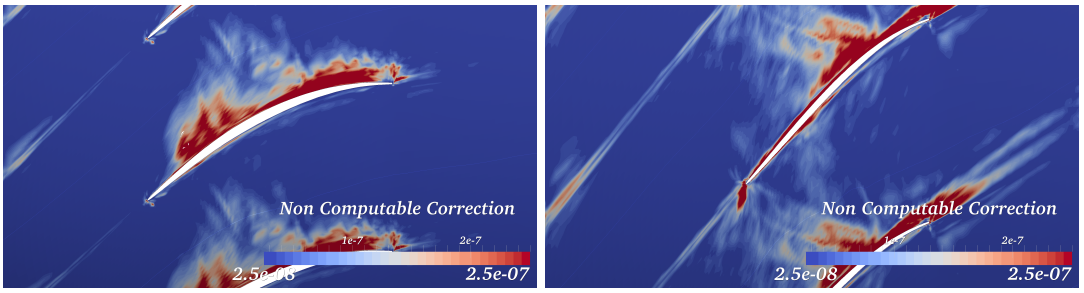
TABLE 8.1: Absolute difference between the adapted and target grids of mass averaged adiabatic efficiency evaluated between inlet and bypass outlet.

The main interest is the functional estimation. To this end table 8.1 is provided. Clearly, the refinement process and the overall combination with the functional-based mesh movement, have had the desired effect of reducing the difference between the starting mesh and a significantly finer one, by approximately 0.5%. The resulting grid could be employed in a blade optimisation strategy for η_{BP} and yield a more reliable final result.
In table 8.1 the change in efficiency evaluated over the engine outlet is also reported. After applying the η_{BP}-based mesh movement, this quantity has improved as well, albeit by a smaller amount *w.r.t.* that seen in section 8.3.2.2. As was shown in figures 8.10 and 8.11 the high error regions for these two values are almost complementary as they are integrated over different outlets. However, in the η_{BP} map, there is a lower degree of error just above the hub, around the blade surface, that was also highlighted by the mesh adjoint sensitivities. This causes a smaller amount of clustering when the mesh movement is employed, thus improving the accuracy of the other functional. On the other hand, table 8.1 shows the η_{BP} refinement process worsening the accuracy of η_{EN}. Again, this behaviour can be attributed to the smaller degree of error present at lower radius. In fact, the η_{BP}-based mesh adaptation will add a small refinement patch in this region. However, this is not sufficiently large to cover the high-error region of η_{EN}, meaning that hybrid cells closing the refinement will fall where the error was already high, further increasing its effect.

8.3.4 Part IIB: Embedded-Grid Edgewise Adjoint Mesh Refinement

8.3.4.1 Adiabatic Efficiency between Inlet & Bypass Exit (Mass Averaged)

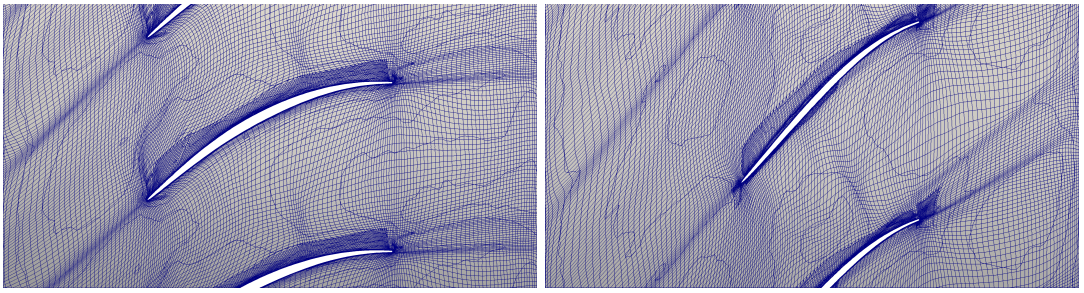
In this case, the standard *non-computable correction* was employed. However, the refinement process was modified by using the embedded mesh error to mark edges on the coarse grid. The resulting sensor map at 50% and 75% span is shown in figure 8.17. Comparing this with that calculated in the previous section, significant differences appear at mid-span. In fact, while both do highlight the blade suction-side, in the previous case flow features clearly stand out. On the other hand, the current error map, flags the near-wall region. At 75% span, both show strong inaccuracies at the shock root and wake, however, visible differences occur in the rest of the passage. It is thought that this behaviour results from the adjoint stronger weighting in the standard *non-computable correction* calculation. This is believed to be caused by the dual residuals on the embedded mesh being determined using the interpolated flow and adjoint solutions, rather than the exact ones.
Employing the error displayed in figures 8.17a and 8.17b to add approximately half the initial mesh nodes, resulted in the grids of figure 8.18. Unfortunately, due to the limited



(A) *Non-computable correction* at 50% span for the 30^{th} adapted mesh.

(B) *Non-computable correction* at 75% span for the 30^{th} adapted mesh.

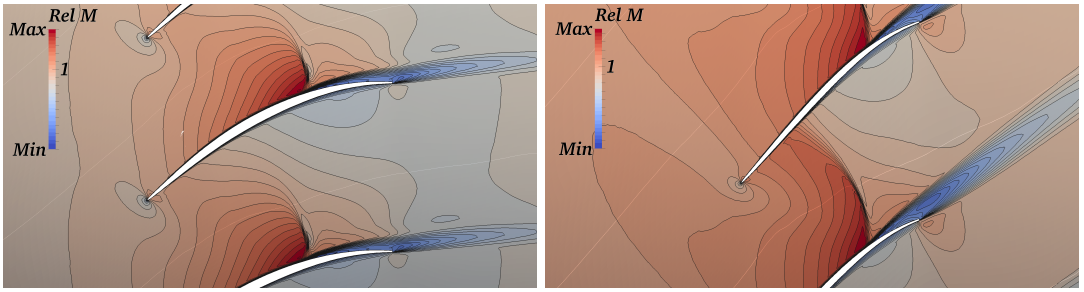
FIGURE 8.17: η_{BP} *non-computable correction* after 30 mesh movement adaptation steps.



(A) Refined mesh at 50% span.

(B) Refined mesh at 75% span.

FIGURE 8.18: Resulting adapted grid after the *non-computable* error adaptation.



(A) 50% span.

(B) 75% span.

FIGURE 8.19: Relative *Mach* number after the *non-computable* error-based refinement.

high inaccuracy regions, *anisotropy* can not be seen. In fact, all the adaptation process has done is to place a regular mesh block around the blade suction-side, covering it from *LE* to *TE* at both span locations. The resulting flow field at 50% and 75% span is reported in figures 8.19a and 8.19b, respectively. The refinement has only targeted the near wall region, therefore very little improvement may be seen at the suction-side shock root, unlike the previous technique that managed to improve the resolution throughout the passage. Nevertheless, the most important value is the mass averaged adiabatic efficiency evaluated between inlet and bypass exit, reported in table 8.2. The process shows excellent improvement, slightly outperforming the previous refinement approach resulting in a minimal difference with the 87M case. The better estimate of this process

w.r.t. the one employing the coarse mesh edgewise flow and dual residuals, is believed to be due to the extra adjoint weighting. Clearly, the improved resolution of the shock throughout its propagation is not necessary to be able to achieve an accurate estimate of the bypass efficiency.

Case	# Nodes	$\Delta\eta_{BP}(\%)$	$Log_{10}(\Delta\eta_{BP})$	$\Delta\eta_{EN}$ (%)
Starting Grid	1.3M	0.5548	-0.2559	0.2085
30^{th} move iter.	1.3M	0.2315	-0.6354	0.0931
Refined 30^{th} move iter.	1.9M	0.050	-1.3037	0.2408

TABLE 8.2: Absolute difference between the adapted and target grids of mass averaged adiabatic efficiency evaluated between inlet and bypass outlet.

A final note concerns the estimated value of η_{EN} on the refined mesh. As in section 8.3.3, the value is worsened by the node enrichment. This is due to the same reason, i.e. the refinement at lower radius being too little to cover the error region for this quantity as well. This causes cell-size variations and hybrid elements to appear where the error is already high, thus increasing the inaccuracies.

8.3.5 Process Time Consumption

The raw *CPU* time required for the process, including nonlinear, flow adjoint and mesh adjoint solvers, is reported in table 8.3.

Case	Minutes	Seconds
30 Adjoint Mesh Mov.	4589	21
1 Adjoint Ref.	277	21
Complete Adapt.	4866	42
87M Mesh	9489	10

TABLE 8.3: Fan blade η_{BP} full adaptation timings: 30 mesh movement and 1 refinement steps.

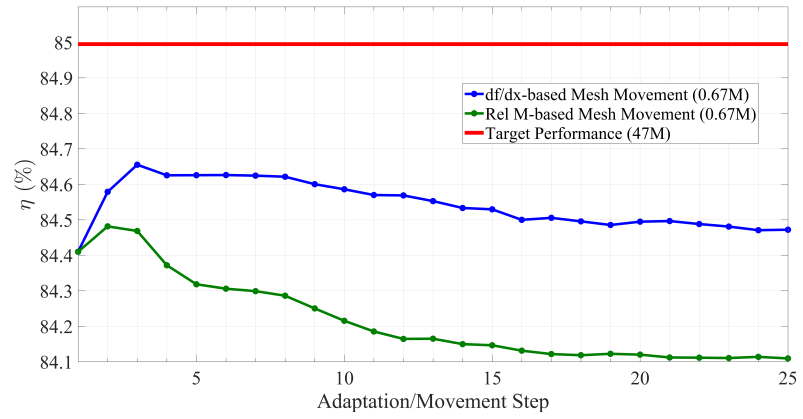
There is a clear advantage in terms of wall clock time *w.r.t.* the much finer mesh, as the adaptation process would require half the time. However, in this case the mesh movement process is much slower. This is due to the adjoint code: during the mesh movement process running the flow solver took an average of 6 minutes and 23 seconds,

the adjoint solution was achieved in 141 minutes and 2 seconds and the mesh adjoint required only 44 seconds. It should be noted that for the mesh movement process, perfect convergence of the solvers is only required for the last step. Therefore the overall mesh movement process was run reducing the convergence of the adjoint solver by two orders of magnitude: this change decreased the average time consumption by half an hour (110 minutes and 30 seconds).
As previously mentioned, the extra timing requirements of the adjoint could be reduced by employing the *GMRES* solver, or by relaxing its order of convergence requirements. Additionally, it may be possible to reduce the amount of mesh movement steps required by increasing the algorithm's relaxation factor ω or the amount of iterations of eq. 6.42.

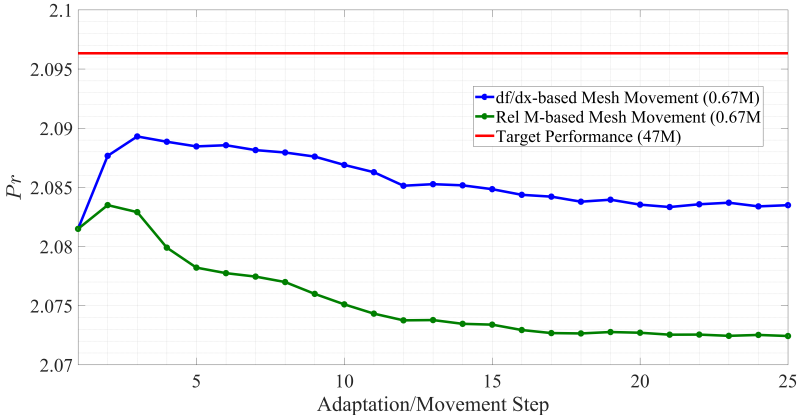
8.4 NASA Rotor 37

8.4.1 Part I: Adjoint Riemannian Metric Mesh Adaptation

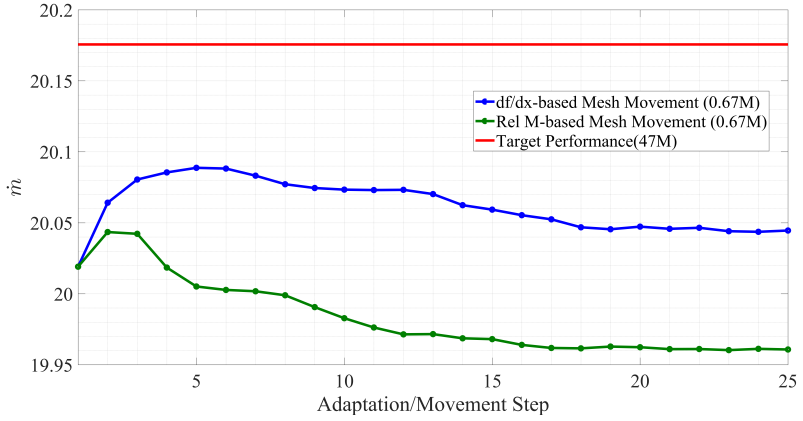
The mesh movement based on $\frac{df}{dx}$ alongside the edgewise-adjoint refinement was also applied to the compressor case employed in chapter 7. The same setup was utilised: 670k node starting grid run at aerodynamic design point with the same boundary conditions (see also section 7.3.1 for the test-case description and 7.3.2 for its setup). The performance quantities of interest were the same, i.e. adiabatic efficiency (η), absolute total pressure ratio (P_r), both mass averaged and evaluated between inlet and outlet, and area averaged mass flow ($\dot{m}$) integrated over the exit plane.
Unlike the fan case analysed in section 8.3.2, no smoothing appeared to be necessary for the functional sensitivity to the grid coordinates, as the erratic node movement in the low sensitivity regions was minimal.
Throughout the analysis of the previous chapter, it was noted that this mesh is not in *asymptotic range w.r.t.* the target grid performance. In particular it was shown that all functionals require good shock and upstream region resolution. Moreover, the mesh has poor quality inter-block corner points that cause numerical dissipation to appear and create fake flow features. Unlike the previous fan case, where the error of both performance quantities was consistently reduced by the mesh-adjoint based movement, for this grid, they started to improve and then slowly reduced in value. This is to be attributed to the starting mesh nature, just as in the case of the feature-based mesh movement in section 7.4.1.
The quantities of interest evolution over the three separate mesh-adjoint-based r-adaptation processes are shown in figure 8.20. As a comparison, the values obtained with the feature-based grid movement are also included. Clearly, there is a large difference in terms of



(A) η.



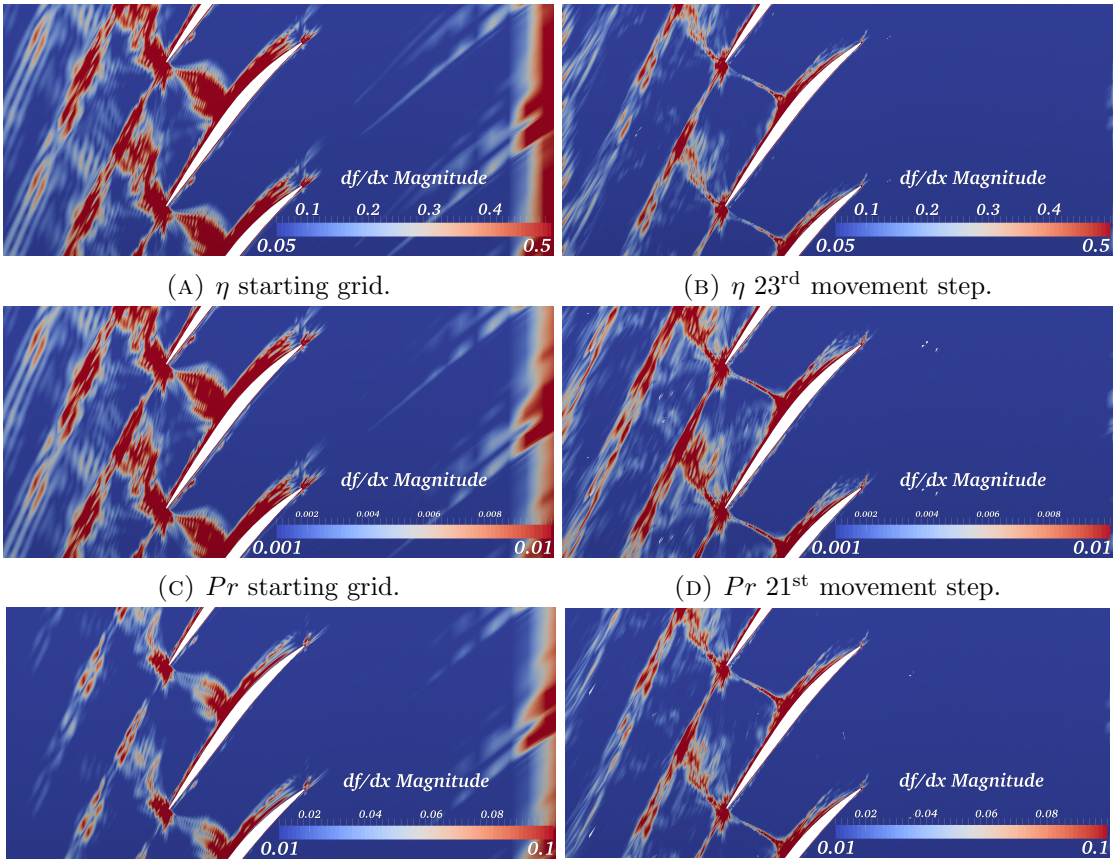
(B) Pr.



(C) $\dot{m}$.

FIGURE 8.20: Evolution of the performance quantities of interest throughout the mesh-adjoint-based node relation. Behaviour of the same functionals using the relative *Mach* number mesh movement of chapter 7 is also provided.

performance. The adjoint-related technique is again showing much better results. In fact, despite the starting grid issues, the final value is slightly closer to that of the target performance *w.r.t.* the hand-generated mesh. On the other hand, the relative *Mach* number approach almost doubles the error relative to the 47M mesh performance.



(A) η starting grid. (B) η 23$^{\text{rd}}$ movement step.

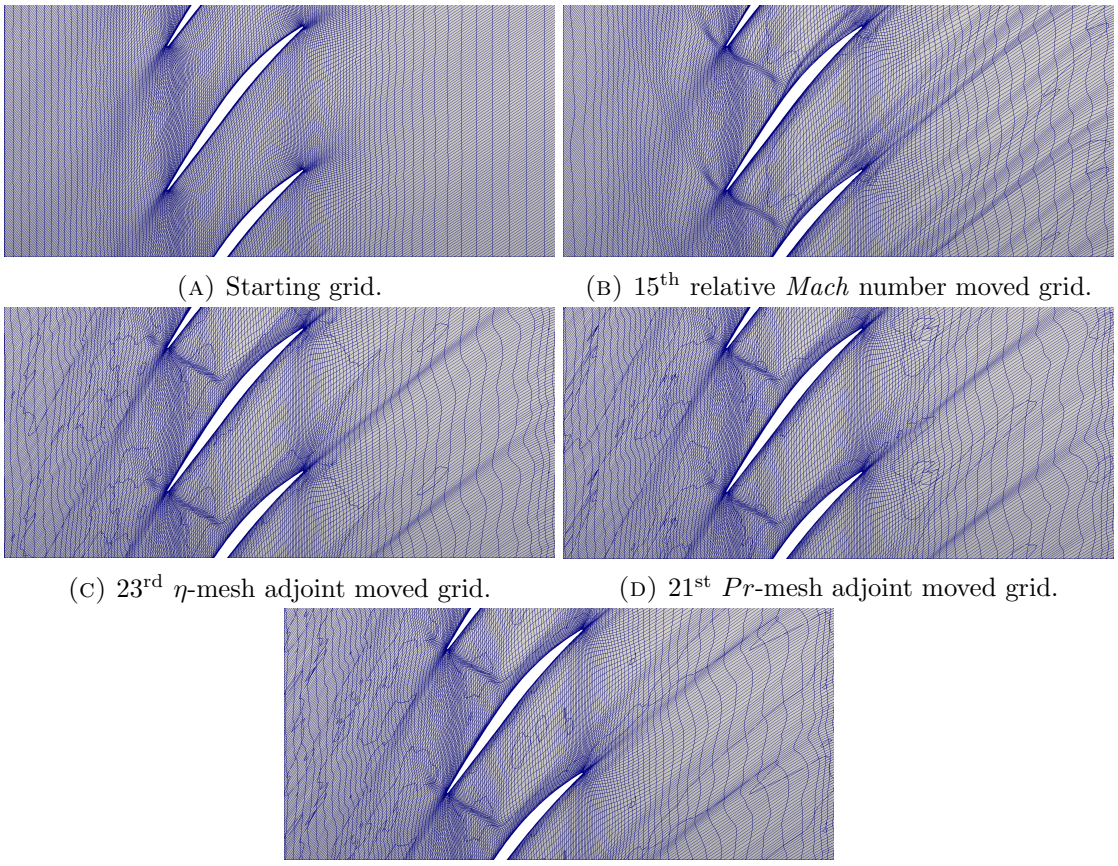
(C) Pr starting grid. (D) Pr 21$^{\text{st}}$ movement step.

(E) $\dot{m}$ starting grid. (F) $\dot{m}$ 17$^{\text{th}}$ movement step.

FIGURE 8.21: Functional sensitivity to grid coordinates at mid-span for the starting and adapted cases.

For all three quantities of interest, the process was repeated 25 times, meaning a total of 75 flow and adjoint simulations were required. Although, given that the nodal connectivity did not change, it was possible to employ the previous steps' solutions as a starting point for every adapted mesh, thus reducing time consumption.

The grid selected for each functional as the refinement starting point, was based on how the quantity behaved, rather than the number of adaptation iterations or being the first mesh with minimal node movement. In fact, repeatedly applying r-adaptation, can sometimes produce poor quality meshes, particularly in the boundary layer region. Moreover, consequent refinement may cause negative volumes. Concerning efficiency, the mesh resulting from the 23$^{\text{rd}}$ iteration was chosen, as its value was approximately the average of all iterations following the 15$^{\text{th}}$. On the other hand, for pressure-ratio, the 21$^{\text{st}}$ grid was selected, as beyond that point, it appeared to oscillate periodically. Finally for mass flow, the mesh movement only affected the 3$^{\text{rd}}$ decimal figure after the 17$^{\text{th}}$ step. By looking at the evolution of the various performance quantities in figures 8.20, it is clear that after the 15$^{\text{th}}$ iteration, the changes appear to be minimal. Therefore,



(A) Starting grid.

(B) 15^{th} relative *Mach* number moved grid.

(C) 23^{rd} η-mesh adjoint moved grid.

(D) 21^{st} Pr-mesh adjoint moved grid.

(E) 17^{th} $\dot{m}$-mesh adjoint moved grid.

FIGURE 8.22: Starting, feature and mesh adjoint-adapted grids at mid-span.

as for the feature-based approach, the grid resulting from this step could have been chosen as a starting point for refinement. In fact, looking at the mid-span sensitivity maps in figures 8.21b, 8.21d and 8.21f, neglecting the order of magnitude difference[3], they are indeed very similar. All of them flag the adjoint *reversed wake*, passage shock, suction side boundary layer and separation. Therefore it may be concluded that for these performance quantities, the grid resulting from the same mesh movement step may be employed. This is reinforced by the resulting grids similarity for each quantity of interest shown in figures 8.22c, 8.22d and 8.22e. However, it was decided to use meshes from different adaptation steps to see the effect this had on the consequent refinement.

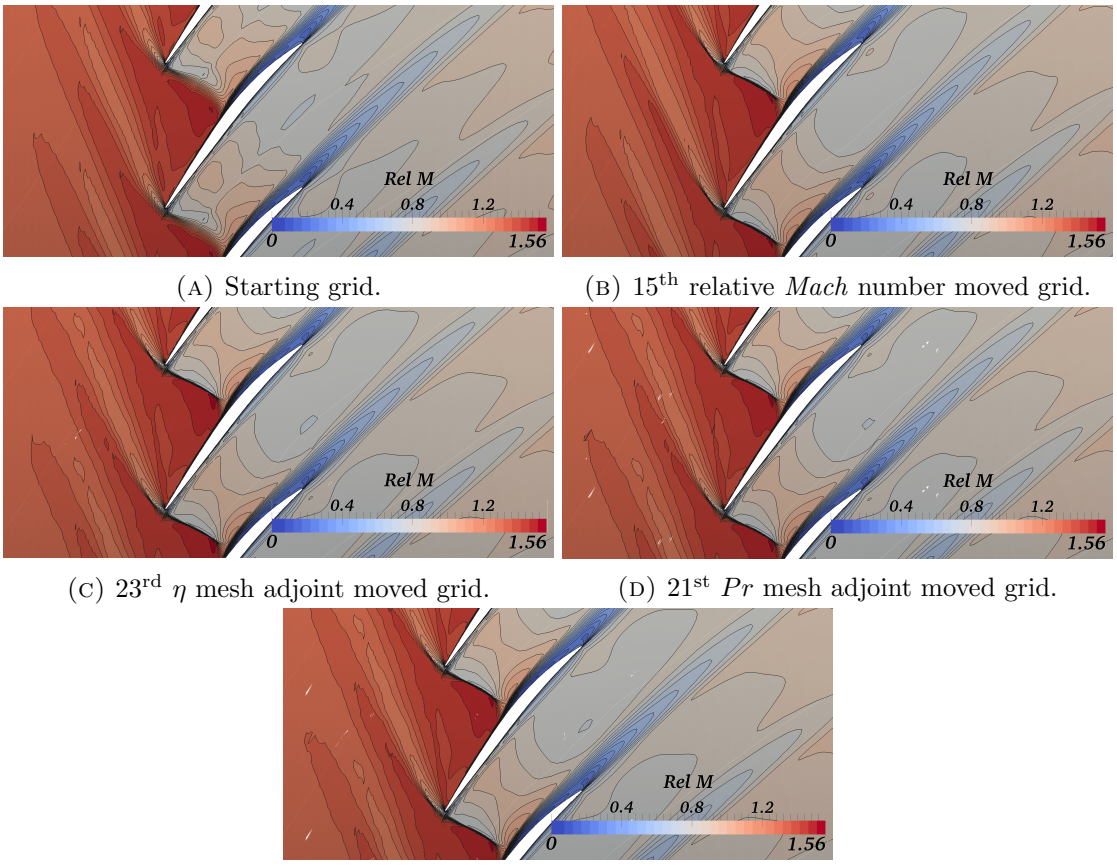
A comparison between the functional sensitivity at mid-span of the starting and final adapted grids is reported in figure 8.21. As for the fan case, the adapted grid's $\frac{df}{dx}$ map is much sharper *w.r.t.* that of the hand-generated one. Now the sensitivity around main flow features will be more accurate, as their location and intensity are better estimated. Again, unlike the feature-based approach, this adaptation method, will also target important regions that are not flagged by any flow quantity. As shown in the previous chapter, significant error is present in the *reversed wake*. This is captured by the initial

[3]As discussed in chapter 7, this is due to the dimensions of the quantity of interest.

mesh and its resolution improved by the adaptation. An additional characteristic that should be mentioned is the downstream region. In fact, the starting grid mesh adjoint, shows a limited sensitivity around the wake and more vigorous variation at the outflow plane. Node movement in these regions has clearly dampened this behaviour that is not present in any of the adapted grid's sensitivity map. This is not to mean that they have disappeared: by reducing the range, it could be seen that they were still present. Therefore the adaptation's effect has been that of improving the functional sensitivity map, thus yielding a more accurate estimation of it in an optimisation process. Of course this will aid the adjoint error estimation, in fact, as was also shown for the fan case, there will be less inaccuracies.

The grids resulting at mid-span, along with the starting and feature-adapted ones are reported in figure 8.22. Clearly, the mesh-adjoint based movement has targeted the strongest parts of the shock, as these have been appropriately clustered. The grid's behaviour is very similar to that of the feature-based technique. However, differences occur at the inlet block. In fact, the relative *Mach* number mesh movement has attempted to cluster nodes towards the shock's upstream propagation. On the other hand, the adjoint process has moved nodes towards the *reversed wake*. A further difference may also be seen where the wake forms on the blade suction side and throughout its propagation. In fact, the feature-based approach has targeted the entire propagation, clustering nodes towards the interface between this and the rest of the flow. On the other hand, in the adjoint-process, nodes have been moved towards the first portion of the wake, just downstream of the separation. Relocation has also occurred at the interface between this feature and the bulk of the flow, however it appears to be weaker *w.r.t.* that of the feature-case.

A final mention concerns the adapted grid's flow. To this end the relative *Mach* number behaviour at mid-span is provided in figure 8.23. All the adapted grids show a very similar behaviour, and have mainly improved the strongest parts of the shock. In fact, the lambda structure of it's interaction with the boundary layer is now clearly visible. Unfortunately, the adjoint-based node movement has not been able to improve the blade *LE* issue: the corner point numerical dissipation is still clearly visible due to the shock's increased thickness.



(A) Starting grid.

(B) 15$^{\text{th}}$ relative *Mach* number moved grid.

(C) 23$^{\text{rd}}$ η mesh adjoint moved grid.

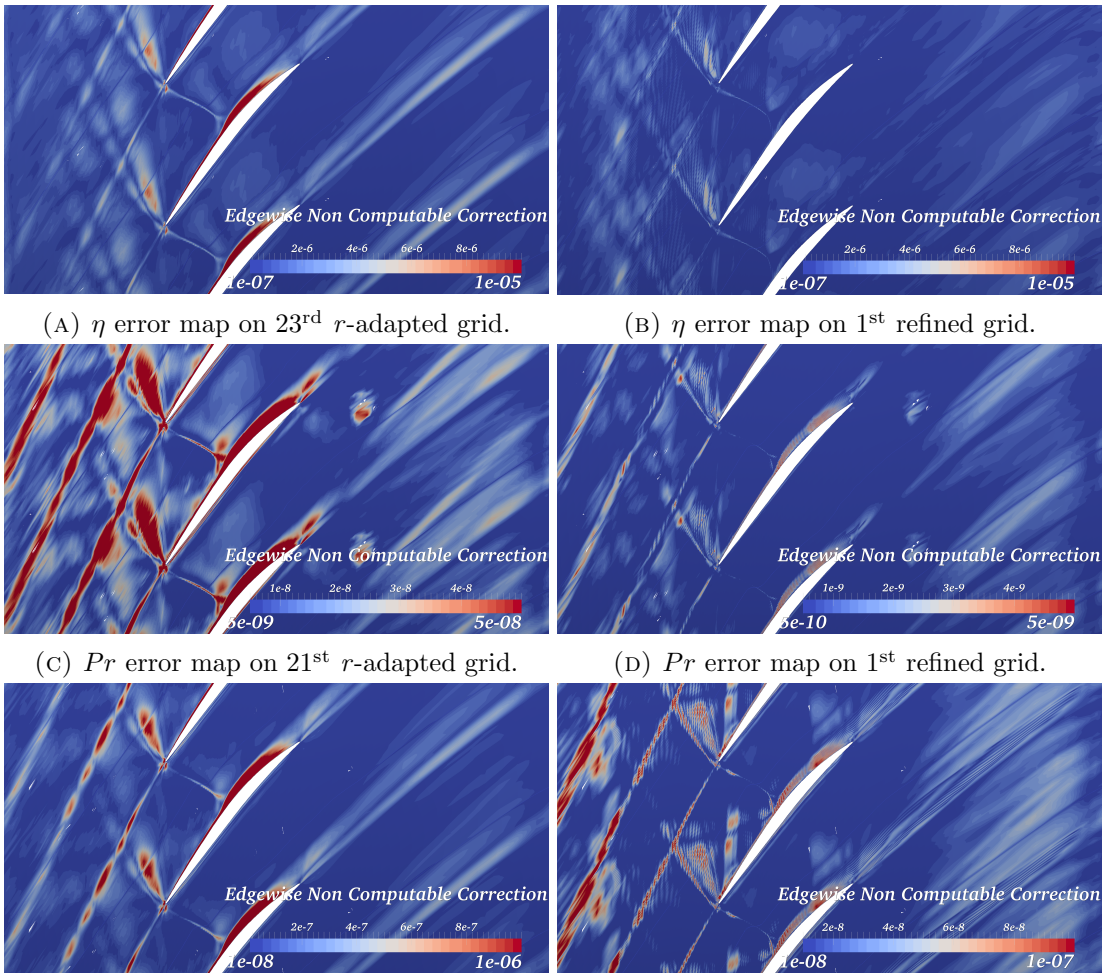
(D) 21$^{\text{st}}$ Pr mesh adjoint moved grid.

(E) 17$^{\text{th}}$ $\dot{m}$ mesh adjoint moved grid.

FIGURE 8.23: Relative *Mach* number at mid-span for starting and r-adapted meshes.

8.4.2 Part IIA: Efficient Edgewise Adjoint Mesh Refinement

The second step in the process consisted of refining the selected r-adapted mesh adding a fixed amount of nodes in the region of $300 \sim 400$k. Therefore, the error on the moved grids was determined employing the efficient edgewise *non-computable correction* (as discussed in section 6.3.2). These maps at mid-span, for the three quantities of interest, are shown in figures 8.24a, 8.24c and 8.24e. As in previous adjoint-based errors, in this case too the difference between them is due to the functional dimensions. In fact, other than that, they are very similar highlighting the same regions that were seen in chapter 7. However, in this case the flow and adjoint components of the error appear to be more evenly weighted, as the wake has a higher error intensity *w.r.t* the standard *non-computable correction* calculation. On the Pr sensor field, the careful reader will have noticed a small region of high inaccuracy just downstream of the blade pressure side. This is actually a patch where noise strongly affected the very low sensitivity mesh adjoint map causing erratic movement (see figure 8.22d for the relative mesh).



(A) η error map on 23$^{\text{rd}}$ r-adapted grid.

(B) η error map on 1$^{\text{st}}$ refined grid.

(C) Pr error map on 21$^{\text{st}}$ r-adapted grid.

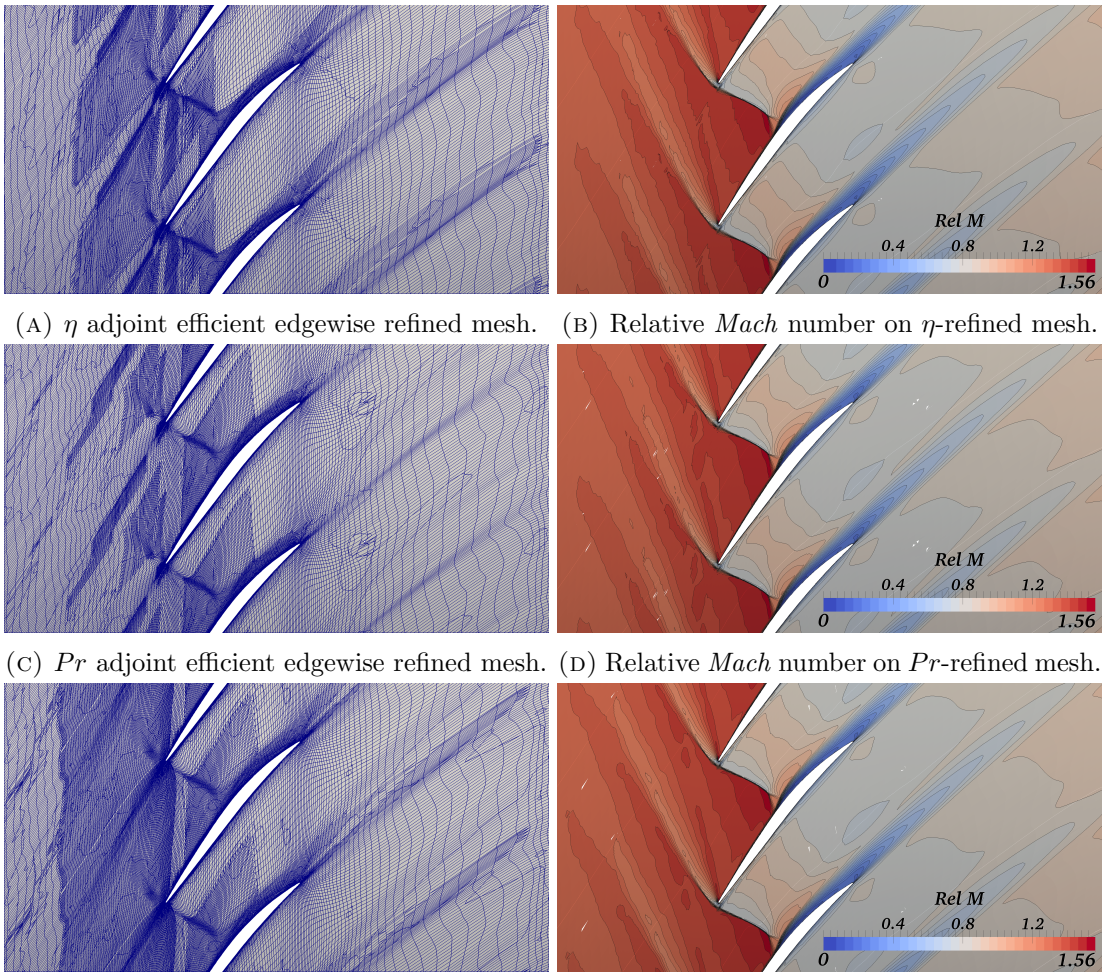
(D) Pr error map on 1$^{\text{st}}$ refined grid.

(E) $\dot{m}$ error map on 17$^{\text{th}}$ r-adapted grid.

(F) $\dot{m}$ error map on 1$^{\text{st}}$ refined grid.

FIGURE 8.24: Efficient edgewise *non-computable correction* error map after the repeated mesh movement and the first refined step at mid-span.

The resulting grids at mid-span, obtained by refining the moved meshes using sensors in figures 8.24a, 8.24c and 8.24e are shown in figures 8.25a, 8.25c and 8.25e, respectively. The differences between the three performance quantities are now strongly visible. Unfortunately, it is difficult to quantify how the movement has affected the refinement. In fact, looking at the pressure-ratio case, this has a more efficient alignment in the upstream block *w.r.t.* efficiency, despite the former having had two r-adaptation steps less. This does not mean that the movement does not bring valuable advantages: in fact, as shown in sections 8.3.2.1 and 8.3.2.2, the node relocation allowed an improved estimation of the error sources. An important difference with the adapted meshes of the previous chapter, concerns the refinement of flow features. In fact, it was previously shown how the nodes were added at the *reversed wake* and entire suction side of the blade, with very limited adaptation being spread along the passage and downstream of the rotor. In this case, the stronger parts of the shock have attracted considerable attention, with the refinement placing large blocks along the passage. This results from



(A) η adjoint efficient edgewise refined mesh. (B) Relative *Mach* number on η-refined mesh.

(C) Pr adjoint efficient edgewise refined mesh. (D) Relative *Mach* number on Pr-refined mesh.

(E) $\dot{m}$ adjoint efficient edgewise refined mesh. (F) Relative *Mach* number on $\dot{m}$-refined mesh.

FIGURE 8.25: Resulting adapted grids using the efficient edgewise *non-computable correction* along with relative solution at mid-span.

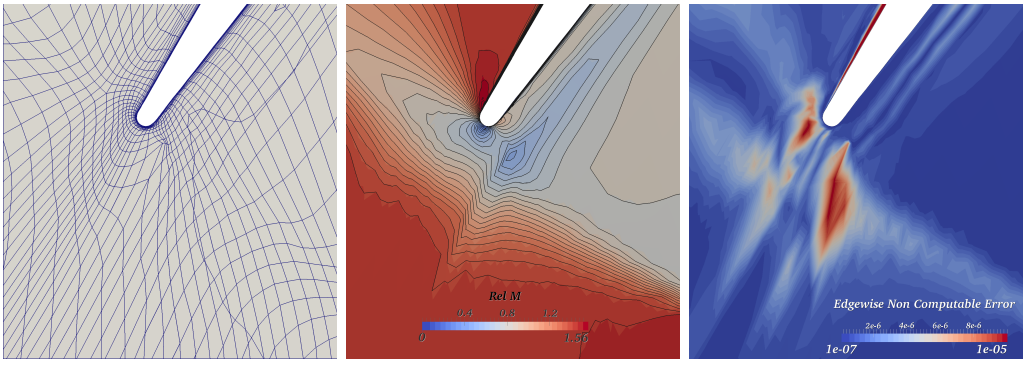
the modified adjoint-error for refinement, equally weighting flow and dual solutions.

The most significant difference between the three adapted grids concerns the wake refinement. While for efficiency and mass-flow this has spread over its entire propagation, for the pressure-ratio it has stopped at the blade *TE*. By looking at the relative error maps in figures 8.24a, 8.24c and 8.24e, it is clear that the efficiency and mass-flow error have approximately the same weighting upstream and downstream of the blade, this is not the case for the pressure-ratio. In fact, the *reversed-wake* magnitude is far stronger than the physical wake's one.

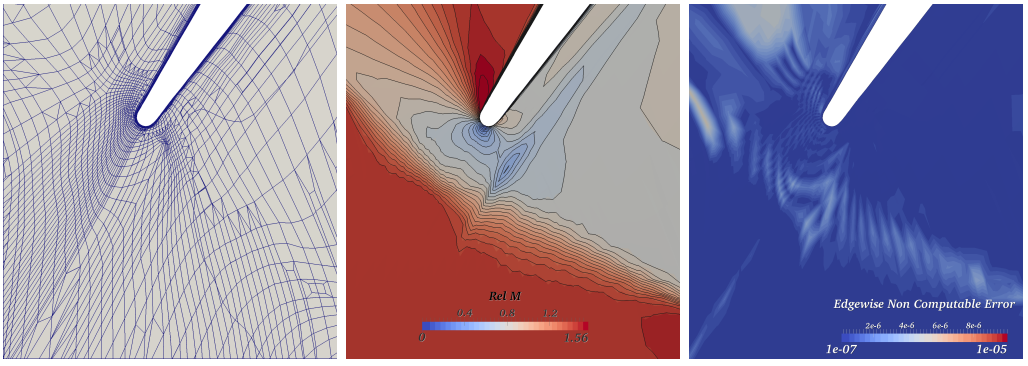
The resulting error on these meshes is shown in figures 8.24b, 8.24d and 8.24f. In all three cases, this has sensibly decreased as all the inaccuracies have now disappeared or are very low in magnitude. In particular, for the pressure-ratio and mass-flow, the scale range had to be decreased to be able to see where any sources of error were.

As for previous cases, it is also interesting to analyse the resulting flow. To this end, the relative *Mach* number at mid-span for the three adapted grids is shown in figures

8.25b, 8.25d and 8.25f. They all present nearly identical behaviour, with the only visible difference with the flow resulting from the previous step movement, being the shock resolution along the passage and around the blade *LE*. In particular, the lambda structure contours where it interacts with the boundary layer are improved.
A final note regarding the error is the high numerical dissipation region at the blade *LE*. To this end, figure 8.26 has been included to show the grid, flow and sensor evolution throughout the refinement for the efficiency, with other quantities having a very similar behaviour. In all cases, the problematic region is highlighted and targeted for refinement. The consequent flow solution shows an improvement, narrowing the erroneous behaviour. Nevertheless, as in chapter 7, further refinement is necessary to completely remove the issue.



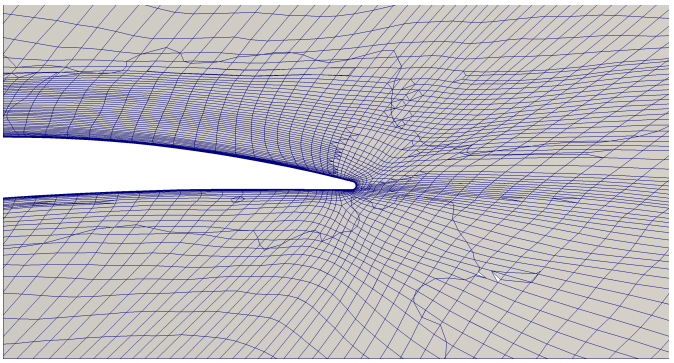
(A) After 23 η mesh adjoint-based movement steps.



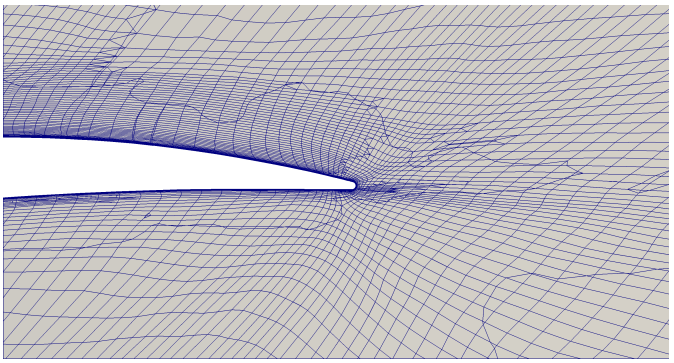
(B) After a single η efficient edgewise *non-computable correction* refinement.

FIGURE 8.26: Blade *LE* numerical diffusion issue for the efficiency adaptation process.

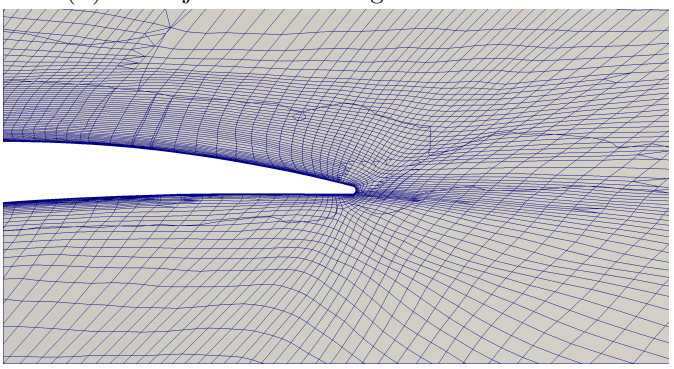
To prove the edgewise nature of the refinement process devised, figures 8.27a, 8.27b and 8.27c have been included. As it can be seen, nodes have only been added to the edges normal to the wake propagation, thus achieving *anisotropic* refinement.
As for all adjoint-related adaptation processes, the most important characteristic concerns the performance quantity estimate of the resulting grids. The evolution of efficiency, pressure-ratio and mass-flow are reported in tables 8.4, 8.5 and 8.6. In all cases, the refinement improves the functional, pushing it towards the target 47M mesh.



(A) η adjoint efficient edgewise refined mesh.



(B) Pr adjoint efficient edgewise refined mesh.



(C) $\dot{m}$ adjoint efficient edgewise refined mesh.

FIGURE 8.27: Refinement *anisotropy* at the blade *TE* at mid-span.

As stated in chapter 7, the accuracy of efficiency, pressure ratio and mass flow are tightly coupled. This because they all share the outlet as one of the surfaces of integration and η and Pr are determined by mass averaging. Moreover, as efficiency is computed using equation 7.1, the pressure ratio ought to be accurately predicted. Therefore, the improvement in one of the quantities will generally aid that of the other two. This is consistent with the behaviour seen in tables 8.4, 8.5 and 8.6. In particular, considering all three quantities, it may be seen that more mesh adjoint-based r-adaptation steps allow better functional estimates. Similarly for the refinement, the more nodes are added, the better is the prediction of all three quantities.

Case	# Nodes	η (%)	$Log_{10}(\Delta\eta)$	Pr	$\dot{m}$ ($\frac{kg \cdot m}{s}$)
Starting Grid	670056	84.4100	-0.2328	2.0815	20.0191
23^{rd} move iter.	670056	84.4708	-0.2807	2.0843	20.0567
Refined 23^{rd} move iter.	966687	84.7412	-0.5952	2.0938	20.1332
Target Mesh	46763130	84.9950	——	2.0963	20.1756

TABLE 8.4: Mass averaged adiabatic efficiency between inlet and outlet comparison. Fourth column represents the difference in performance *w.r.t.* that of the target grid.

Case	# Nodes	Pr	$Log_{10}(\Delta Pr)$	η (%)	$\dot{m}$ ($\frac{kg \cdot m}{s}$)
Starting Grid	670056	2.0815	-1.8297	84.4100	20.0191
21^{st} move iter.	670056	2.0835	-1.8928	84.4590	20.0482
Refined 21^{st} move iter.	1010220	2.1014	-2.2914	85.0284	20.2081
Target Mesh	46763130	2.0963	——	84.9950	20.1756

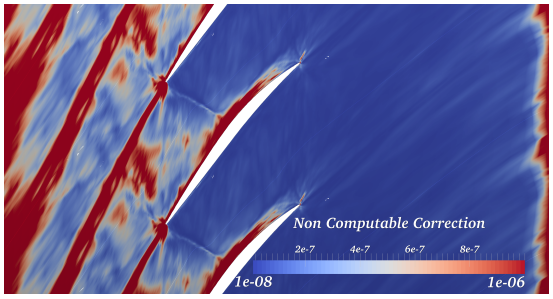
TABLE 8.5: Mass averaged absolute total pressure between inlet and outlet comparison. Fourth column represents the difference in performance *w.r.t.* that of the target grid.

Case	# Nodes	$\dot{m}$ ($\frac{kg \cdot m}{s}$)	$Log_{10}(\Delta\dot{m})$	η (%)	Pr
Starting Grid	670056	20.0191	-0.8055	84.4100	2.0815
17^{th} move iter.	670056	20.0469	-0.8904	84.4414	2.0834
Refined 17^{th} move iter.	1083435	20.1771	-2.8239	84.9221	2.0980
Target Mesh	46763130	20.1756	——	84.9950	2.0963

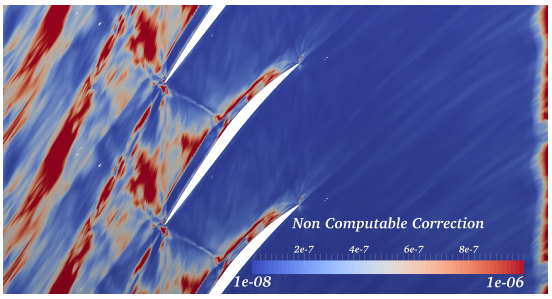
TABLE 8.6: Area averaged mass flow over the outlet comparison. Fourth column represents the difference in performance *w.r.t.* that of the target grid.

8.4.3 Part IIB: Embedded-Grid Edgewise Adjoint Mesh Refinement

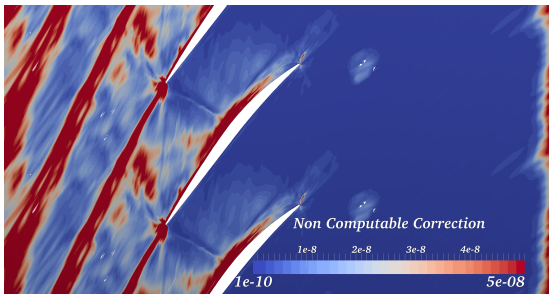
Starting from the selected r-adapted grids, in this case the mesh error was determined in the same manner as the standard procedure devised by [64]. However, the refinement process was modified by marking edges on the coarse mesh based on the embedded grid *non-computable correction*.



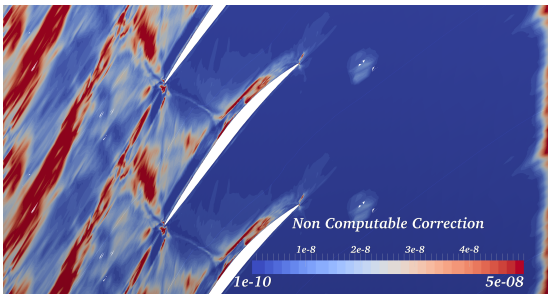
(A) η error map on 23$^{\text{rd}}$ r-adapted grid.



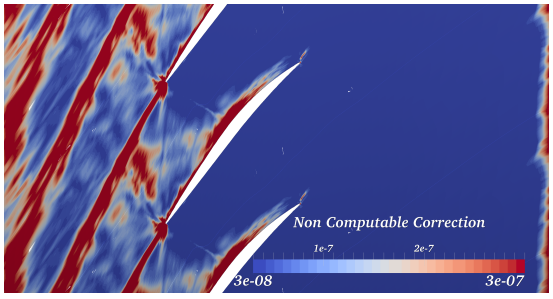
(B) η error map on 1$^{\text{st}}$ refined grid.



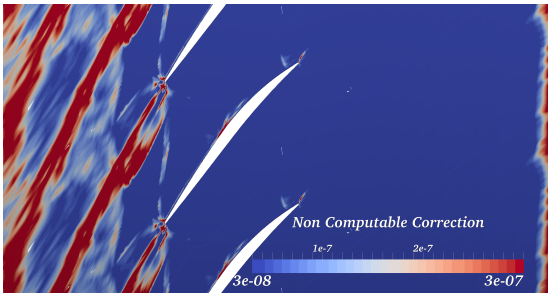
(C) Pr error map on 21$^{\text{st}}$ r-adapted grid.



(D) Pr error map on 1$^{\text{st}}$ refined grid.



(E) $\dot{m}$ error map on 17$^{\text{th}}$ r-adapted grid.

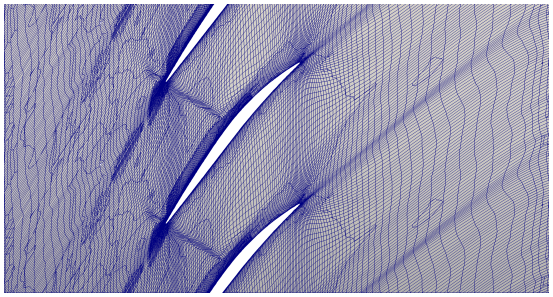


(F) $\dot{m}$ error map on 1$^{\text{st}}$ refined grid.

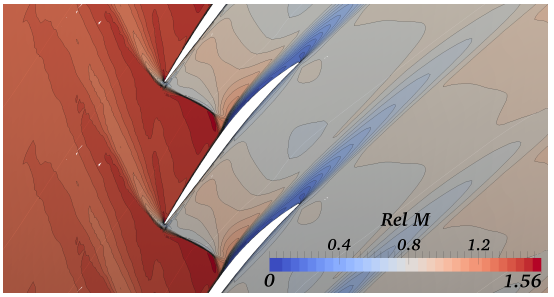
FIGURE 8.28: Embedded *non-computable correction* error map after the repeated mesh movement and the first refined step at mid-span.

The error map at mid-span is displayed in figures 8.28a, 8.28c and 8.28e. Unlike the *non-computable correction* evaluated in the previous chapter, the range had to be reduced to be able to appropriately highlight the regions with high inaccuracies. This is the case for all performance quantities, indicating that the mesh-adjoint r-adaptation has been more successful in reducing errors *w.r.t.* the feature-based approach, as expected. On the other hand, the same regions of the domain are flagged as having a high error, and as previously discussed, these appear mainly in the upstream parts where adjoint weighting

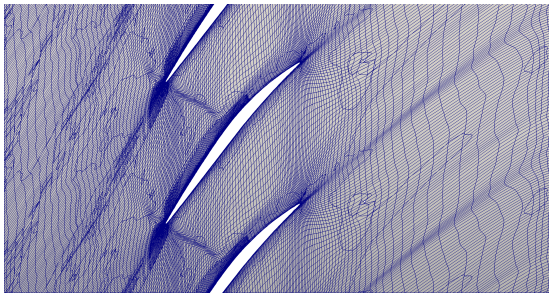
is stronger. An interesting characteristic that appears only in the pressure-ratio *non-computable correction* map, is the high error spot just downstream of the blade *TE* on the pressure side. This is due to noise in the mesh adjoint causing a limited region of erratic movement, also highlighted by the sensor discussed in the previous section.



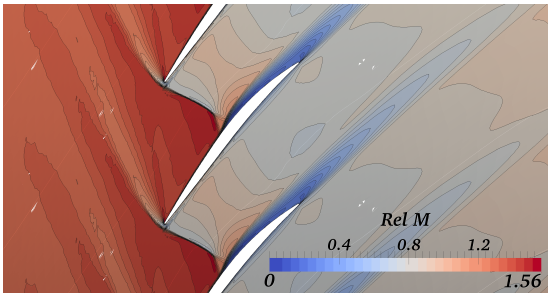
(A) η adjoint edgewise refined mesh.



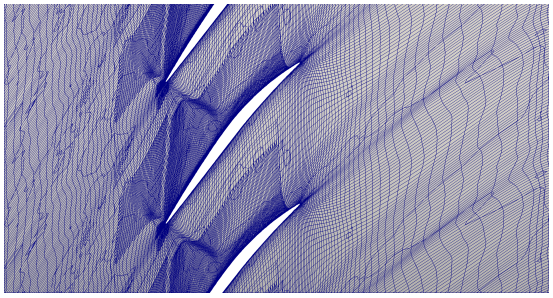
(B) Relative *Mach* number on η-refined mesh.



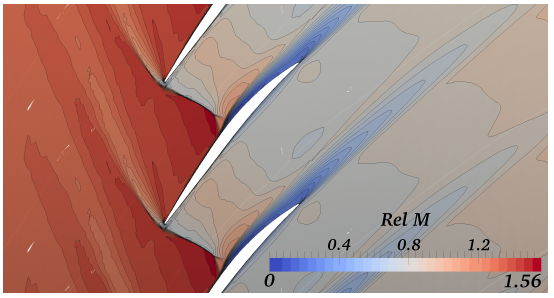
(C) Pr adjoint edgewise refined mesh.



(D) Relative *Mach* number on Pr-refined mesh.



(E) $\dot{m}$ adjoint edgewise refined mesh.



(F) Relative *Mach* number on $\dot{m}$-refined mesh.

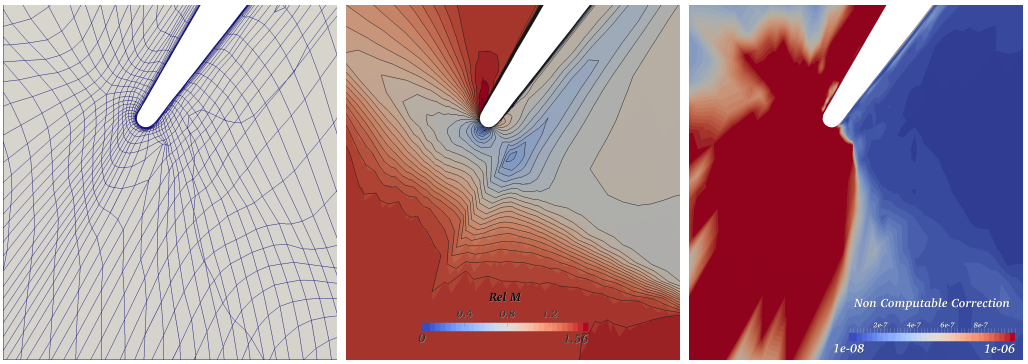
FIGURE 8.29: Resulting adapted grids using the embedded *non-computable correction* with edge-based marking along with relative solution at mid-span.

The resulting grids achieved by refining based on the sensors shown in figure 8.28a, 8.28c and 8.28e are reported in figures 8.29a, 8.29c and 8.29e. The first characteristic that may be noted is the limited refinement that efficiency and pressure-ratio require *w.r.t.* mass-flow. While in the previous case it was not possible to conclude whether the extra grid movement steps for the η and Pr processes had any beneficial effect, in this case it is obvious. In fact, the mass-flow refinement at mid-span propagates further into the passage. It is thought that in the previous section, the stronger contribution of the flow features to the overall error masked part of the movement's impact. A significant difference with the results in the previous chapter concerns the *reversed wake*. In fact, the node enrichment is rather limited upstream of the rotor, indicating that either the

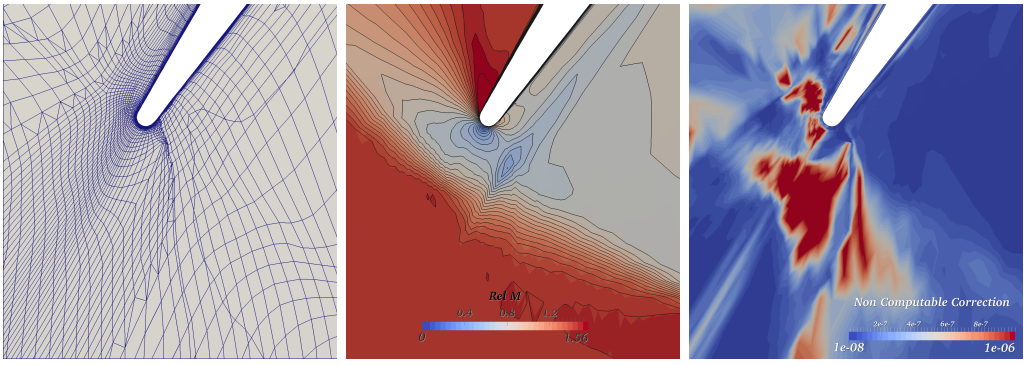
movement has reduced its contribution to the overall error, or that it uncovered an area of high inaccuracy elsewhere.

The *non-computable correction* evaluated on the refined grids are reported in figures 8.28b, 8.28d and 8.28f. As in chapter 7, the regions where the refinement has been introduced have caused the insurgence of low error voids. Again, the range of the sensor fields had to be reduced to be able to see where the sources of inaccuracies still occurred. To this end, comparing these maps with those of the previous chapter, it can be seen how the *non-computable correction* range has been decreased for every quantity of interest. This is indicative of the fact that the mesh-adjoint-based r-adaptation has worked better than its feature counterpart in reducing the sources of inaccuracies.

In terms of the resulting flow benefits, figures 8.29b, 8.29d and 8.29f show improvement of the resolution of the shock/boundary-layer interaction and the *LE* behaviour. To this end figure 8.30 has been included to display the evolution of shock capturing and error minimisation at the inter-block boundary. As in previous cases, further refinement is required to completely eliminate the numerical dissipation.



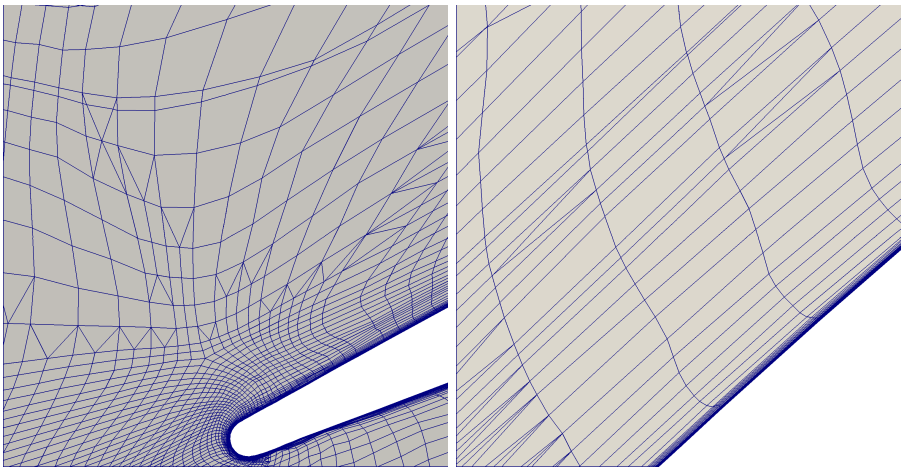
(A) After 23 η mesh adjoint-based movement steps.



(B) After a single η embedded *non-computable correction* edgewise refinement.

FIGURE 8.30: Blade *LE* numerical diffusion issue for the efficiency adaptation process.

As in section 8.3.4, despite the refinement being carried out where the edgewise error met the threshold, the resulting grid shows very limited *anisotropic* node enrichment. In this case, this behaviour is thought to be due to the combination of typical over-refinement caused by the adaptation software and the shape of error regions, which are



(A) Blade *LE*. (B) Suction-side near *TE*.

FIGURE 8.31: Examples of edge-based refinement.

relatively regular blocks. Nevertheless, limited examples of directional edge refinement are provided in figure 8.31.

Concerning the performance quantities of interest, these are reported in tables 8.7, 8.8 and 8.9. Comparing these with the values obtained in section 8.4.2, there is no clear better approach. While efficiency employing the embedded-grid error estimation sensibly outperforms that using the edgewise residuals, for pressure-ratio they seem to have similar accuracies, while for mass-flow the situation reverses. Nevertheless, for all quantities of interest, using either approach has improved the accuracy.

As in the previous analysis, it is clear that once again the adaptation of one of the three quantities of interest has improved them all. This was related to the functional's surface of integration being the same for each one, the dependence of η and Pr on mass flow and that of efficiency on pressure-ratio.

Case	# Nodes	η (%)	$Log_{10}(\Delta\eta)$	Pr	$\dot{m}$ $(\frac{kg \cdot m}{s})$
Starting Grid	670056	84.4100	-0.2328	2.0815	20.0191
23^{rd} move iter.	670056	84.4708	-0.2807	2.0843	20.0567
Refined 23^{rd} move iter.	1091344	84.9558	-1.4089	2.1015	20.2048
Target Mesh	46763130	84.9950	——	2.0963	20.1756

TABLE 8.7: Mass averaged adiabatic efficiency between inlet and outlet comparison. Fourth column represents the difference in performance *w.r.t.* that of the target grid.

Case	# Nodes	Pr	$Log_{10}(\Delta Pr)$	η (%)	$\dot{m}$ ($\frac{kg \cdot m}{s}$)
Starting Grid	670056	2.0815	-1.8297	84.4100	20.0191
21^{st} move iter.	670056	2.0835	-1.8928	84.4590	20.0482
Refined 21^{st} move iter.	995568	2.1003	-2.3979	84.9462	20.1933
Target Mesh	46763130	2.0963	——	84.9950	20.1756

TABLE 8.8: Mass averaged absolute total pressure between inlet and outlet comparison. Fourth column represents the difference in performance *w.r.t.* that of the target grid.

Case	# Nodes	$\dot{m}$ ($\frac{kg \cdot m}{s}$)	$Log_{10}(\Delta \dot{m})$	η (%)	Pr
Starting Grid	670056	20.0191	-0.8055	84.4100	2.0815
17^{th} move iter.	670056	20.0469	-0.8904	84.4414	2.0834
Refined 17^{th} move iter.	1040010	20.1429	-1.485	84.7644	2.0942
Target Mesh	46763130	20.1756	——	84.9950	2.0963

TABLE 8.9: Area averaged mass flow over the outlet comparison. Fourth column represents the difference in performance *w.r.t.* that of the target grid.

8.4.4 Process Time Consumption

Estimated raw *CPU* time consumption for the adaptation process including 23 mesh-adjoint based mesh movement steps and a single refinement are reported in table 8.10.

Case	Minutes	Seconds
23 Adjoint Mesh Mov.	3081	16
1 Adjoint Ref.	453	49
Complete Adapt.	3535	5
47M Mesh	7233	1

TABLE 8.10: *NASA Rotor* 37 η full adaptation raw *CPU* timing: 23 mesh movement and 1 refinement steps.

Again, the adaptation process is faster *w.r.t.* the 47M grid, with the latter requiring over 50% more time. However, by comparing the results with table 7.9, it is clearly visible

that the feature-based mesh movement process is much faster, albeit less accurate. Again this can be traced back to the adjoint solver as on average the nonlinear took 5 minutes and 47 seconds, the dual solver 127 minutes and 18 seconds and the mesh adjoint 38 seconds. As previously discussed, speed improvements may be achieved by relaxing the convergence requirements of the adjoint solver. Moreover the use of the *GMRES* option or increasing the node movement for each adaptation step could also be beneficial.
Finally, concerning the refinement, by comparing its timing with that achieved in chapter 7 and factoring in the reduced number of adaptation steps, it may be seen that there is a significant gain in time consumption. In this case, the flow solver was run one time less, while the adjoint code two times less, thus avoiding the issues arising with hybrid elements and shortening time consumption.

8.5 Conclusion

Two different cases were analysed: a typical fan blade with splitter simulated at choke and a compressor case run at aerodynamic design point. Both cases present interesting flow features that complicate the adaptation process and require relatively robust flow and adjoint solvers.
New adjoint-based mesh adaptation techniques were developed and applied to either geometry. The first approach consisted in an r-adaptation employing the mesh adjoint output to appropriately cluster and align nodes in the regions of interest. By applying a *GG* operator to the mesh sensitivities, it was possible to determine the *Hessian* matrix. By splitting this into eigenvalues and eigenvectors, shrinking and orientation of edges in the direction of greatest variation of the functional could be achieved. By repeatedly applying this approach, it was shown that for different performance quantities, the technique consistently outperformed the feature-based mesh movement utilised in chapter 7. Additionally, clustering and alignment towards flow and adjoint complexities was clearly visible in the adapted grids.
Two different refinement strategies were employed to further enhance the functional accuracy. In the first case, the original *non-computable correction* formulation of [64] was modified by substituting flow and adjoint residuals evaluated on the embedded grid with the edge fluxes of the coarse mesh. The aim of this modification was to achieve a fully edgewise adjoint error. For both fan and compressor cases, this technique always showed an improvement of the quantity of interest, most importantly though, managing to refine in a directional manner. Comparison of this calculation with the standard *non-computable correction* showed that the adjoint weighting is weaker, thus allowing to better highlight flow complexities.

The second refinement approach consisted in a simple modification of the standard procedure devised by [64]. In fact, once the error was calculated on the embedded grid, rather than restricting it to the coarse mesh, it was employed to mark edges on the parent grid. Despite being an edge-based refinement process, it was noted that directionality is difficult to achieve due to over-refinement typical of the algorithm employed and the limited regions of error in the domain. Nevertheless, the process aided the overall estimation of the functional of interest in all cases analysed. Again, it was noticed that the main refinement occurred upstream where the adjoint variables are typically strongest.

Chapter 9

Conclusions and Future Work

The main aim and novelties of this work relate to the development of efficient and robust adjoint-adaptation methods, for application to turbomachinery grids. Most adjoint error estimation processes utilised to date have mainly been concerned with the procedure devised by [64]. This technique either employs cell-based refinement or complete mesh-regeneration. Moreover, the geometries generally considered are either 2D (e.g. *NACA* 0012) or 3D (e.g. *ONERA M6*) external flows, with very little effort to adapt grids for turbomachinery cases. In fact, to the best of the author's knowledge, no adjoint mesh adaptation technique has been successfully applied to jet engine components. Being able to automatically and reliably improve grid accuracy for more complex test-cases, would allow these techniques to be considered in an industrial setting.

The first approach devised employs well-known technology to modify a coarse starting grid for *NASA Rotor* 37. It consists of a sequential combination of feature-based *Hessian* mesh movement and cell-based adjoint refinement. Although, the two mesh adaptation strategies have been successfully combined before, this has been done simultaneously, by modifying the *Hessian* matrix to include adjoint error scaling. Moreover, this quantity has been employed to regenerate the entire mesh. In this case, the feature-based mesh movement has been used to improve the initial grid's alignment, thus avoiding the need to run the dual solver and separate adaptation processes for each functional. Once the quantities of interest converged, cell-based *non-computable correction* refinement was applied. It was shown that better alignment allows a more accurate representation of the adjoint error, thus improving the overall process's efficiency. The estimated accuracy of the three quantities of interest was improved, except for errors caused by repeated refinement of hybrid elements appearing to avoid *hanging nodes*. While the starting mesh movement step was fully automated, the refinement process was arbitrary. In fact, half the starting mesh nodes were added at each adaptation step, in an attempt to control the overall size of the embedded grid required by the adjoint-error estimation process.

Comparison with a fully feature-based approach showed significant benefits in using the combined process devised.

The second mesh adaptation approach, was fully adjoint-based and employed node movement and edge refinement to improve the grid accuracy. The r-adaptation strategy used the mesh adjoint output to determine the *Hessian* matrix of second derivatives. By decomposing this into its eigenvalues and eigenvectors, it was possible to determine stretching and directionality of the functional sensitivity. By relocating the nodes appropriately, the regions strongly impacting the performance quantity were better resolved. This process on its own showed consistent improvement of the functional estimates for a typical fan and a compressor blades. While in the former case improvements were clearly visible, for the compressor these were less obvious as the starting grid was not in *asymptotic range w.r.t.* the target grid. Additionally, in either case, the grid-sensitivity mesh movement significantly outperformed the case using relative *Mach* number.

Two different adjoint-based mesh refinement strategies were devised. Considering the *non-computable correction*, in the first case, embedded grid flow and adjoint residuals were substituted with edgewise ones on the coarse mesh. This allowed to define the error sensor on each coarse mesh edge. Additionally, it removed the need to generate an embedded mesh. Refinement based on this quantity showed a consistent improvement of the functional of interest for both fan and compressor cases. Moreover, directional refinement was clearly visible at the downstream wake. Interestingly, this error evaluation appeared to weight more evenly the flow and adjoint components of the sensor *w.r.t.* the standard computation on the embedded grid. This is believed to be due to the convergence level of the two fields on the coarse mesh: here they will have both reached numerical precision accuracy. On the other hand, when using the embedded grid, neither interpolated quantity is converged, meaning that the adjoint solution would be determined using an inaccurate flow field to determine the *Jacobian* matrix, additionally to the adjoint approximate quantities.

The second refinement strategy consisted in a simple modification to the standard *non-computable correction* adaptation strategy. In fact, it was noted that once the error had been evaluated on the embedded grid, this will be defined for nodes splitting the coarse mesh edges as well. Therefore, rather than restricting the sensor map onto the parent mesh, it is possible to mark for refinement coarse mesh edges if the fine grid nodes splitting them meet the desired threshold. Despite managing to achieve an edgewise adaptation procedure, the resulting refinement formed finer patches without any real directionality. This was attributed to the combination of the adaptation algorithm, causing a degree of over-refinement, and the error being high in fairly regular blocks. Nevertheless, this approach showed consistent improvement of the quantities of interest.

To summarise: a variety of efficient adjoint-based methods to adapt grids for turbomachinery cases have been developed, and positive improvements have been recorded.

9.1 Future Work

Concerning the mesh movement techniques employed, questions still remain as to the amount of mesh movement iterations that one may reliably carry out. In fact, while the quantity of interest may stall after a certain number of steps, the grid alignment continues to improve, albeit at extra expense in terms of simulation time and reduced mesh quality. Additionally, issues in terms of smoothness of the computed field were noticed. For example, in the case of the mesh adjoint movement, noise in the very low sensitivity regions could cause erratic movement that may be detrimental in terms of accuracy and convergence. Moreover, in the author's experience, when computing the *Hessian* matrix for any quantity, noise will generally affect the near wall regions. Therefore, it is necessary to devise a standard smoothing approach that may be reliably utilised in all cases. Once such a technique has been created, it will be possible to finalise the surface mesh movement, thus allowing all the mesh nodes to relocate to the optimal position.
Another necessary enhancement to the process concerns the implementation of alternative hexahedral refinement algorithms. In fact, it was seen that the increase in hybrid elements and their consequent refinement, caused significant errors in terms of accuracy. Therefore, it is suggested to introduce techniques from existing publications that remove hybrid elements marked for refinement (these would have been added by previous adaptation steps), and replace them with regularly split hexahedra.

Appendix A

Error Equidistribution Principle

In this section, the *principle of equidistribution* will be demonstrated following the same philosophy as [16]. The validity of it depends on whether the grid is in *asymptotic range*, i.e. the local error in each mesh cell scales with its size. In other words, all the physical features have been captured, but consequent reduction in mesh spacing will allow better resolution. If a grid does meet this requirement, it is possible to write the point-wise error E_r^i of cell i as:

$$E_r^i = \zeta_i \, h_i^p \tag{A.1}$$

Where it has been assumed that within the control volume ζ is a constant and h is the local spacing. The exponent p represents the solver's scheme order. As the domain's volume (V) can be written as a sum of the local mesh spacing (h) cubed ($3D$ domain), the average error becomes:

$$E_r^{\text{avg}} = \frac{1}{V} \sum_i \zeta_i \, h_i^p \, h^3 \Rightarrow E_r^{\text{avg}} = \frac{1}{V} \sum_i \zeta_i \, h_i^{p+3} \tag{A.2}$$

To minimise the overall error with a constant volume, it is possible to determine the following *Lagrangian* form:

$$\mathscr{L} = E_r^{\text{avg}} + \Lambda V \tag{A.3}$$

With Λ being the *Lagrangian* multiplier. As min ($\mathscr{L} = \frac{\partial \mathscr{L}}{\partial h_i} = 0$):

$$\frac{\partial\left(\frac{1}{V}\sum_i \zeta_i h_i^{p+3}\right)}{\partial h_i} + \frac{\partial \Lambda V}{\partial h_i} = 0$$

$$\frac{p+3}{V}\sum_i \zeta_i h_i^{p+2} + \Lambda \frac{\partial \sum_i h_i^3}{\partial h_i} = 0$$

$$\frac{p+3}{V}\sum_i \zeta_i h_i^p \cancel{h_i^2}^{1} + 3\Lambda \sum_i \cancel{h_i^2}^{1} = 0$$

$$\sum_i E_r^i = \frac{-3\Lambda V N_N}{p+3}$$

Where N_N is the number of grid nodes. Therefore:

$$\frac{\sum_i E_r^i}{N_N} = \frac{-3\Lambda V}{p+3}$$

Meaning that $E_r^{\text{avg}} = E_r^i$, i.e. to minimise the overall error, the individual cell one must be equal throughout the mesh [16].

Appendix B

Automatic Differentiation Example

In section 5.11, the main characteristics of automatic differentiation have been described. In this appendix a simple example will be provided to be able to clearly show how it is carried out (this is a more detailed version of that found in [168]), however, for more examples and in-depth analysis, the reader is referred to [199].
Considering the set of equations:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2 \\ 3x_1 + sin(x_2) \end{bmatrix} \tag{B.1}$$

The set of m variables is defined as:

$$\begin{cases} \widehat{t_1} = x_1 \\ \widehat{t_2} = x_2 \end{cases} \quad \textit{Inputs}$$

$$\begin{cases} \widehat{t_3} = \widehat{t_1}^2 \\ \widehat{t_4} = 3 \\ \widehat{t_5} = \widehat{t_4}\widehat{t_1} \\ \widehat{t_6} = sin(\widehat{t_2}) \end{cases} \quad \textit{Intermidiate variables}$$

$$\begin{cases} \widehat{t_7} = z_1 = \widehat{t_3} + \widehat{t_2} \\ \widehat{t_8} = z_2 = \widehat{t_5} + \widehat{t_6} \end{cases} \quad \textit{Outputs}$$

Then forward and reverse differentiation based on $\widehat{t_1}$ and $\widehat{t_7}$, respectively, can be written as:

Forward Mode $\widehat{t}_1$	**Reverse Mode $\widehat{t}_7$**
$\frac{\partial \widehat{t}_1}{\partial \widehat{t}_1} = 1$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_7} = 1$
$\frac{\partial \widehat{t}_2}{\partial \widehat{t}_1} = 0$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_6} = 0$
$\frac{\partial \widehat{t}_3}{\partial \widehat{t}_1} = \frac{\partial \widehat{t}_3}{\partial \widehat{t}_1}\frac{\partial \widehat{t}_1}{\partial \widehat{t}_1} = 2\widehat{t}_1$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_5} = 0$
$\frac{\partial \widehat{t}_4}{\partial \widehat{t}_1} = 0$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_4} = 0$
$\frac{\partial \widehat{t}_5}{\partial \widehat{t}_1} = \frac{\partial \widehat{t}_5}{\partial \widehat{t}_1}\frac{\partial \widehat{t}_1}{\partial \widehat{t}_1} + \frac{\partial \widehat{t}_5}{\cancel{\partial \widehat{t}_4}}\frac{\cancel{\partial \widehat{t}_4}}{\partial \widehat{t}_1}^{0} = \widehat{t}_4$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_3} = \frac{\partial \widehat{t}_7}{\partial \widehat{t}_7}\frac{\partial \widehat{t}_7}{\partial \widehat{t}_3} = 1$
$\frac{\partial \widehat{t}_6}{\partial \widehat{t}_1} = \frac{\partial \widehat{t}_6}{\partial \widehat{t}_2}\frac{\partial \widehat{t}_2}{\partial \widehat{t}_1} = 0$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_2} = \frac{\partial \widehat{t}_7}{\partial \widehat{t}_7}\frac{\partial \widehat{t}_7}{\partial \widehat{t}_2} = 1$
$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_1} = \frac{\partial \widehat{t}_7}{\partial \widehat{t}_3}\frac{\partial \widehat{t}_3}{\partial \widehat{t}_1}\frac{\partial \widehat{t}_1}{\partial \widehat{t}_1} + \frac{\partial \widehat{t}_7}{\cancel{\partial \widehat{t}_2}}\frac{\cancel{\partial \widehat{t}_2}}{\partial \widehat{t}_1}^{0} = 2\widehat{t}_1$	$\frac{\partial \widehat{t}_7}{\partial \widehat{t}_1} = \frac{\partial \widehat{t}_7}{\partial \widehat{t}_7}\frac{\partial \widehat{t}_7}{\partial \widehat{t}_3}\frac{\partial \widehat{t}_3}{\partial \widehat{t}_1} = 2\widehat{t}_1$

From this derivation, it should be observed that, in forward mode, the quantity $\frac{\partial \widehat{t}_8}{\partial \widehat{t}_1}$ is determined with little extra effort:

$$\frac{\partial \widehat{t}_8}{\partial \widehat{t}_1} = \frac{\partial \widehat{t}_8}{\partial \widehat{t}_5}\frac{\partial \widehat{t}_5}{\partial \widehat{t}_1}\frac{\partial \widehat{t}_1}{\partial \widehat{t}_1} + \frac{\partial \widehat{t}_8}{\cancel{\partial \widehat{t}_6}}\frac{\partial \widehat{t}_6}{\partial \widehat{t}_2}\frac{\cancel{\partial \widehat{t}_2}}{\partial \widehat{t}_1}^{0} = \widehat{t}_4 = 3$$

while, on the other hand, in reverse mode $\frac{\partial \widehat{t}_7}{\partial \widehat{t}_2}$ is easily computed. For what concerns the rest of the quantities, forward mode on $\widehat{t}_2$ is computed as:

Forward Mode $\widehat{t_2}$

$$\frac{\partial \widehat{t_1}}{\partial \widehat{t_2}} = 0$$

$$\frac{\partial \widehat{t_2}}{\partial \widehat{t_2}} = 1$$

$$\frac{\partial \widehat{t_3}}{\partial \widehat{t_2}} = 0$$

$$\frac{\partial \widehat{t_4}}{\partial \widehat{t_2}} = 0$$

$$\frac{\partial \widehat{t_5}}{\partial \widehat{t_2}} = \frac{\partial \widehat{t_5}}{\cancel{\partial \widehat{t_4}}} \overset{0}{\cancel{\frac{\partial \widehat{t_4}}{\partial \widehat{t_2}}}} + \frac{\partial \widehat{t_5}}{\cancel{\partial \widehat{t_1}}} \overset{0}{\cancel{\frac{\partial \widehat{t_1}}{\partial \widehat{t_2}}}} = 0$$

$$\frac{\partial \widehat{t_6}}{\partial \widehat{t_2}} = \frac{\partial \widehat{t_6}}{\partial \widehat{t_2}} \frac{\partial \widehat{t_2}}{\partial \widehat{t_2}} = cos(\widehat{t_2})$$

$$\frac{\partial \widehat{t_7}}{\partial \widehat{t_2}} = \frac{\partial \widehat{t_7}}{\cancel{\partial \widehat{t_3}}} \frac{\partial \widehat{t_3}}{\partial \widehat{t_1}} \overset{0}{\cancel{\frac{\partial \widehat{t_1}}{\partial \widehat{t_2}}}} + \frac{\partial \widehat{t_7}}{\partial \widehat{t_2}} \frac{\partial \widehat{t_2}}{\partial \widehat{t_2}} = 1$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_2}} = \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_5}}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_4}} \overset{0}{\cancel{\frac{\partial \widehat{t_4}}{\partial \widehat{t_2}}}} + \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_5}}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_1}} \overset{0}{\cancel{\frac{\partial \widehat{t_1}}{\partial \widehat{t_2}}}} + \frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} \frac{\partial \widehat{t_6}}{\partial \widehat{t_2}} = cos(\widehat{t_2})$$

while reverse mode on $\widehat{t_8}$ is:

Reverse Mode $\widehat{t_8}$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_8}} = 1$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} = \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\cancel{\partial \widehat{t_8}}}{\partial \widehat{t_5}} \frac{\cancel{\partial \widehat{t_5}}^{0}}{\partial \widehat{t_6}} + \frac{\partial \widehat{t_8}}{\partial \widehat{t_8}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} = 1$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} = \frac{\partial \widehat{t_8}}{\partial \widehat{t_8}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} + \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\cancel{\partial \widehat{t_8}}}{\partial \widehat{t_6}} \frac{\cancel{\partial \widehat{t_6}}^{0}}{\partial \widehat{t_5}} = 1$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_4}} = \frac{\partial \widehat{t_8}}{\partial \widehat{t_8}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_4}} + \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} \frac{\cancel{\partial \widehat{t_6}}^{0}}{\partial \widehat{t_4}} = \widehat{t}_1$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_3}} = \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_4}} \frac{\cancel{\partial \widehat{t_4}}^{0}}{\partial \widehat{t_3}} + \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} \frac{\cancel{\partial \widehat{t_6}}^{0}}{\partial \widehat{t_3}} = 0$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_2}} = \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_4}} \frac{\cancel{\partial \widehat{t_4}}^{0}}{\partial \widehat{t_2}} + \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_1}} \frac{\cancel{\partial \widehat{t_1}}^{0}}{\partial \widehat{t_2}} + \frac{\partial \widehat{t_8}}{\partial \widehat{t_8}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} \frac{\partial \widehat{t_6}}{\partial \widehat{t_2}} \frac{\partial \widehat{t_2}}{\partial \widehat{t_2}} = cos(\widehat{t}_2)$$

$$\frac{\partial \widehat{t_8}}{\partial \widehat{t_1}} = \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_4}} \frac{\cancel{\partial \widehat{t_4}}^{0}}{\partial \widehat{t_1}} + \frac{\partial \widehat{t_8}}{\partial \widehat{t_8}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_5}} \frac{\partial \widehat{t_5}}{\partial \widehat{t_1}} + \frac{\partial \widehat{t_8}}{\cancel{\partial \widehat{t_8}}} \frac{\partial \widehat{t_8}}{\partial \widehat{t_6}} \frac{\cancel{\partial \widehat{t_6}}}{\partial \widehat{t_2}} \frac{\cancel{\partial \widehat{t_2}}^{0}}{\partial \widehat{t_1}} = \widehat{t}_4$$

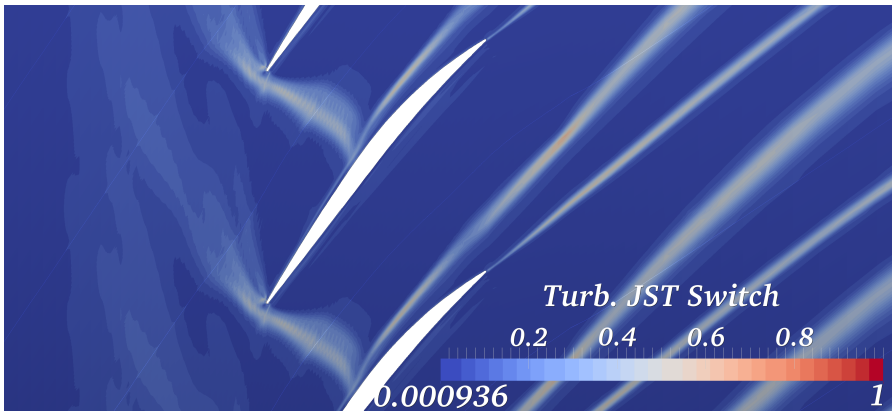
Appendix C

Feature-Based Mesh Adaptation Process applied to NASA Rotor 37

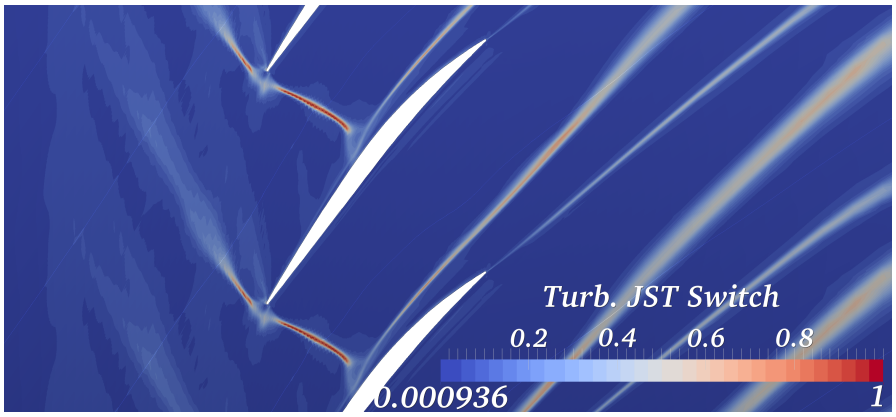
As discussed in chapters 7 and 8, a fully feature-based procedure was employed as a comparison for the processes applied to *NASA Rotor* 37. This will be described in this appendix.

The starting point for feature-based refinement was the 15^{th} relative *Mach* number-based moved grid extensively described in section 7.4.1. At this point the functionals of interest had converged, therefore refinement would be applied to further improve the grid accuracy. The sensor employed here is the edgewise *JST* switch as determined by the flow solver. The author applied a simple modification to *Hydra*'s source code to output the switch value for every equation for all edges. This was then read into the mesh adaptation software. The user can select whether to employ the *JST* switch of one of the *NS p.d.e.*s or the turbulence model. The distinction between the two possible choices, relates to the presence of differences of turbulent viscosity terms in the switch. A user defined threshold or range between 0 and 1, allows the edge marking process to be carried out, as described in section 6.4.

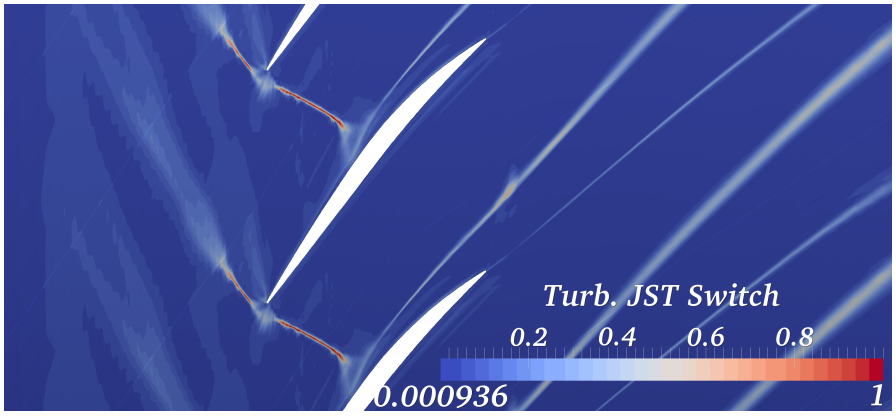
The *JST* value is 0 where flow has 2^{nd} order accuracy. This is generally the case for smooth parts of the flow. On the other hand, to avoid very large oscillations where flow complexities occur, it will be 1, corresponding to 1^{st} order accuracy. Therefore, to be able to have the most accurate solution, most edges should have a switch value of 0, unless a complex feature is present, where it should be 1 to dampen any oscillatory behaviour. Values in-between are not, strictly speaking, necessary, and should be targeted for refinement.



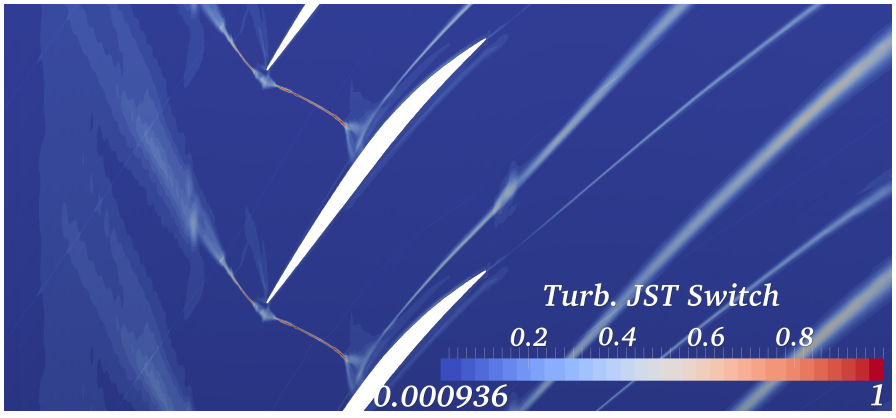
(A) Starting grid.



(B) 15$^{\text{th}}$ feature-based moved grid.

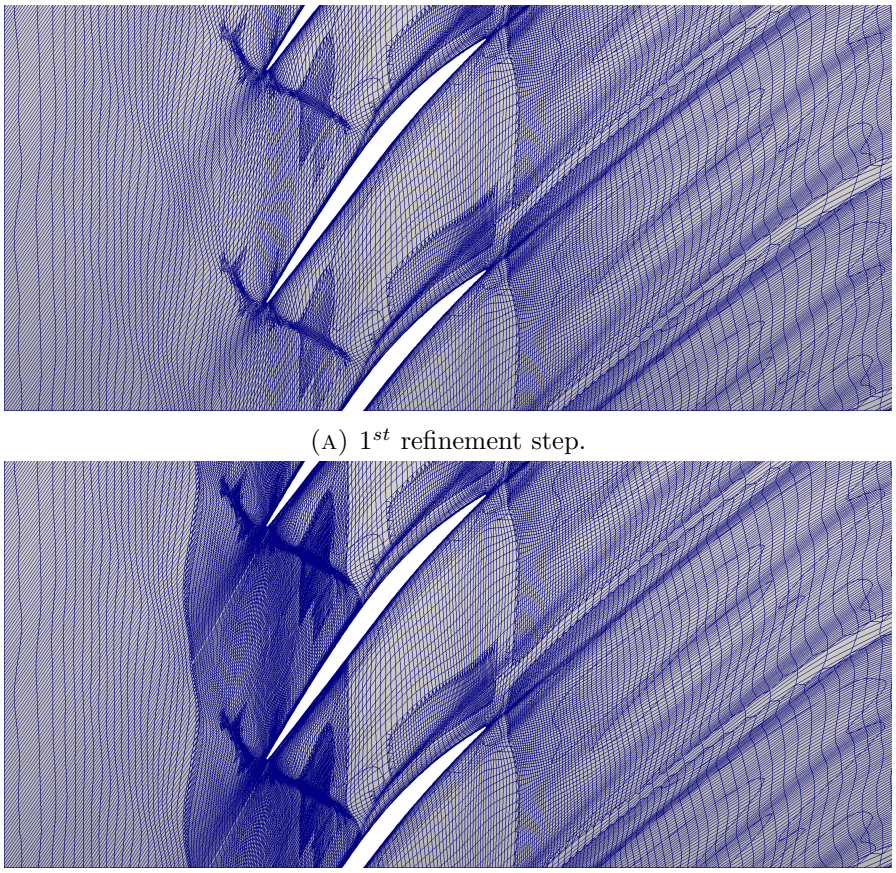


(C) 1$^{\text{st}}$ feature refinement.



(D) 2$^{\text{nd}}$ feature refinement.

FIGURE C.1: *NASA Rotor* 37 mid-span cuts of the turbulence model *JST*-switch.



(A) 1^{st} refinement step.

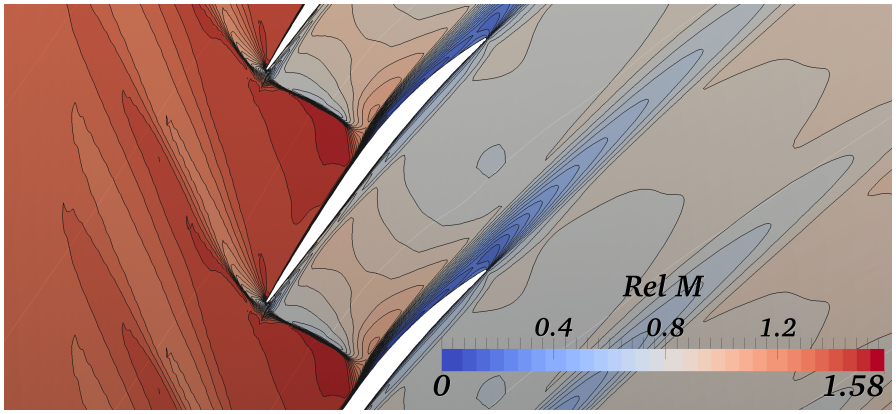
(B) 2^{nd} refinement step

FIGURE C.2: *NASA Rotor* 37 mid-span cuts of the two refined grids.

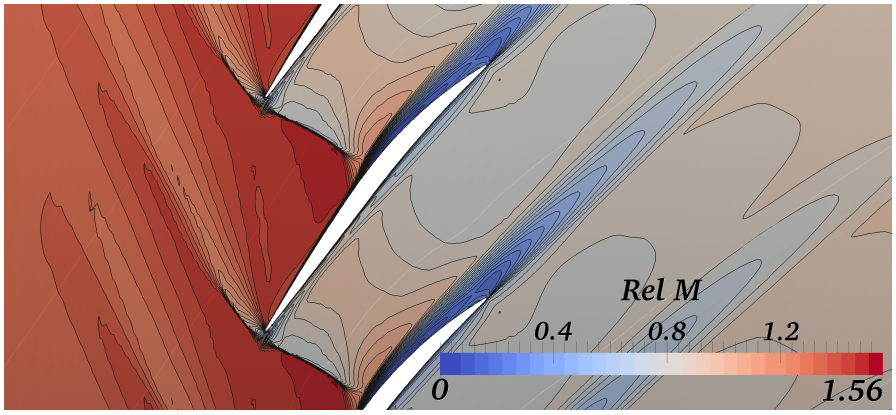
The *JST* switch value evolution from starting grid to that of the last mesh movement iteration, can be seen in figures C.1a and C.1b, respectively. While in the beginning, its value varies very smoothly across the flow complexities, the mesh movement has been able to cluster nodes appropriately. In fact, the sensor variation is constrained to a narrower region for the strongest features. It is interesting to note that the lambda nature of the shock, where it interacts with the boundary layer, is now visible. While the strongest regions of the discontinuity are improved, this is not the case at the blade *LE*. In fact, a very smooth patch can be seen.

Refining the moved grid, based on *JST* values of the turbulence model equation, results in the mesh shown in figure C.2a. In this case the edge-marking range was set to $0.45 - 0.55$. Given that the adjoint-based process also required the computation of the dual solution, along with the embedded mesh approximations, the feature-based refinement was allowed to approximately double the node count. Clearly, the adaptation process has mainly targeted the wake, which has been refined in an *isotropic* fashion. The reason behind this is a combination of flow characteristics and the refinement algorithm implementation. In fact, the wake changes position in the circumferential direction along the blade span. This will cause the edge marking process to spread out between the

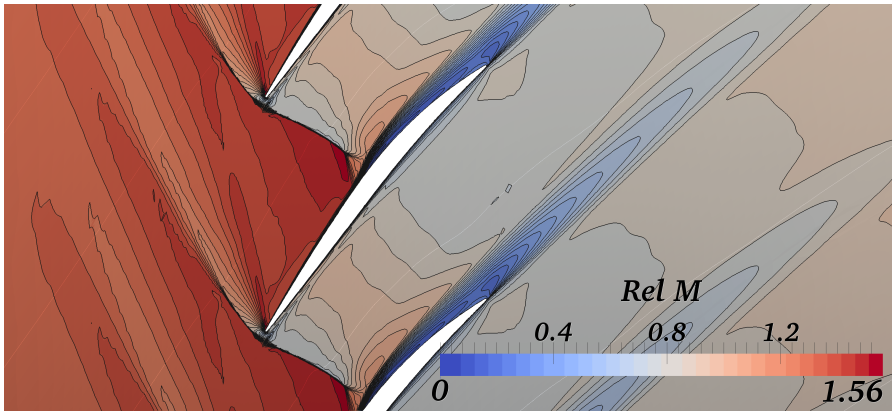
periodic boundaries throughout the rotor height. The refinement algorithm will then attempt to form new cells with edges meeting the error threshold. Unfortunately, this will cause a degree of over-refinement due to the limited templates available in the code. In fact, the edge-marking will start to propagate to meet the requirements. The other part of the flow targeted by the adaptation algorithm is the shock, particularly where it is strongest. The over-refinement is less accentuated, but still visible in the passage.



(A) 15^{th} feature-mesh-movement step.



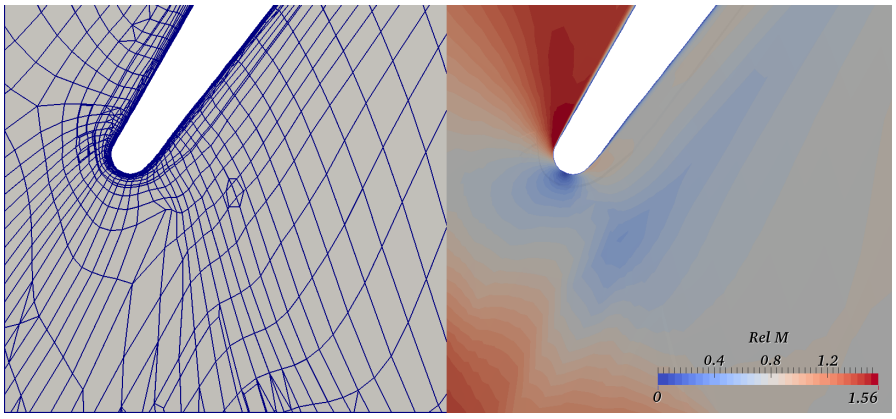
(B) 1^{st} refinement step.



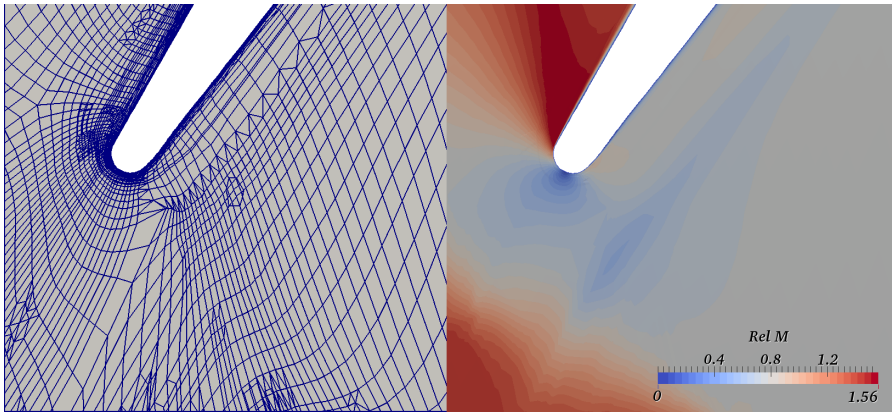
(C) 2^{nd} refinement step.

FIGURE C.3: *NASA Rotor* 37 mid-span cuts of the relative *Mach* number after feature-based refinement.

Comparing the flow field with that of the last feature-mesh movement step (figures C.3a and C.3b), clearly, the wake refinement has brought no visible improvement. However, the shock resolution is slightly better. The benefits can be seen in the *JST* switch map in figure C.1c. The high values are now constrained to very thin lines. Unfortunately, the upstream section is still suffering from a smoothly varying switch, indicating that further adaptation is needed to capture the complete shock propagation.
Adopting the *JST* parameter calculated for the turbulence model, has not brought great benefits by refining the downstream parts of the domain. Therefore, to be able to target the regions necessitating further refinement, the *NS* equations switch was used instead. The resulting grid at mid-span can be seen in figure C.2b. As desired, the adaptation has targeted the upstream region, neglecting the wake. Unfortunately, though, over-refinement is clearly visible. In fact, a finer square block has been placed around the blade *LE*. This behaviour is typical of the adaptation algorithm: it can be seen how the hybrid patches closing the refinement follow the original mesh lines. Nevertheless, there is further improvement in the resolution of the stronger parts of the shock, as shown in the relative *Mach* number and *JST* switch fields in figures C.3c and C.1d, respectively. More importantly, though, its variation in front of the blade *LE* and on the suction side has tightened up.
A final point of interest concerning the flow field is the blade *LE* numerical dissipation caused by a significantly stretched inter-block boundary corner point in the starting grid. The different flow field behaviour at this location on the two grids obtained employing the refinement, are shown in figure C.4. Even though it has decreased in size, the problematic patch is still present, thus causing error to occur. Despite applying significantly more refinement *w.r.t.* the adjoint case, it has not been removed.
The values of mass averaged adiabatic efficiency and absolute total pressure ratio between inlet and outlet, along with the area averaged mass flow evaluated over the exit plane are summarised in table C.1. Unfortunately, the feature-based refinement has had little impact in improving the values of interest. As discussed in section 7.4.1, the starting grid manages to compute a reasonable estimate of the performance quantities by chance, as it is not appropriately replicating the flow field. Therefore, the consequent relative *Mach* number-based mesh movement, is able to improve the feature resolution, at the expense of reducing the performance values. While the adjoint-based adaptation showed increased accuracy for all the quantities of interest, hindered only by the presence of hybrid elements, the fully feature process is not able to show the desired benefit. Of course, part fo the issue is due to the over-refinement, as if this were less aggressive nodes could be placed in more important regions of the flow. Nevertheless, it does help future adaptation steps, as fewer hybrid element templates mean better grid quality in following refinement steps.



(A) 1^{st} refinement step.



(B) 2^{nd} refinement step.

FIGURE C.4: *NASA Rotor* 37 blade *LE* grid (*LHS*) and relative *Mach* number (*RHS*).

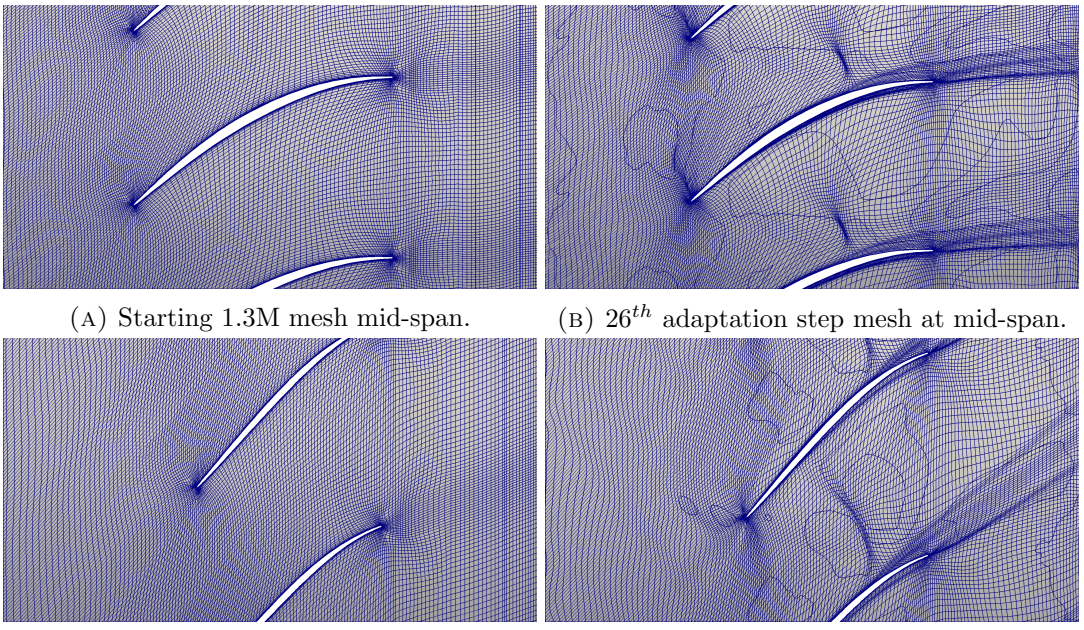
Grid	$N°Nodes$	η (%)	Pr	$\dot{m}$ ($\frac{kg \cdot m}{s}$)
Start	670056	84.4100	2.0815	20.0191
Final Feature Moved	670056	84.1287	2.0729	19.9638
1^{st} *JST* Refinement	1373112	83.9633	2.0704	19.9341
2^{nd} *JST* Refinement	2694775	84.3509	2.0805	20.0289
Target	46763130	84.9950	2.0963	20.1756

TABLE C.1: *NASA Rotor* 37 evolution of functionals of interest throughout the feature adaptation process.

Appendix D

Feature-Based Mesh Movement Applied to a Fan Blade

As was discussed in section 8.3.2, to provide a comparison with the mesh adjoint-based approach, a fully feature-based mesh movement employing the relative *Mach* number was also run. Examples of the relevant grids and resulting flow were not provided in that chapter to avoid confusing the reader. A brief comparison of the final adapted grid and solution with the starting ones, is carried out here.



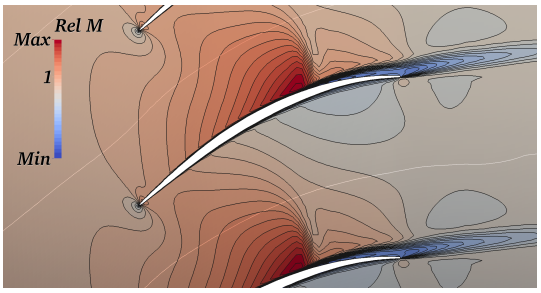
(A) Starting 1.3M mesh mid-span. (B) 26^{th} adaptation step mesh at mid-span.

(C) Starting 1.3M mesh 75% span. (D) 26^{th} adaptation step mesh at 75% span.

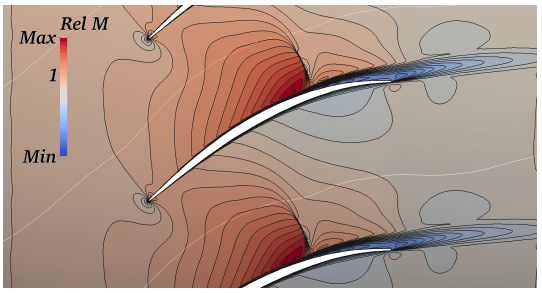
FIGURE D.1: Comparison of starting mesh with that obtained after 26 feature-based mesh adaptation steps.

In figure D.1, the main features of the flow have been captured, as they are clearly visible by simply observing figures D.1b and D.1d. Shock and wake have been attracting a considerable amount of nodes, as well as the boundary layer around the blade and, in particular, the *LE*. As expected, there is no clustering in regions where the adjoint sensitivity is high. In particular, the upstream mesh, presents a very smooth movement with nothing of particular relevance being visible.

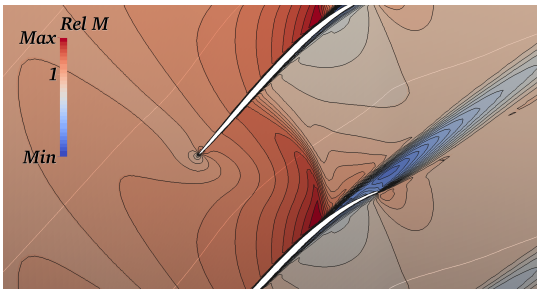
Analysing the resulting flow at mid- and 75% span, in figures D.2b and D.2d, it can clearly be seen that there is a strong improvement in the shock resolution as expected. By comparing these images with those produced by the equivalent adjoint process (see figures 8.7b and 8.7d), the two produce a very similar resolution of the flow complexities, however, by careful analysis, the feature-based technique seems to slightly outperform the other process.



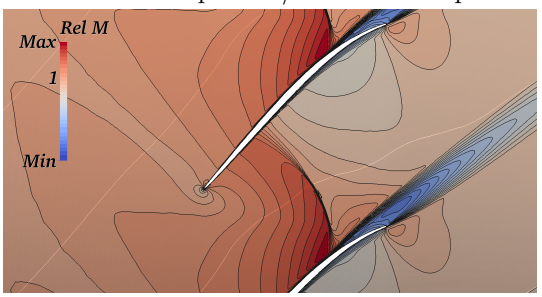
(A) Relative *Mach* number at mid span for the starting 1.3M mesh.



(B) Relative *Mach* number at mid span after 26 mesh adaptation/movement steps.



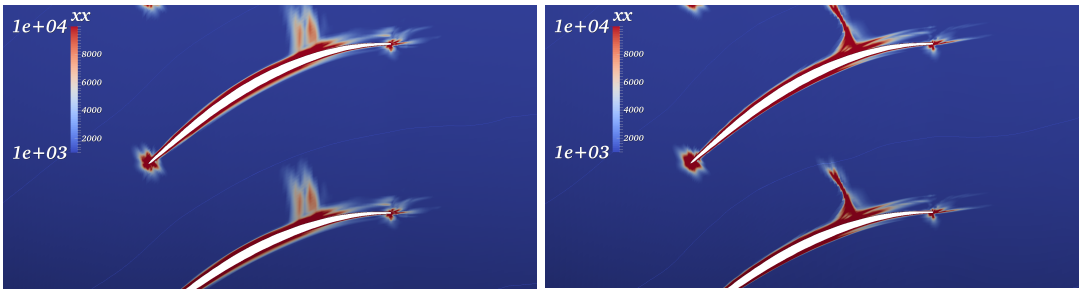
(C) Relative *Mach* number at 75% span for the starting 1.3M mesh.



(D) Relative *Mach* number at 75% span after 26 mesh adaptation/movement steps.

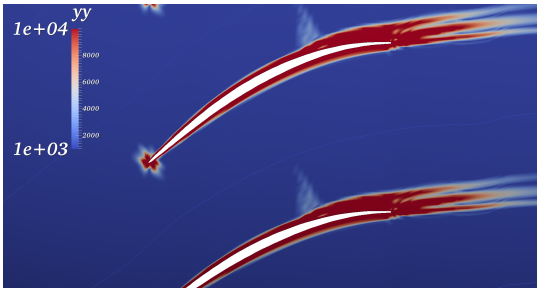
FIGURE D.2: Comparison of the starting mesh flow with that obtained after 26 mesh adaptation steps.

The error driving the mesh relocation has also been included, as its evolution is interesting to analyse. At mid-span, in figure D.3, the staring mesh diagonal terms of the *Hessian* matrix show limited resolution of the shock present in the vicinity of the blade *TE*. In fact, only the xx-component actually highlights the presence of the physical feature. On the other hand, the starting mesh seems appropriate to resolve the wake with good accuracy: this is strongly highlighted by the yy and zz components. Concerning the final mesh obtained through the adaptation process, it is clear that the shock is now visible in all terms and is slender, indicating good resolution. Moreover, the wake error

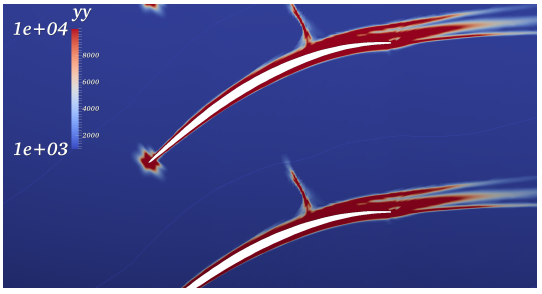


(A) Starting mesh xx-component.

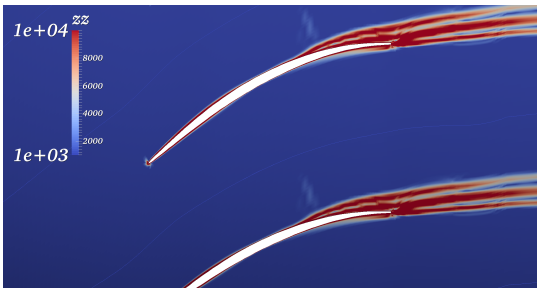
(B) 26^{th} adaptation step xx-component.



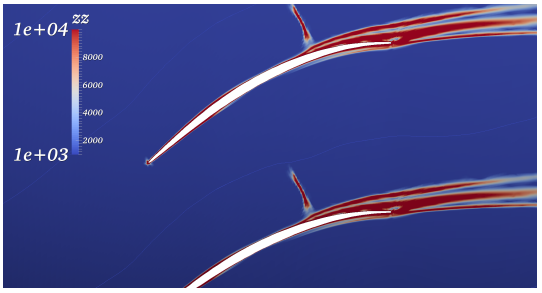
(C) Starting mesh yy-component.



(D) 26^{th} adaptation step yy-component.



(E) Starting mesh zz-component.

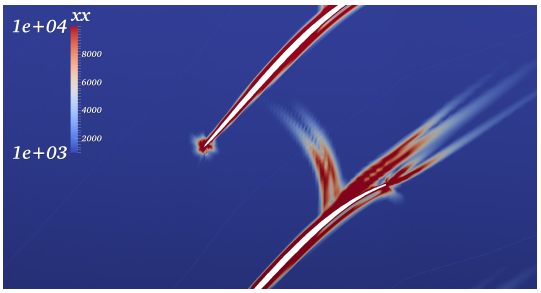


(F) 26^{th} adaptation step zz-component.

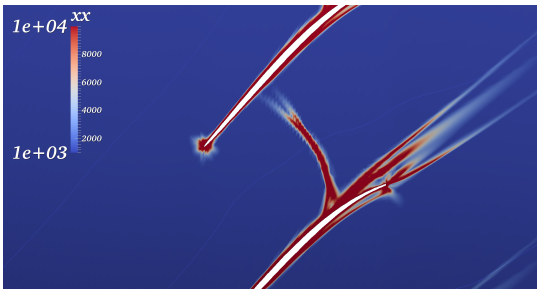
FIGURE D.3: Comparison of the diagonal terms of the *Hessian* matrix of relative *Mach* number at mid-span of the starting mesh after positive-semi-definite reconstruction.

has also reduced in size, showing its shrinking in the flow field.

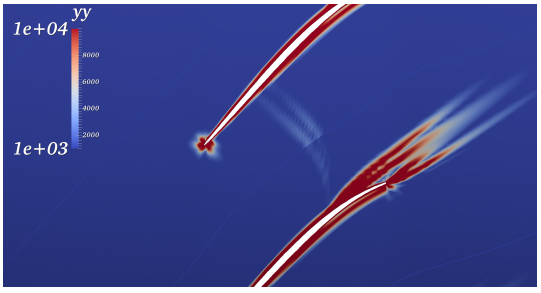
A similar overall performance can be noted at 75% span, where the shock is present in all components of the starting mesh error, albeit with very poor intensity. Additionally, its structure seems to comprise of two elements, however repeated adaptation has correctly managed to merge them into a single, thin discontinuity. The other region of high derivatives, is again the wake. Little change has occurred with hardly any improvement, with all diagonal terms managing to capture it well even on the staring grid.



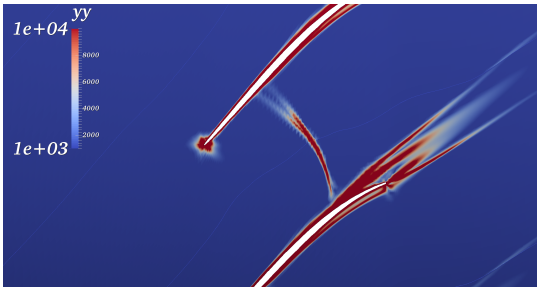
(A) Starting mesh xx-component.



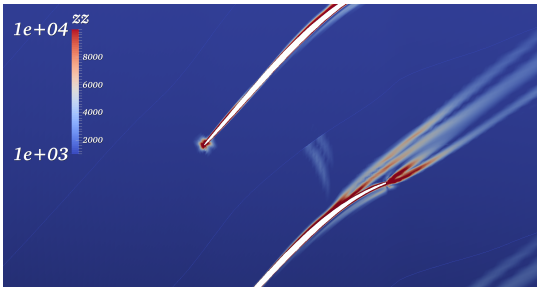
(B) 26^{th} adaptation step xx-component.



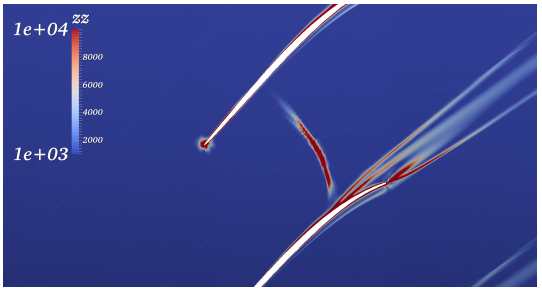
(C) Starting mesh yy-component.



(D) 26^{th} adaptation step yy-component.



(E) Starting mesh zz-component.



(F) 26^{th} adaptation step zz-component.

FIGURE D.4: Comparison of the diagonal terms of the *Hessian* matrix at 75% span of the starting mesh after positive-semi-definite reconstruction.

Bibliography

[1] F. T. Johnson, E. N. Tinoco, and N. J. Yu. Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle. *Computers & Fluids*, 34(10):1115–1151, 2005.

[2] F. D. Witherden and A. Jameson. Future Directions in Computational Fluid Dynamics. In *23rd AIAA Computational Fluid Dynamics Conference*, page 3791, 2017.

[3] G. M. Laskowski, J. Kopriva, V. Michelassi, S. Shankaran, U. Paliath, R. Bhaskaran, Q. Wang, C. Talnikar, Z. J. Wang, and F. Jia. Future Directions of High Fidelity CFD for Aerothermal Turbomachinery Analysis and Design. In *46th AIAA Fluid Dynamics Conference*, page 3322, 2016.

[4] C. J. Fischberg, C. M. Rhie, R. M. Zacharias, P. C. Bradley, and T. M. DesSureault. Using hundreds of workstations for production running of parallel CFD applications. In *Parallel Computational Fluid Dynamics 1995*, pages 9–22. Elsevier, 1996.

[5] J. Slotnick, A. Khodadoust, J. J. Alonso, D. L. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: a path to revolutionary computational aerosciences. 2014.

[6] F. Alauzet and A. Loseille. A decade of progress on anisotropic mesh adaptation for computational fluid dynamics. *Computer-Aided Design*, 72:13–39, 2016.

[7] K. J. Fidkowski and D. L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*, 49(4): 673–694, 2011.

[8] M. A. Park, J. A. Krakos, T. Michal, A. Loseille, and J. J. Alonso. Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Toward CFD Vision 2030. 2016.

[9] D. A. Venditti and D. L. Darmofal. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–69, 2002.

[10] R. Balasubramanian and J. C. Newman. Comparison of adjoint-based and feature-based grid adaptation for functional outputs. *International Journal for Numerical Methods in Fluids*, 53(10):1541–1569, 2007.

[11] R. Löhner. Mesh adaptation in fluid mechanics. *Engineering Fracture Mechanics*, 50(5-6):819–847, 1995.

[12] C. Roy. Strategies for driving mesh adaptation in CFD. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 1302, 2009.

[13] C. Roy. Review of discretization error estimators in scientific computing. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 126, 2010.

[14] K. J. Fidkowski. Output Based Error Estimation and Mesh Adaptation for Steady and Unsteady Flow Problems. In H. Deconinck and T. Horvath, editors, *38th Advanced CFD Lectures Series; von Karman Institute for Fluid Dynamics (September 14–16, 2015)*. von Karman Institute for Fluid Dynamics, 2015.

[15] T. J. Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(3-4):243–273, 1997.

[16] T. J. Baker. Mesh generation: Art or Science? *Progress in Aerospace Sciences*, 41(1):29–63, 2005.

[17] W. G. Habashi, J. Dompierre, Y. Bourgault, M. Fortin, and M. G. Vallet. Certifiable computational fluid dynamics through mesh optimization. *AIAA Journal*, 36(5):703–711, 1998.

[18] M. Nemec, M. Aftosmis, and M. Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, page 725, 2008.

[19] D. A. Venditti and D. L. Darmofal. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22–46, 2003.

[20] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *International Journal for Numerical Methods in Fluids*, 25(4):475–491, 1997.

[21] N. Qin and X. Liu. Flow feature aligned grid adaptation. *International Journal for Numerical Methods in Engineering*, 67(6):787–814, 2006.

[22] M. J. Harris and N. Qin. Using the medial axis to represent flow features for feature-aligned unstructured quad-dominant mesh generation. *Computers & Fluids*, 102:1–14, 2014.

[23] M. J. Harris and N. Qin. Geometric representation of flow features using the medial axis for mesh generation. *AIAA Journal*, 53(1):246–259, 2014.

[24] F. Alauzet. Metric Based Anisotropic Mesh Adaptation, June 2010.

[25] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72(2): 449–466, 1987.

[26] R. Löhner. Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing. *Computing Systems in Engineering*, 1(2-4): 257–272, 1990.

[27] J. Peraire, J. Peiro, and K. Morgan. Adaptive remeshing for three-dimensional compressible flow computations. *Journal of Computational Physics*, 103(2): 269–285, 1992.

[28] P. L. George, F. Hecht, and M. G. Vallet. Creation of internal points in Voronoi's type method. Control adaptation. *Advances in Engineering Software and Workstations*, 13(5-6):303–312, 1991.

[29] D. Ait-Ali-Yahia, W. G. Habashi, A. Tam, M. G. Vallet, and M. Fortin. A Directionally Adaptive Methodology Using an Edge-Based Error Estimate on Quadrilateral Grids. *International Journal for Numerical Methods in Fluids*, 23 (7):673–690, 1996.

[30] M. Fortin. Error Estimation and Directionally-Adaptive Meshing. 1994.

[31] W. G. Habashi, M. Fortin, J. Dompierre, M. G. Vallet, and Y. Bourgault. Anisotropic mesh adaptation: a step towards a mesh-independent and user-independent CFD. In *Barriers and Challenges in Computational Fluid Dynamics*, pages 99–117. Springer, 1998.

[32] M. Robichaud, D. Ait-Ali-Yahia, M. Peeters, G. Baruzzi, V. Kozel, and W. Habashi. 3-D anisotropic adaptation for external and turbomachinery flows on hybrid unstructured grids. In *Fluids 2000 Conference and Exhibit, Denver, CO*, 2000.

[33] W. G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin, and M. G. Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles. *International Journal for Numerical Methods in Fluids*, 32(6):725–744, 2000.

[34] D. Ait-Ali-Yahia, G. Baruzzi, W. G. Habashi, M. Fortin, J. Dompierre, and M. G. Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part II. Structured grids. *International Journal for Numerical Methods in Fluids*, 39(8):657–673, 2002.

[35] F. Suerich-Gulick, C. Lepage, and W. G. Habashi. Anisotropic 3-D Mesh Adaptation for Turbulent Flows. In *34th AIAA Fluid Dynamics Conference and Exhibit*, page 2533, 2004.

[36] O. Sahni, K. E. Jansen, M. S. Shephard, C. A. Taylor, and M. W. Beall. Adaptive boundary layer meshing for viscous flow simulations. *Engineering with Computers*, 24(3):267, 2008.

[37] K. C. Chitale. *Anisotropic Mesh Adaptivity for Turbulent Flows with Boundary Layers*. PhD thesis, University of Colorado, 2013.

[38] K. F. Tchon, J. Dompierre, and R. Camarero. Conformal refinement of all-quadrilateral and all-hexahedral meshes according to an anisotropic metric. In *11th International Meshing Roundtable*. Citeseer, 2002.

[39] K. F. Tchon, J. Dompierre, and R. Scamarero. Automated refinement of conformal quadrilateral and hexahedral meshes. *International Journal for Numerical Methods in Engineering*, 59(12):1539–1562, 2004.

[40] S. A. Mitchell and T. J. Tautges. Pillowing doublets: refining a mesh to ensure that faces share at most one edge. In *4th International Meshing Roundtable*, pages 231–240. Citeseer, 1995.

[41] R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers*, 12(3-4):168–177, 1996.

[42] M. J. Harris. *Flow feature aligned mesh generation and adaptation*. PhD thesis, University of Sheffield, 2013.

[43] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49): 5068–5082, 2005.

[44] A. Loseille, A. Dervieux, P. Frey, and F. Alauzet. Achievement of global second order mesh convergence for discontinuous flows with adapted unstructured meshes. In *18th AIAA Computational Fluid Dynamics Conference*, page 4186, 2007.

[45] Y. Mesri, F. Alauzet, A. Loseille, L. Hascoët, B. Koobus, and A. Dervieux. Continuous mesh adaptation models for CFD. *CFD Journal*, 16(4):346–355, 2008.

[46] A. Loseille and F. Alauzet. Continuous mesh framework part I: well-posed continuous interpolation error. *SIAM Journal on Numerical Analysis*, 49(1): 38–60, 2011.

[47] A. Loseille and F. Alauzet. Continuous mesh framework part II: validations and applications. *SIAM Journal on Numerical Analysis*, 49(1):61–86, 2011.

[48] L. Frazza, A. Loseille, and F. Alauzet. Anisotropic mesh adaptation for turbomachinery applications. In *23rd AIAA Computational Fluid Dynamics Conference*, page 3299, 2017.

[49] M. G. Vallet, C. M. Manole, J. Dompierre, S. Dufour, and F. Guibault. Numerical comparison of some Hessian recovery techniques. *International Journal for Numerical Methods in Engineering*, 72(8):987–1007, 2007.

[50] M. Picasso, F. Alauzet, H. Borouchaki, and P. L. George. A numerical study of some Hessian recovery techniques on isotropic and anisotropic meshes. *SIAM Journal on Scientific Computing*, 33(3):1058–1076, 2011.

[51] H. Borouchaki, F. Hecht, and P. J. Frey. Mesh gradation control. *International Journal for Numerical Methods in Engineering*, 43(6):1143–1165, 1998.

[52] X. Li, J. F. Remacle, N. Chevaugeon, and M. S. Shephard. Anisotropic Mesh Gradation Control. In *International Meshing Roundtable*, pages 401–412, 2004.

[53] F. Alauzet. Size gradation control of anisotropic meshes. *Finite Elements in Analysis and Design*, 46(1-2):181–202, 2010.

[54] E. Joubarne and F. Guibault. 3D Metric-based anisotropic mesh adaptation for vortex capture. *Mathematics and Computers in Simulation*, 82(1):163–180, 2011.

[55] R. Becker and R. Rannacher. Weighted a posteriori error control in finite element methods. Technical report, 1994.

[56] M. B. Giles, M. Larson, M. Levenstam, and E. Suli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. Technical report, University of Oxford, 1997.

[57] D. A. Venditti and D. L. Darmofal. A multilevel error estimation and grid adaptive strategy for improving the accuracy of integral outputs. In *14th Computational Fluid Dynamics Conference*, page 3292, 1999.

[58] M. B. Giles. On adjoint equations for error analysis and optimal grid adaptation in CFD. In *Frontiers of Computational Fluid Dynamics 1998*, pages 155–169. World Scientific, 1998.

[59] N. A. Pierce and M. B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247–264, 2000.

[60] J. D. Müller and M. Giles. Solution adaptive mesh refinement using adjoint error analysis. In *15th AIAA Computational Fluid Dynamics Conference*, page 2550, 2001.

[61] D. A. Venditti and D. L. Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164(1):204–227, 2000.

[62] P. W. Power, M. D. Piggott, F. Fang, G. J. Gorman, C. C. Pain, D. P. Marshall, A. J. H. Goddard, and I. M. Navon. Adjoint goal-based error norms for adaptive mesh ocean modelling. *Ocean Modelling*, 15(1-2):3–38, 2006.

[63] P. W. Power, C. C. Pain, M. D. Piggott, F. Fang, G. J. Gorman, A. P. Umpleby, A. J. H. Goddard, and I. M. Navon. Adjoint a posteriori error measures for anisotropic mesh optimisation. *Computers & Mathematics with Applications*, 52 (8-9):1213–1242, 2006.

[64] D. A. Venditti. *Grid adaptation for functional outputs of compressible flow simulations*. PhD thesis, Massachusetts Institute of Technology, 2002.

[65] M. A. Park. Adjoint-based, three-dimensional error prediction and grid adaptation. *AIAA Journal*, 42(9):1854–1862, 2004.

[66] M. Park. Three-dimensional turbulent RANS adjoint-based error correction. In *16th AIAA Computational Fluid Dynamics Conference*, page 3849, 2003.

[67] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. The Netherlands: Elsvier; ISBN 0080430090, 2001.

[68] M. Nemec and M. Aftosmis. Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes. In *18th AIAA Computational Fluid Dynamics Conference*, page 4187, 2007.

[69] T. Barth and D. Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*, page 366, 1989.

[70] M. Aftosmis and M. Berger. Multilevel error estimation and adaptive h-refinement for cartesian meshes with embedded boundaries. In *40th AIAA Aerospace Sciences Meeting & Exhibit*, page 863, 2002.

[71] E. Lee-Rausch, M. Park, W. Jones, D. Hammond, and E. Nielsen. Application of parallel adjoint-based error estimation and anisotropic grid adaptation for three-dimensional aerospace configurations. In *23rd AIAA Applied Aerodynamics Conference*, page 4842, 2005.

[72] H. J. Kim, Y. Takano, and K. Nakahashi. Error estimation and grid adaptation using Euler adjoint method. *Journal of Aircraft*, 43(5):1317–1324, 2006.

[73] H. J. Kim and K. Nakahashi. Output-based error estimation and adaptive mesh refinement using viscous adjoint method. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 1395, 2006.

[74] W. Jones, E. Nielsen, and M. Park. Validation of 3D adjoint based error estimation and mesh adaptation for sonic boom prediction. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 1150, 2006.

[75] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[76] J. Ponsin, A. Caloto, E. Andrés, P. Bitrián, and C. Lozano. Implementation of an Adjoint-Based Error Estimation and Grid Adaptation Module in the DLR TAU Code. ICAS, 2010.

[77] K. Ding, K. J. Fidkowski, and P. L. Roe. Acceleration Techniques for Adjoint-Based Error Estimation and Mesh Adaptation. In *Eighth International Conference on Computational Fluid Dynamics (ICCFD8) ICCFD8-2014-0249*, 2014.

[78] K. J. Fidkowski. High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics. In H. Deconinck and R. Abgrall, editors, *37th Advanced CFD Lectures series; von Karman Institute for Fluid Dynamics (December 9–12 2013)*. von Karman Institute for Fluid Dynamics, 2013.

[79] T. A. Eymann. *Active Flux Schemes*. PhD thesis, University of Michigan, 2013.

[80] N. A. Pierce and M. B. Giles. Adjoint and defect error bounding and correction for functional estimates. *Journal of Computational Physics*, 200(2):769–794, 2004.

[81] K. Duraisamy, J. J. Alonso, P Chandrasekhar, and F. Palacios. Error estimation for high speed flows using continuous and discrete adjoints. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 128, 2010.

[82] F. Palacios, K. Duraisamy, J. J. Alonso, and E. Zuazua. Robust grid adaptation for efficient uncertainty quantification. *AIAA Journal*, 50(7):1538–1546, 2012.

[83] B. A. Rothacker, M. Ceze, and K. Fidkowski. Adjoint-based error estimation and mesh adaptation for problems with output constraints. In *32nd AIAA Applied Aerodynamics Conference*, page 2576, 2014.

[84] R. Hartmann. Multitarget error estimation and adaptivity in aerodynamic flow simulations. *SIAM Journal on Scientific Computing*, 31(1):708–731, 2008.

[85] R. Hartmann, J. Held, and T. Leicht. Adjoint-based error estimation and adaptive mesh refinement for the RANS and k–ω turbulence model equations. *Journal of Computational Physics*, 230(11):4268–4284, 2011.

[86] R. P. Dwight. Goal-oriented mesh adaptation for finite volume methods using a dissipation-based error indicator. *International Journal for Numerical Methods in Fluids*, 56(8):1193–1200, 2008.

[87] R. P. Dwight. Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation. *Journal of Computational Physics*, 227(5):2845–2863, 2008.

[88] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *14th Fluid and Plasma Dynamics Conference*, page 1259, 1981.

[89] K. F. Fidkowski and P. L. Roe. Entropy-based mesh refinement, i: the entropy adjoint approach. In *19th AIAA Computational Fluid Dynamics*, page 3790. 2009.

[90] N. Yamaleev, B. Diskin, and K. Pathak. Error minimization via adjoint-based anisotropic grid adaptation. In *40th Fluid Dynamics Conference and Exhibit*, page 4436, 2010.

[91] B. Diskin and N. Yamaleev. Grid Adaptation using Adjoint-Based Error Minimization. In *20th AIAA Computational Fluid Dynamics Conference*, page 3986, 2011.

[92] J. Peter, M. Nguyen-Dinh, and P. Trontin. Goal oriented mesh adaptation using total derivative of aerodynamic functions with respect to mesh coordinates. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 158, 2012.

[93] J. Peter, M. Nguyen-Dinh, and P. Trontin. Goal oriented mesh adaptation using total derivative of aerodynamic functions with respect to mesh coordinates–With applications to Euler flows. *Computers & Fluids*, 66:194–214, 2012.

[94] M. Nguyen-Dinh, J. Peter, R. Sauvage, M. Meaux, and J. A. Désidéri. Mesh quality assessment based on aerodynamic functional output total derivatives. *European Journal of Mechanics-B/Fluids*, 45:51–71, 2014.

[95] J. E. Peter and J. A. Desideri. Unstructured mesh adaptation for functional outputs. with application to two dimensionnal inviscid flows. In *54th AIAA Aerospace Sciences Meeting*, page 1930, 2016.

[96] G. Todarello, F. Vonck, S. Bourasseau, J. Peter, and J. A. Désidéri. Finite-volume goal-oriented mesh adaptation for aerodynamics using functional derivative with respect to nodal coordinates. *Journal of Computational Physics*, 313:799–819, 2016.

[97] A. Resmini, J. Peter, and D. Lucor. Mono-block and non-matching multi-block structured mesh adaptation based on aerodynamic functional total derivatives for RANS flow. *International Journal for Numerical Methods in Fluids*, 83(11): 866–884, 2017.

[98] G. Vivarelli, N. Qin, and S. Shahpar. Combined Hessian and Adjoint Error-Based Anisotropic Mesh Adaptation for Turbomachinery Flows. In *55th AIAA Aerospace Sciences Meeting*, page 1946, 2017.

[99] G. Vivarelli, N. Qin, S. Shahpar, and D. Radford. Efficient Adjoint-Based Mesh Adaptation Applied to Turbo-Machinery Flows. In *ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition*.

[100] T. Zervogiannis, P. I. K. D. Liakopoulos, D. I. Papadimitriou, and K. C. Giannakoglou. A Grid Enrichment and Movement Strategy for A Posteriori Error Analysis in Viscous Flows. In *7th European Conference on Turbomachinery, Fluid Dynamics and Thermodynamics, Athens*, 2007.

[101] P. I. K. D. Liakopoulos and K. C. Giannakoglou. Unstructured remeshing using an efficient smoothing scheme approach. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee,*

The Netherlands, September 5-8, 2006. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.

[102] M. Gugala, M. Meyer, and J. D. Muller. Towards an output-Based Re-Meshing for Turbomachinery Applications. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, 2016.

[103] L. Lapworth. Hydra-CFD: a framework for collaborative CFD development. In *International Conference on Scientific and Engineering Computation (IC-SEC), Singapore*, volume 30, June 2004.

[104] F. Fraysse, E. Valero, and J. Ponsín. Comparison of mesh adaptation using the adjoint methodology and truncation error estimates. *AIAA Journal*, 50(9): 1920–1932, 2012.

[105] Y. Kallinderis and C. Kontzialis. A priori mesh quality estimation via direct relation between truncation error and mesh distortion. *Journal of Computational Physics*, 228(3):881–902, 2009.

[106] Y. Kallinderis and S. Fotia. A priori mesh quality metrics for three-dimensional hybrid grids. *Journal of Computational Physics*, 280:465–488, 2015.

[107] A. Brandt and O. E. Livne. *Multigrid techniques: 1984 guide with applications to fluid dynamics*, volume 67. SIAM, 2011.

[108] F. Fraysse. *Numerical Error Prediction and its Applications in CFD using τ-estimation*. PhD thesis, Universidad Politécnica de Madrid, 2012.

[109] J. M. Derlaga. *Application of Improved Truncation Error Estimation Techniques to Adjoint Based Error Estimation and Grid Adaptation*. PhD thesis, Virginia Tech, 2015.

[110] T. Tang. Moving mesh methods for computational fluid dynamics. *Contemporary mathematics*, 383:141–174, 2005.

[111] C. J. Budd, W. Huang, and R. D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18:111–241, 2009.

[112] M. J. Baines. Grid adaptation via node movement. *Applied Numerical Mathematics*, 26(1-2):77–96, 1998.

[113] M. M. Selim and R. P. Koomullil. Mesh deformation approaches–a survey. *Journal of Physical Mathematics*, 7(2), 2016.

[114] R. Löhner and J. D. Baum. Adaptive h-refinement on 3D unstructured grids for transient problems. *International Journal for Numerical Methods in Fluids*, 14 (12):1407–1419, 1992.

[115] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *Mathematics of Computation of the American Mathematical Society*, 65(215):1183–1200, 1996.

[116] R. Biswas and R. C. Strawn. Mesh quality control for multiply-refined tetrahedral grids. *Applied Numerical Mathematics*, 20(4):337–348, 1996.

[117] R. Biswas and R. C. Strawn. Tetrahedral and hexahedral mesh adaptation for CFD problems. *Applied Numerical Mathematics*, 26(1-2):135–151, 1998.

[118] D. J. Mavriplis. Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes. *International Journal for Numerical Methods in Fluids*, 34(2):93–111, 2000.

[119] N. J. Harris, S. E. Benzley, and S. J. Owen. Conformal Refinement of All-Hexahedral Element Meshes Based on Multiple Twist Plane Insertion. In *International Meshing Roundtable*, pages 157–168, 2004.

[120] S. E. Benzley, N. J. Harris, M. Scott, M. Borden, and S. J. Owen. Conformal refinement and coarsening of unstructured hexahedral meshes. *Journal of Computing and Information Science in Engineering*, 5(4):330–337, 2005.

[121] M. Parrish, M. Borden, M. Staten, and S. Benzley. A selective approach to conformal refinement of unstructured hexahedral finite element meshes. In *Proceedings of the 16th international meshing roundtable*, pages 251–268. Springer, 2008.

[122] G. Nicolas, T. Fouquet, S. Geniaut, and S. Cuvilliez. Improved adaptive mesh refinement for conformal hexahedral meshes. *Advances in Engineering Software*, 102:14–28, 2016.

[123] S. J. Owen and R. M. Shih. A Template-Based Approach for Parallel Hexahedral Two-Refinement. *Procedia Engineering*, 124:31–43, 2015.

[124] Y. Ito, A. M. Shih, R. P. Koomullil, and B. K. Soni. A solution-based adaptive redistribution method for unstructured meshes. In *Proceedings of the 15th International Meshing Roundtable*, pages 147–161. Springer, 2006.

[125] A. Loseille. Unstructured Mesh Generation and Adaptation. In *Handbook of Numerical Analysis*, volume 18, pages 263–302. Elsevier, 2017.

[126] A. Milli and S. Shahpar. PADRAM: PArametric Design and RApid Meshing system for complex turbomachinery configurations. In *ASME Turbo Expo 2012: Turbine Technical Conference and Exposition*, pages 2135–2148. American Society of Mechanical Engineers, 2012.

[127] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH Computer Graphics*, 20(4):151–160, 1986.

[128] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[129] Hirarchical Datal Format. `https://support.hdfgroup.org/HDF5/`. Accessed: 29-05-2018.

[130] CFD General Notation System. `https://cgns.github.io`. Accessed: 29-05-2018.

[131] P. Moinier. *Algorithm developments for an unstructured viscous flow solver*. PhD thesis, Oxford University, 1999.

[132] P. I. Crumpton, P. Moinier, and M. B. Giles. An unstructured algorithm for high Reynolds number flows on highly stretched grids. *Numerical Methods in Laminar and Turbulent Flow*, pages 561–572, 1997.

[133] J. D. Müller and M. B. Giles. Edge-based multigrid schemes for hybrid grids. In *Numerical Methods for Fluid Dynamics, VI, ICFD*. Citeseer, 1998.

[134] J. D. Müller. Coarsening 3-D Hybrid Meshes for Multigrid Methods. In *In 9th Copper Mountain Multigrid Conference*. Citeseer, 1999.

[135] P. I. Crumpton and M. B. Giles. *OPLUS Programmer's Manual*, 1993.

[136] D. A. Burgess, P. I. Crumpton, and M. B. Giles. A parallel framework for unstructured grid solvers. In *Programming Environments for Massively Parallel Distributed Systems*, pages 97–106. Springer, 1994.

[137] P. Crumpton and M. Giles. Aircraft computations using multigrid and an unstructured parallel library. In *33rd Aerospace Sciences Meeting and Exhibit*, page 210, 1995.

[138] I. Tristanto. Private communications, 2015-2018.

[139] R. C. Swanson and E. Turkel. On central-difference and upwind schemes. *Journal of Computational Physics*, 101(2):292–306, 1992.

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

[141] S. Allmaras. Analysis of semi-implicit preconditioners for multigrid solution of the 2-D compressible Navier-Stokes equations. In *12th Computational Fluid Dynamics Conference*, page 1651, 1995.

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

[143] P. K. Sweby. Godunov methods. In *Godunov Methods*, pages 879–898. Springer, 2001.

[144] B. van Leer. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 14(4):361–370, 1974.

[145] C. Hirsch. *Numerical computation of internal and external flows: volume 2: computational methods for inviscid and viscous flows*. Chichester: John Wiley & Sons, 2002.

[146] H. Luo, J. D. Baum, and R. Lohner. Edge-based finite element scheme for the Euler equations. *AIAA Journal*, 32(6):1183–1190, 1994.

[147] R. C. Swanson, R. Radespiel, and E. Turkel. On some numerical dissipation schemes. *Journal of Computational Physics*, 147(2):518–544, 1998.

[148] B. van Leer, W. T. Lee, and K. G. Powell. Sonic-point capturing. 1989.

[149] A. Haselbacher and J. Blazek. Accurate and efficient discretization of Navier-Stokes equations on mixed grids. *AIAA Journal*, 38(11):2094–2102, 2000.

[150] A. Syrakos, S. Varchanis, Y. Dimakopoulos, A. Goulas, and J. Tsamopoulos. On the order of accuracy of the divergence theorem (Green-Gauss) method for calculating the gradient in finite volume methods. 2017.

[151] P. I. Crumpton. An efficient cell vertex method for unstructured tetrahedral grids. *Numerical Methods for Fluid Dynamics V*, 5:369, 1995.

[152] A. Jameson. Origins and further development of the Jameson–Schmidt–Turkel scheme. *AIAA Journal*, pages 1–23, 2017.

[153] L. Martinelli. *Calculations of viscous flows with a multigrid method*. PhD thesis, 1987.

[154] W. Hackbusch. *Multi-grid methods and applications*, volume 4. 1985.

[155] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.

[156] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(3):357–393, 1983.

[157] S. Allmaras. Analysis of a local matrix preconditioner for the 2-D Navier-Stokes equations. In *11th Computational Fluid Dynamics Conference*, page 3330, 1993.

[158] N. A. Pierce and M. B. Giles. Preconditioning on stretched meshes. Technical report, University of Oxford, 1995.

[159] B. van Leer, W. T. Lee, and P. L. Roe. Characteristic time-stepping or local preconditioning of the Euler equations. 1991.

[160] C. F. Ollivier-Gooch. Towards Problem-Independent Multigrid Convergence Rates for Unstructured Mesh Methods I: Inviscid and Laminar Viscous Flows. In *Proceedings of the Sixth International Symposium on Computational Fluid Dynamics*. Citeseer, 1995.

[161] N. Pierce and M. Giles. Preconditioning compressible flow calculations on stretched meshes. In *34th Aerospace Sciences Meeting and Exhibit*, page 889, 1996.

[162] N. A. Pierce. *Preconditioned Multigrid Methods for Compressible Flow Calculations on Stretched Meshes*. PhD thesis, Oxford University, 1997.

[163] J. L. Lions. *Optimal control of systems governed by partial differential equations (Grundlehren der Mathematischen Wissenschaften)*, volume 170. Springer Berlin, 1971.

[164] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97–110, 1974.

[165] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[166] A. Jameson. Optimum aerodynamic design using CFD and control theory. In *12th Computational Fluid Dynamics Conference*, page 1729, 1995.

[167] A. Jameson. Optimum aerodynamic design using control theory. *Computational Fluid Dynamics Review*, 3:495–528, 1995.

[168] J. J. Alonso. Introduction to Sensitivity Analysis and the Adjoint Method: Derivations, Uses and Properties. In H. Deconinck and T. Horvath, editors, *38th Advanced CFD Lectures Series; von Karman Institute for Fluid Dynamics (September 14–16, 2015)*. von Karman Institute for Fluid Dynamics, 2015.

[169] M. B. Giles and N. Pierce. Adjoint equations in CFD-duality, boundary conditions and solution behaviour. In *13th Computational Fluid Dynamics Conference*, page 1850, 1997.

[170] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4):393–415, 2000.

[171] P. Cusdin and J. D. Müller. Deriving linear and adjoint codes for CFD using Automatic Differentiation. *School of Aeronautical Engineering, Queen's University Belfast*, 2003.

[172] M. H. Hekmat and M. Mirzaei. A comparison of the continuous and discrete adjoint approach extended based on the standard Lattice Boltzmann method in flow field inverse optimization problems. *Acta Mechanica*, 227(4):1025–1050, 2016.

[173] M. D. Greenberg. *Applications of Green's functions in science and engineering*. Courier Dover Publications, 2015.

[174] M. B. Giles. On the use of Runge-Kutta time-marching and multigrid for the solution of steady adjoint equations. Technical report, University of Oxford, 2000.

[175] M. Giles, M. Duta, and J. D. Müller. Adjoint code developments using the exact discrete approach. In *15th AIAA Computational Fluid Dynamics Conference*, page 2596, 2001.

[176] F. G. Cervera. *Resolution of the adjoint Navier-Stokes equations using a preconditioned multigrid method*. PhD thesis, Universidad Politécnica de Madrid, 2007.

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

[178] M. C. Duta, M. B. Giles, and M. S. Campobasso. The harmonic adjoint approach to unsteady turbomachinery design. *International Journal for Numerical Methods in Fluids*, 40(3-4):323–332, 2002.

[179] E. J. Nielsen and M. A. Park. Using an adjoint approach to eliminate mesh sensitivities in computational design. *AIAA Journal*, 44(5):948–953, 2006.

[180] E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. *AIAA Journal*, 40(6):1155–1163, 2002.

[181] D. Radford. Private communications, 2015-2018.

[182] INRIA. TAPENADE. `http://www-sop.inria.fr/tropics/tapenade.html`. Accessed: 15-08-2018.

[183] T. Barth. A 3-D upwind Euler solver for unstructured meshes. In *10th Computational Fluid Dynamics Conference*, page 1548, 1991.

[184] W. K. Anderson and D. L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. 1994.

[185] W. Gander. Algorithms for the QR decomposition. *Res. Rep*, 80(02):1251–1268, 1980.

[186] MATLAB® QR Factorisation. `https://uk.mathworks.com/help/symbolic/qr.html`. Accessed: 12-07-2018.

[187] T. J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[188] R. Löhner. Regridding surface triangulations. *Journal of Computational Physics*, 126(1):1–10, 1996.

[189] J. T. Batina. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388, 1990.

[190] R. V. Garimella, M. J. Shashkov, and P. M. Knupp. Triangular and quadrilateral surface mesh quality optimization using local parametrization. *Computer Methods in Applied Mechanics and Engineering*, 193(9-11):913–928, 2004.

[191] J.D. Denton. Lessons from Rotor 37. *Journal of Thermal Science*, 6(1):1–13, 1997.

[192] J. Dunham. CFD validation for propulsion system components (la validation CFD des organes des propulseurs). Technical report, Advisory Group for Aerodspace Research and Development Neuilly-Sur-Seine (France), 1998.

[193] A. Ameri. NASA Rotor 37 CFD Code Validation Glenn-HT code. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 1060, 2009.

[194] K. L. Suder. Experimental investigation of the flow field in a transonic, axial flow compressor with respect to the development of blockage and loss, 1996.

[195] A. Shabbir, M. L. Celestina, J. J. Adamczyk, and A. J. Strazisar. The effect of hub leakage flow on two high speed axial flow compressor rotors. In *ASME 1997 International Gas Turbine and Aeroengine Congress and Exhibition.*

[196] C. Hah. Large Eddy Simulation of Transonic Flow Field in NASA Rotor 37. In *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 1061, 2009.

[197] K. L. Suder, R. V. Chima, A. J. Strazisar, and W. B. Roberts. The Effect of Adding Roughness and Thickness to a Transonic Axial Compressor Rotor. *Journal of Turbomachinery*, 117(4):491–505, 1995.

[198] *The HYDRA User's Guide Version v*7.0.12, 30^{th} June 2014.

[199] R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474, 1998.