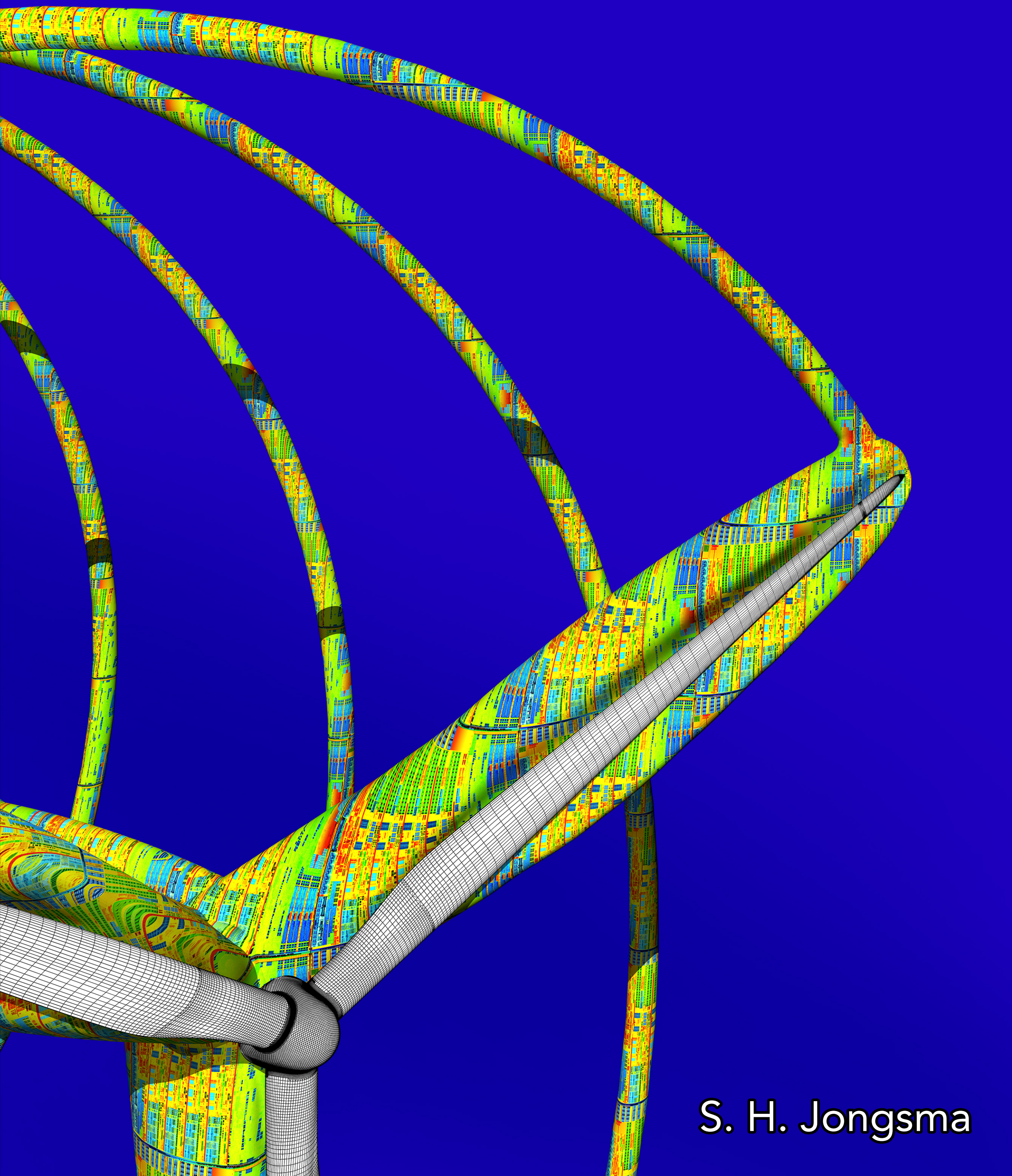
On a Method for Simulation-Based Wind Turbine Blade Design



S. H. Jongsma

On a Method for Simulation-Based Wind Turbine Blade Design

S. H. Jongsma

On a Method for Simulation-Based Wind Turbine Blade Design
S. H. Jongsma

Thesis University of Twente, Enschede, The Netherlands
July 2014
ISBN: 978-90-365-3698-1
DOI: 10.3990/1.9789036536981
`http://dx.doi.org/10.3990/1.9789036536981`

Cover design: Kay Jongsma

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0
International Public License, S. H. Jongsma (2014)

ON A METHOD FOR SIMULATION-BASED WIND TURBINE BLADE DESIGN

PROEFSCHRIFT

ter verkrijgen van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 11 juli 2014 om 12:45 uur

door

Sietse Harmen Jongsma

geboren op 15 april 1985
te Doetinchem, Nederland

Dit proefschrift is goedgekeurd door de promotor:
prof. dr. ir. H. W. M. Hoeijmakers

en de co-promotor:
dr. ir. E. T. A. van der Weide

SUMMARY

WIND turbines are an important means for the production of renewable energy. This energy, commonly referred to as wind energy, is produced by extracting the kinetic energy from the wind and to subsequently convert it, usually via a rotational motion, to electrical energy. Wind conditions vary from one site to another and the design of a horizontal axis wind turbine depends on these local wind conditions. One of the important aspects of the design of a wind turbine concerns the aerodynamic shape of the rotor blades. The research presented in this thesis focusses on the development of a computational method that can be used for aerodynamic shape optimization.

Other aspects — apart from the aerodynamic shape of a wind turbine rotor blade — are also important in the design of a wind turbine, for instance the structural design of the blade. To take this multidisciplinary character of wind turbine design into account, one of the aims of the research presented in this thesis has been that the aerodynamic shape optimization method seamlessly fits into the design process. Moreover, the optimization method is required to be efficient and fully automatic.

A suitable method for the parametrization of the wind turbine blade geometry has been investigated. This investigation lead to the choice to use a non-uniform rational basis spline (NURBS) surface for representing the aerodynamic shape of the wind turbine blade. Such a description of the parametrization of the geometry is compatible with computer aided design methods, which eases the incorporation of the aerodynamic shape optimization method in the multidisciplinary design process. A model optimization problem has been considered to determine the relation between the number of design variables that is required to sufficiently accurately represent the design space for solving an aerodynamic shape optimization problem to the number of design variables that is required to realize an accurate NURBS representation of an aerodynamic shape. For the purpose of obtaining a NURBS surface representation of a wind turbine rotor blade, a fitting procedure has been developed. The accuracy of the resulting fit for two representative wind turbine blades is presented. It has been found that sufficient accuracy can be achieved with a NURBS surface described by 13 chordwise sections with 13 control points each.

In the present research, objective functions are considered for which the evaluation requires the solution of partial differential equations that govern flow: the Euler equations. Since only for very particular flow configurations an analytical solution of these equations can be found, an approximate solution is computed by solving the discretized equations. Solving the discretized Euler equations requires the discretization of the whole flow domain. During the optimization procedure, the geometry of the rotor blade adapts, requiring a discrete representation of the new flow domain. To accommodate the change in shape during the optimization, a hyperbolic grid generation method has been implemented, which provides a high quality grid in the near-field region of the blade throughout the optimization process.

In conjunction with the hyperbolic grid generation method, for the remainder of the computational domain additional grid blocks are used that are allowed to overlap. This approach — commonly referred to as a composite overset grid strategy — requires that the flow solution is transferred between grid blocks in the regions of overlap. To determine the so-called connectivity information, required for transferring the flow solution, an implicit hole cutting method is used.

A solution to the Euler equations for steady flow — for a flow domain, discretized by means of a structured composite overset grid — is obtained, using a cell-centred second-order finite volume method. Steady-state solutions are computed using Newton's method. The initial guess required for Newton's method is obtained by performing a series of explicit Runge-Kutta pseudo-time steps. The order of accuracy of the numerical method is verified, furthermore the numerical method is validated against experimental results.

For the aerodynamic shape optimization method a gradient-based approach is adopted to allow for the use of a large number of design variables, while keeping the number of times the objective function must be evaluated limited. The discrete adjoint equation method is used to efficiently compute the total derivative of the objective function with respect to all design variables. Dual number arithmetic is employed to compute partial derivatives. By using template functions and graph vertex colouring a method is realized that can be used for efficiently computing partial derivatives that are fully consistent with the associated function. Moreover, this approach eliminates the need for creating new functions for computing these partial derivatives. Consistency of the derivatives, computed by means of the adjoint equation method, has been demonstrated by comparison with the derivatives computed using a different method.

An optimization framework has been developed by coupling the different components implemented in the present research to a gradient-based optimizer. Subsequently, two different optimization problems have been considered. One problem involves the reduction of the drag for a swept wing subject to transonic flow conditions. The other problem concerns the maximization of the wing span efficiency for a wing subject to subcritical flow conditions. The results show that the method is robust and can be used to effectively solve such aerodynamic shape optimization problems. However, it is also found that a more careful consideration of the constraints used might be required to increase the feasibility of the optimal aerodynamic shape that is obtained.

Subsequently, a discussion is presented on solving aerodynamic shape optimization problems involving a wind turbine rotor blade. For that purpose, possible objective functions for the present optimization framework are discussed. Moreover, the issues that arose solving the optimization problem for a wind turbine blade are identified and suggestions are put forward for resolving these issues.

SAMENVATTING

Voor het opwekken van duurzame energie levert de windturbine een belangrijke bijdrage. De zogenaamde windenergie die hiermee wordt geproduceerd, wordt opgewekt door de kinetische energie — ook wel bewegingsenergie genaamd — uit de wind te winnen. Dit gebeurt door een roterende beweging om te zetten in elektrische energie. Op verschillende geografische locaties heersen verschillende windomstandigheden en het ontwerp van een windturbine is afhankelijk van deze omstandigheden. Een belangrijk aspect bij het ontwerp van een windturbine betreft de aerodynamische vorm van het rotorblad. Het onderzoek dat in dit proefschrift wordt gepresenteerd, richt zich op het ontwikkelen van een rekenmethode die gebruikt kan worden voor aerodynamische optimalisatie van windturbinebladen.

Andere dan aerodynamische aspecten zijn ook belangrijk bij het ontwerp van een windturbine, bijvoorbeeld het constructietechnische ontwerp van het rotorblad. Eén van de doelen van het onderzoek dat in dit proefschrift wordt gepresenteerd is dat de methode voor het aerodynamische bladontwerp zo naadloos mogelijk in te passen is in het multidisciplinaire ontwerpproces. Daarnaast moet de optimalisatiemethode efficiënt en volledig automatisch zijn.

Na onderzoek naar een geschikte parametrisatiemethode voor de geometrie van het windturbineblad is er gekozen om gebruik te maken van het niet-uniforme rationale basis spline (NURBS) oppervlak om de aerodynamische bladvorm van het windturbineblad te representeren. Een dergelijke beschrijving van de parametrisatie van de geometrie is verenigbaar met zogenaamde CAD methoden, waardoor het opnemen van de aerodynamische optimalisatiemethode in het multidisciplinaire ontwerpproces wordt vergemakkelijkt. Er is een modeloptimalisatievraagstuk beschouwd om te bepalen welk verband er bestaat tussen het aantal ontwerpvariabelen dat nodig is om de ontwerpruimte voldoende nauwkeurig te beschrijven voor het succesvol oplossen van een aerodynamisch optimalisatievraagstuk en het aantal ontwerpvariabelen dat nodig is voor een nauwkeurige NURBS representatie van een aerodynamische vorm. Om een NURBS oppervlak representatie van een windturbineblad te verkrijgen, is er een rekenmethode ontwikkeld die passende waarden voor de NURBS parameters bepaalt, zodanig dat het NURBS oppervlak de oorspronkelijke bladvorm nauwkeurig benadert. De nauwkeurigheid waarmee het NURBS oppervlak de referentiegeometrie benadert, is gepresenteerd voor twee representatieve windturbinebladen. Daaruit blijkt, dat voldoende nauwkeurigheid kan worden behaald met een NURBS oppervlak dat bestaat uit doorsnedes, ter plaatse van 13 stations langs de spanwijdte. Elke sectie heeft 13 controlepunten.

In het onderhavige onderzoek zijn doelfuncties beschouwd, waarvoor de oplossing

nodig is van partiële differentiaalvergelijkingen die de stroming beschrijven: de Euler vergelijkingen. Een benadering van de oplossing van deze vergelijkingen wordt bepaald door middel van het oplossen van de gediscretiseerde vergelijkingen. Voor het oplossen van de gediscretiseerde Euler vergelijkingen is discretisatie van het volledige stromingsdomein noodzakelijk. Tijdens het optimalisatieproces wordt de vorm van het rotorblad aangepast, zodat er telkens een nieuwe discretisatie van het rekendomein nodig is. Om hierin te voorzien, is er een hyperbolische rekenroostergeneratiemethode geïmplementeerd die voor elke stap in het optimalisatieproces een rekenrooster van hoge kwaliteit kan verschaffen, met name in het deel van het stromingsdomein in de buurt van het oppervlak van het rotorblad.

In combinatie met de hyperbolische rekenroostergeneratiemethode, toegepast in de buurt van het blad, is er gebruik gemaakt van extra rekenroosterblokken in het resterende deel van het rekendomein waarbij overlap van de verschillende blokken is toegestaan. Deze aanpak — die meestal wordt aangeduid met de Engelse term '*composite overset grid*' strategie — vereist, dat in het gebied waar blokken elkaar overlappen, de stromingsoplossing wordt overgedragen tussen de verschillende rekenroosterblokken. Om de zogenaamde verbindingsinformatie te bepalen, die nodig is om de stromingsoplossing te kunnen overdragen, is er gebruik gemaakt van een zogenaamde *implicit hole cutting* methode.

Een oplossing van de Euler vergelijkingen voor stationaire stroming — voor een stromingsdomein gediscretiseerd door middel van een gestructureerd *composite overset* rekenrooster — is verkregen door gebruik te maken van een celgecentreerde tweede-orde eindige-volume methode. Stationaire oplossingen worden berekend door gebruik te maken van de methode van Newton. De geschikte beginoplossing die nodig is voor de methode van Newton wordt verkregen door een reeks expliciete Runge-Kutta pseudotijdstappen uit te voeren. De orde van nauwkeurigheid van de numerieke methode is geverifiëerd en bovendien is de numerieke methode gevalideerd, door de resultaten van de numerieke methode te vergelijken met uit de literatuur verkregen resultaten van een experiment.

Voor de aerodynamische vormoptimalisatiemethode wordt een gradiënt-gebaseerde aanpak toegepast, zodat er gebruik gemaakt kan worden van een groot aantal ontwerpvariabelen, terwijl het benodigde aantal bepalingen van de doelfunctie beperkt blijft. De discrete geadjungeerde vergelijking-methode wordt gebruikt, om efficiënt de totale afgeleide van de doelfunctie naar alle ontwerpvariabelen te bepalen. Er is gebruik gemaakt van duale getallen rekenkunde om de partiële afgeleiden te bepalen. Door het toepassen van sjabloon functies en de knopenkleuring van grafen is er een methode ontwikkeld, die kan worden gebruikt om op een efficiënte wijze partiële afgeleiden te berekenen die volledig consistent zijn met de bijbehorende functie. Bovendien vermijdt deze aanpak de noodzaak om een nieuwe functie te gebruiken waarmee de partiële afgeleiden kunnen worden bepaald. Verder is er aangetoond dat de afgeleiden die berekend zijn met behulp van de geadjungeerde vergelijking-methode consistent zijn met afgeleiden die door een andere methode zijn bepaald.

Er is een optimalisatiekader ontwikkeld, door de verschillende onderdelen die geïmplementeerd zijn te koppelen aan een gradiënt-gebaseerd optimalisatiealgoritme. Vervolgens zijn er twee verschillende vormoptimalisatievraagstukken beschouwd. Het ene vraagstuk betreft de reductie van de weerstand van een pijlvleugel in transone stromingscondities. Het tweede vraagstuk heeft betrekking op de maximalisatie van de vleugelspanwijdte-efficiëntie voor een vleugel die onderhevig is aan subcritische stromingscondities. De resultaten laten zien dat de methode robuust is en kan worden gebruikt om op een effectieve manier aerodynamische vormoptimalisatievraagstukken op te lossen.

Vervolgens wordt het oplossen van een vormoptimalisatievraagstuk besproken, dat betrekking heeft op een windturbinerotorblad. Mogelijke doelfuncties zijn bediscussieerd. Verder zijn de problemen die optreden bij het oplossen van een optimalisatievraagstuk voor een windturbineblad geïdentificeerd en er worden suggesties gedaan voor het oplossen van de betreffende kwesties.

CONTENTS

Summary **i**

Samenvatting **iii**

Contents **vii**

Nomenclature and notation **xi**

1 Introduction **1**
- 1.1 Objectives . . . 2
- 1.2 Wind turbine blade design . . . 2
- 1.3 Flow model . . . 2
  - 1.3.1 Discretization method . . . 3
- 1.4 Flow domain discretization . . . 3
  - 1.4.1 Overset grids . . . 5
- 1.5 Optimization . . . 5
  - 1.5.1 Optimization methods . . . 5
  - 1.5.2 Parametrization method . . . 6
- 1.6 Sensitivity analysis . . . 7
  - 1.6.1 Analytical differentiation . . . 7
  - 1.6.2 Finite-difference approximation . . . 7
  - 1.6.3 Complex-step finite-difference method . . . 9
  - 1.6.4 Dual number method . . . 10
  - 1.6.5 Algorithmic differentiation . . . 10
  - 1.6.6 Adjoint equation method . . . 12
  - 1.6.7 Approach . . . 13
- 1.7 Implementation on parallel computers . . . 14
- 1.8 Related work . . . 14
- 1.9 Thesis outline . . . 16

2 Shape parametrization **17**
- 2.1 Introduction . . . 17
  - 2.1.1 Requirements . . . 17
  - 2.1.2 Methods employed in aerodynamic shape optimization . . . 18
  - 2.1.3 Considerations . . . 21

2.2 Non-uniform rational basis spline surface 21
  2.2.1 Basis spline curve 21
  2.2.2 Rational basis spline curve 24
  2.2.3 NURBS surface 24
2.3 Aerofoil parametrization with NURBS curve 25
  2.3.1 Aerofoil optimization 26
2.4 Parametrization of the wind turbine rotor blade 29
2.5 Summary 31

3 Flow domain discretization 33
3.1 Introduction 33
  3.1.1 Requirements 34
  3.1.2 Considerations 34
3.2 Surface grid generation 35
  3.2.1 Vertex distribution on boundary curves 35
  3.2.2 Transfinite interpolation 38
  3.2.3 Elliptical surface grid generation 40
3.3 Field grid generation 43
  3.3.1 Hyperbolic field grid equations 43
  3.3.2 Spatial discretization 43
  3.3.3 Implementation details 45
  3.3.4 Boundary conditions 48
3.4 Background grids 49
  3.4.1 Cartesian 49
  3.4.2 Cylindrical 50
  3.4.3 Rhombus 50
3.5 Summary and results 50

4 Overset block connectivity 53
4.1 Introduction 53
  4.1.1 Concept and terminology 54
4.2 Alternating binary tree search 55
  4.2.1 Tree generation 55
  4.2.2 Search procedure 56
  4.2.3 Other entities 56
4.3 Evaluation of surface integrals 58
  4.3.1 Zipper grid generation method 58
  4.3.2 Verification 62
  4.3.3 Use of zipper grid 63
4.4 Elimination of cells outside physical flow domain 64
  4.4.1 Ray-casting method 65
4.5 Domain connectivity 68
  4.5.1 Concept 68
  4.5.2 In cell qualification 69
  4.5.3 Interpolation coefficients 70
  4.5.4 Explicit expression in terms of field cells 71
4.6 Summary 74

5 Flow model and solution method 75
5.1 Governing equations . . . 76
5.2 Spatial discretization . . . 77
  5.2.1 Finite volume method . . . 77
  5.2.2 Computation of metrics . . . 79
5.3 Numerical fluxes . . . 80
  5.3.1 Jameson-Schmidt-Turkel scheme . . . 80
  5.3.2 Roe's approximate Riemann solver . . . 81
5.4 Reconstruction of state at interface . . . 83
  5.4.1 Reconstruction about centre of mass . . . 83
  5.4.2 Reconstruction considering grid as uniform Cartesian . . . 84
  5.4.3 MUSCL-type reconstruction . . . 84
5.5 Boundary conditions . . . 85
  5.5.1 Solid wall . . . 85
  5.5.2 Periodicity . . . 86
  5.5.3 Symmetry . . . 87
  5.5.4 Far-field . . . 87
  5.5.5 Subsonic outflow . . . 88
  5.5.6 Overset grids . . . 88
5.6 Pseudo-time-integration to steady state . . . 89
  5.6.1 Runge-Kutta time-integration . . . 89
  5.6.2 Newton's method . . . 90
5.7 Non-inertial frame of reference . . . 96
  5.7.1 Geometric conservation law . . . 97
5.8 Verification . . . 97
  5.8.1 Spatial order of convergence . . . 98
  5.8.2 Flow configuration . . . 98
  5.8.3 Results for single block grid . . . 100
  5.8.4 Results for composite overset discretization . . . 109
  5.8.5 Summary of the results and concluding remarks . . . 110
5.9 Validation . . . 114
  5.9.1 Numerical set-up . . . 114
  5.9.2 Results . . . 115

6 Sensitivity analysis 121
6.1 Dual numbers . . . 121
  6.1.1 Implementation . . . 122
  6.1.2 Approach . . . 124
6.2 Discrete adjoint equation method . . . 125
  6.2.1 Derivation . . . 125
  6.2.2 Discrete versus continuous . . . 126
6.3 Efficient application of dual number method . . . 127
  6.3.1 Graph vertex colouring . . . 127
6.4 Jacobian matrix . . . 129
  6.4.1 Verification . . . 133
6.5 Grid sensitivity of flow residual . . . 134
6.6 Flow sensitivity of objective function . . . 136

6.7 Grid sensitivity of objective function 137
6.8 Design variable sensitivity of objective function 138
6.9 Design variable sensitivity of grid 139
6.10 Solution method for the adjoint equations 140
6.11 Verification of adjoint implementation 141
6.11.1 Approach 141
6.11.2 Geometry and flow configuration 142
6.11.3 Results 143
6.12 Summary 143

7 Optimization method and results 147
7.1 Introduction 147
7.2 General aspects 149
7.2.1 Initial geometry and parametrization 149
7.2.2 Flow domain and discretization 150
7.2.3 Geometrical constraints 151
7.3 Drag minimization in transonic flow 152
7.3.1 Definition of optimization problem 152
7.3.2 Results and discussion 154
7.4 Wing span efficiency maximization 156
7.4.1 Definition of optimization problem 156
7.4.2 Results and discussion 157
7.5 On solving a rotor blade optimization problem 160
7.5.1 Geometry and parametrization 160
7.5.2 Flow conditions 161
7.5.3 Flow domain and discretization 162
7.5.4 Objective function and constraints 163
7.5.5 Approach and discussion 167
7.6 Concluding remarks 169

8 Concluding remarks and recommendations 171
8.1 Concluding remarks 171
8.2 Recommendations 174

Bibliography 179

Index 195

A Method for computing grid generation dissipation coefficient 199

B Kármán-Trefftz O-grid generation 203

C Results spatial order of convergence study 207

D Measuring the efficiency of the dual number method in vector mode 213

Acknowledgements 217

Curriculum Vitae 219

NOMENCLATURE AND NOTATION

FOR the convenience of the reader of this thesis, a list of most symbols used in this dissertation is provided here. Furthermore, the reader is informed about the typographical custom that is applied to improve the readability and understandability of formulas.

Notation

To improve the readability and understandability of formulas, figures and text presented in this thesis, a particular custom has been adhered to. All variables for which a Latin letter has been used and which can only take an integer value have been printed in roman typeface, e.g. n. Dimensionless quantities, like the Reynolds number Re, are also printed in roman typeface. Other variables for which these conditions do not apply are printed in italic typeface, e.g. x. However, units are also printed in roman typeface, but these are always presented between square brackets, e.g. $[\mathrm{m}]$. Furthermore, sets are always represented by so-called black-board bold letters, e.g. $\mathbb{R}$. Moreover, when a subscript in roman serif typeface is encountered, it is used to specify an element of, for instance, a matrix or a vector, e.g. u_i. On the other hand, if the subscript is in roman sans-serif typeface, it is used as a qualification or specification of a variable, e.g. c_d. Vectors are depicted in bold face, e.g. $\boldsymbol{x}$, while matrices can be recognized by a double underline, e.g. $\underline{\underline{A}}$.

Nomenclature

Most special characters, symbols and abbreviations encountered in this thesis are listed here. The symbols are grouped according to alphabet and case. They are listed in alphabetic order and for some symbols a reference is given to the most significant equation in which the symbol appears, by means of the equation number between square brackets at the end of the row. Some variables have been used more than once to represent different quantities, the correct meaning is in that case believed to be clear from the context.

Capital Roman

$\underline{\underline{\tilde{A}}}$	approximation to flux Jacobian	
$\underline{\underline{A}}$	matrix in hyperbolic field grid equations	[3.25]
$\underline{\underline{B}}$	matrix in hyperbolic field grid equations	[3.26]
C	Courant number	[5.47]
$\boldsymbol{C}$	vector of constraint functions	[1.1]
$\boldsymbol{C}^{\mathrm{p}}$	non-uniform rational basis spline (NURBS) curve of order p	[2.5]
$\underline{\underline{C}}$	matrix in hyperbolic field grid equations	[3.27]
$\mathbb{C}$	set of complex numbers	
$\mathcal{C}$	parametric curve	
C_{D}	drag coefficient	[5.80]
C_{L}	lift coefficient	[5.79]
C_{p}	pressure coefficient	[5.81]
C_{power}	power coefficient	[7.5]
$\boldsymbol{D}$	artificial dissipation term central flux discretization	[5.17]
$\boldsymbol{D}^{\mathrm{II}}$	shock capturing term central flux discretization	[5.18]
$\boldsymbol{D}^{\mathrm{IV}}$	background dissipation term central flux discretization	[5.19]
$\mathbb{D}$	set of dual numbers	[1.9]
$\overline{\mathbb{D}}^{\mathrm{n}}$	set of dual numbers, with a non-real component of dimension n	[6.2]
$\mathfrak{D}\left(\cdot\right)$	non-real part of dual number	
$\boldsymbol{F}_{\mathsf{a}}$	force per unit span exerted on aerofoil	[5.74]
$\boldsymbol{F}_{\mathrm{x}}$	first column vector of the convective flux tensor	
$\underline{\underline{F}}$	convective flux tensor	[5.5]
$\underline{\underline{F}}_{\mathsf{rot}}$	convective flux tensor considered in co-rotating frame of reference	[5.67]
$\boldsymbol{\mathcal{F}}$	numerical convective flux	[5.11]
H	total enthalpy per unit mass	
$\underline{\underline{H}}_{\mathrm{m}}$	upper Hessenberg matrix of dimensions $(\mathrm{m}+1)\times\mathrm{m}$	[5.55]
$\underline{\underline{I}}$	identity matrix	
$\mathcal{I}$	objective function	[1.1]
$\mathfrak{I}\left(\cdot\right)$	imaginary part of complex number	
J_{0}	Bessel function of the first kind	
$\tilde{\boldsymbol{K}}$	right eigenvector of approximation to flux Jacobian	[5.26]
$\mathcal{K}_{\mathrm{m}}$	Krylov subspace of dimension m	[5.53]
$\mathcal{L}$	characteristic length scale	
M	Mach number	
$M_{\mathsf{z},\frac{\bar{c}}{4}}$	pithing moment of aerofoil	[5.77]
N_{F}	number of donors for fringe control volume	[5.44]
N_{c}	number of control volumes	
$N_{\mathrm{i}}^{\mathrm{q}}$	polynomial basis function of degree q	[2.2]
N_{η}	approximation of local matrix norm	[A.1]
N_{ξ}	approximation of local matrix norm	[A.1]
$\mathbb{N}_{0}$	set of natural numbers, including zero	
$\mathbb{N}_{1}$	set of natural numbers	
$\mathcal{O}\left(\cdot\right)$	order of magnitude	
P	control function	[3.19]

$\bar{P}$	power extracted by the wind turbine rotor	[7.5]
$\boldsymbol{P}$	coordinate[1] of control point of NURBS curve/ surface	[2.6]
$\underline{\underline{P}}$	preconditioning matrix	[5.58]
$\bar{Q}$	control function	[3.19]
$\boldsymbol{Q}$	volumetric source term in conservation equations due to rotation	[5.68]
$\underline{\underline{Q}}_{\mathrm{m}}$	matrix containing the first m basis vectors of the Krylov subspace	[5.56]
$\bar{R}$	radius of the wind turbine rotor	[7.5]
R_{c}	radius of the core of isentropic vortex	
$\boldsymbol{R}_{\mathrm{a}}$	residual of the semi-discrete governing equations for control volume a	[5.45]
$\boldsymbol{R}_{\mathrm{i}}^{\mathrm{q}}$	rational basis function NURBS curve	[2.4]
$\boldsymbol{R}_{\mathrm{ij}}^{\mathrm{pq}}$	rational basis function NURBS surface	[2.7]
$\bar{\boldsymbol{R}}$	right hand side of equation used to compute control functions	[3.19]
$\mathbb{R}$	set of real numbers	
$\boldsymbol{\mathcal{R}}$	vector of residuals of discretized flow equations.	[5.49]
Re	Reynolds number	
S	planform area wing	
S_{k}	scaling function for grid layer k	[A.6]
S_{wing}	surface area wing	
$\mathcal{S}$	parametric surface	
$\boldsymbol{S}^{\mathrm{pq}}$	non-uniform rational basis spline surface of order p and q	[2.6]
T	triangle	
$\underline{\underline{T}}$	rotation matrix	[5.10]
$\bar{\boldsymbol{U}}$	vector with conserved variables	[5.4]
$\mathbb{U}_{\mathsf{b}}$	set containing values of parametric variables for boundary vertices	[3.10]
$\mathbb{U}_{\mathsf{i}}$	set containing values of parametric variables for inner vertices	[3.11]
$\mathbb{U}_{\mathrm{m}}$	element m of the ordered set of vectors with control-volume-averaged conserved variables	[5.44]
$\mathcal{U}$	characteristic velocity	
$\boldsymbol{\mathcal{U}}$	vector with conserved variables for the discrete flow solution	[5.45]
$\boldsymbol{\mathcal{U}}^*$	steady-state solution	[5.51]
V	volume of a cell	[5.13]
$\mathcal{V}$	signed volume of a tetrahedron	[4.4]
V_{wind}	wind speed	[7.5]
V_{wing}	internal volume of wing	[7.3]
$\boldsymbol{X}$	computational grid	
$\mathbb{X}_{\mathsf{s}}$	set containing all coordinates of a surface grid	[3.17]

Lower-case Roman

b	wing span	[7.4]
c	speed of sound	
$\bar{c}$	chord length of aerofoil	
$\bar{c}_{\mathsf{loc}}$	local chord length of wing section	

---

[1] In this dissertation, a coordinate is considered a vector quantity. Therefore, the singular form is used for denoting a single physical location.

$\hat{c}$	mean aerodynamic chord length of wing	
c_{d}	drag coefficient for aerofoil section	[5.76]
c_{l}	lift coefficient for aerofoil section	[5.75]
c_{m}	moment coefficient for aerofoil section	[5.78]
$d^{\eta}_{\mathrm{i,j,k}}$	ratio of the distance between grid vertices from two subsequent grid layers in η-direction	[A.5]
$\tilde{d}^{\eta}_{\mathrm{i,j,k}}$	grid point distribution sensor function for η-direction	[A.3]
$d^{\xi}_{\mathrm{i,j,k}}$	ratio of the distance between grid vertices from two subsequent grid layers in ξ-direction	[A.4]
$\tilde{d}^{\xi}_{\mathrm{i,j,k}}$	grid point distribution sensor function for ξ-direction	[A.2]
∂V	surface bounding a control volume	
e	wing span efficiency	[7.4]
e_{int}	internal energy per unit mass	
e_{t}	total energy per unit mass	[5.4]
$\boldsymbol{e}_{\mathrm{x}}$	unit vector in x-direction	[5.77]
$\boldsymbol{e}_{\mathrm{z}}$	unit vector in z-direction	[5.77]
$\boldsymbol{e}_{0}$	first column vector of identity matrix	[5.56]
f	arbitrary analytic function	
$\boldsymbol{f}$	right hand side vector for hyperbolic field grid equations	[3.28]
h	magnitude of finite-difference disturbance	
h	typical control volume size	[5.72]
h_{c}	typical control volume size for coarse grid	
h_{f}	typical control volume size for fine grid	
h_{m}	typical control volume size for medium grid	
h_{ij}	non-zero entry of upper Hessenberg matrix	[5.54]
i	imaginary unit, i.e. $\sqrt{-1}$	
i	index	
j	index	
k	index	
$\boldsymbol{k}_{\parallel}$	unit vector parallel to the flow	
$\boldsymbol{k}_{\perp}$	unit vector perpendicular to the flow	
m	index	
n	number of control volumes	
n	number of vertices	
$\boldsymbol{n}$	unit normal vector	
$\mathrm{n}_{\mathrm{add}}$	number of floating-point additions	[D.5]
n_{c}	number of control volumes	
$\mathrm{n}_{\mathrm{cons}}$	number of conserved flow variables	
n_{d}	dimension of the non-real part of the dual number	
$\mathrm{n}_{\mathrm{der}}$	number of derivatives that must be computed to populate the Jacobian matrix	[D.1]
n_{fp}	number of floating-point operations required to populate the Jacobian matrix	[D.5]
$\bar{\mathrm{n}}_{\mathrm{fp}}$	number of floating-point operations required to populate the Jacobian matrix by means of a central-difference approach	
$\mathrm{n}_{\mathrm{add}}$	number of floating-point multiplications	[D.5]
n_{p}	number of design parameters	

n_{v}	number of vertices	
p	static pressure	[5.6]
p	order of NURBS curve/ NURBS surface	
$\bar{p}$	spatial order of convergence	[5.72]
q	order of NURBS surface for second direction	
$\boldsymbol{q}$	orthonormal basis vector of Krylov subspace	[5.54]
$\boldsymbol{r}$	coordinate of grid vertex	
r_{le}	leading edge radius of aerofoil	
s	arc length along curve	
s	entropy	[5.40]
t	time	
t	parametric variable	
$\bar{t}$	relative maximum thickness of aerofoil	
$\bar{t}_{\mathsf{jac}}$	time required to populate the Jacobian matrix	[D.2]
$\bar{t}_{\mathsf{min}}$	minimum thickness aerofoil	
$\bar{t}_{\mathsf{ref}}$	reference time	[D.2]
$\boldsymbol{u}$	velocity of the fluid $\in \mathbb{R}^3$, with components $(u, v, w)^T$	
$\boldsymbol{u}$	parametric variable NURBS surface $\in \mathbb{R}^2$, with components $(u, v)^T$	
u	parametric variable NURBS curve/ NURBS surface	
$\boldsymbol{u}_{\mathsf{rot}}$	velocity considered in co-rotating frame of reference	[5.64]
$\boldsymbol{u}_{\partial V}$	velocity of surface bounding a control volume	[5.1]
$\boldsymbol{u}_{\mathsf{D0}}$	velocity for first control volume near boundary	[5.38]
$\boldsymbol{u}_{\mathsf{H0}}$	velocity for first halo control volume	[5.38]
$\boldsymbol{u}_{\mathrm{m}}$	vector that minimizes ℓ^2-norm of the projection of a vector on the orthonormal basis of the Krylov subspace of dimension m	[5.56]
v	second parametric variable NURBS surface	
v_{a}	weighting factor cell volume smoothing	[3.36]
$\boldsymbol{v}_{01}$	vector pointing from vertex 0 to 1	[4.2]
x	first component of coordinate in physical space	
$\tilde{\boldsymbol{x}}$	coordinate in transformed coordinate system	
$\boldsymbol{x}_{\mathsf{cm}}$	centre of mass of control volume	[5.14]
x_{te}	x-component of coordinate of trailing edge	
y	second component of coordinate in physical space	
z	third component of coordinate in physical space	
z_1	mapping parameter Kármán-Trefftz conformal transformation	[5.73]
z_2	mapping parameter Kármán-Trefftz conformal transformation	[5.73]

Capital Greek

Δt_{a}	local time-step size for control volume a	[5.47]
Δt_{res}	time required for a single computation of residual vector	[D.2]
Γ	strength of isentropic vortex	
Θ	shock sensor	[5.24]
Λ	estimate of spectral radius for local convective flux Jacobian	[5.20]
Λ_{le}	leading edge sweep angle	

Λ_{te}	trailing edge sweep angle	
Ξ_{j}	perimeter of local blade contour	[2.8]
Υ	ratio of curvature-weighted arc length for two subsequent elements of a discretized curve	[3.6]
Φ	grid orthogonality measure	[3.43]
Φ^{-}	limiting function left of the interface	[5.33]
Φ^{+}	limiting function right of the interface	[5.34]
$\boldsymbol{\Omega}$	angular velocity	[5.64]

Lower-case Greek

α	angle of attack	
α_{m}	wave strength corresponding to eigenvector m	[5.26]
$\bar{\alpha}$	maximum allowable relative distance between consecutive vertices on a curve	[3.1]
α_{splay}	extrapolation factor that specifies amount of splay	[3.42]
$\tilde{a}^{\eta}_{\mathrm{i,j,k}}$	grid angle function for η-direction	
$\tilde{a}^{\xi}_{\mathrm{i,j,k}}$	grid angle function for ξ-direction	[A.12]
β_{m}	Runge-Kutta time-integration coefficient	[5.46]
δ	stretching factor	[3.7]
δ_{η}	finite-difference operator for first derivative in η-direction	[3.32]
δ^{2}_{η}	finite-difference operator for second derivative in η-direction, implicit part	[3.34]
$\bar{\delta}^{2}_{\eta}$	finite-difference operator for second derivative in η-direction, explicit part	[3.34]
δ_{ξ}	finite-difference operator for first derivative in ξ-direction	[3.32]
δ^{2}_{ξ}	finite-difference operator for second derivative in ξ-direction, implicit part	[3.34]
$\bar{\delta}^{2}_{\xi}$	finite-difference operator for second derivative in ξ-direction, explicit part	[3.34]
γ	ratio of specific heats	[5.6]
ϵ_{e}	dissipation coefficient proportionality factor, explicit part	
ϵ_{i}	dissipation coefficient proportionality factor, implicit part	[3.37]
ϵ_{i}	variable for preventing undefined result of limiter function	[5.35]
ϵ_{η}	dissipation coefficient grid generation, implicit part	[A.14]
$\bar{\epsilon}_{\eta}$	dissipation coefficient grid generation, explicit part	[3.34]
ϵ_{ξ}	dissipation coefficient grid generation, implicit part	[A.13]
$\bar{\epsilon}_{\xi}$	dissipation coefficient grid generation, explicit part	[3.34]
ϵ^{II}	artificial dissipation coefficient central flux discretization	[5.21]
ϵ^{IV}	artificial dissipation coefficient central flux discretization	[5.22]
ε	dual number unit, i.e. $\varepsilon^{2} \equiv 0$	[1.9]
ζ	third component of coordinate in computational space	
ζ_{c}	mapped coordinate of centre of line connecting leading edge to trailing edge in Kármán-Trefftz conformal transformation	
ζ_{le}	mapped coordinate of leading edge in Kármán-Trefftz conformal transformation	

ζ_1	mapping parameter Kármán-Trefftz conformal transformation	[5.73]
ζ_2	mapping parameter Kármán-Trefftz conformal transformation	[5.73]
η	second component of coordinate in computational space	
$\eta_{\bar{\mathrm{n}}}$	parameter for controlling required convergence accuracy of the GMRES method	[5.63]
θ	ratio of curvature weighted arc length	[3.6]
$\bar{\theta}$	implicit weighting factor	[3.34]
κ	local curvature	[3.3]
$\bar{\kappa}_{\mathsf{max}}$	maximum allowable curvature	
$\hat{\kappa}$	linear reconstruction weighting factor	[5.28]
λ	relative importance of curvature	[3.5]
λ	taper ratio	
$\boldsymbol{\lambda}$	vector of Lagrange multipliers	[1.12]
$\bar{\lambda}$	approximation of spectral radius	[5.48]
$\tilde{\lambda}$	eigenvalue of approximation to flux Jacobian	[5.27]
μ_0	characteristic viscosity	
ν	second knot vector NURBS surface	[2.6]
ξ	first component of coordinate in computational space	
π	ratio between a circle's circumference and diameter	
ρ	density of fluid	[5.1]
ρ_0	characteristic density	
σ	scaled computational coordinate	[3.12]
τ	scaled computational coordinate	[3.12]
$\underline{\underline{\tau}}$	stress tensor	[5.1]
τ_{te}	trailing edge angle aerofoil	
υ	knot vector NURBS curve/ NURBS surface	[2.5]
ϕ	arbitrary scalar field	[4.1]
ϕ^*	estimate for functional value for infinite resolution	[5.72]
$\boldsymbol{\chi}$	vector of design variables	[1.1]
$\boldsymbol{\psi}$	adjoint vector	[6.9]
$\boldsymbol{\omega}$	interpolation weight fringe cell	[4.8]

Other symbols

$=$	equal to
$\equiv$	identically equal to
$:=$	equal to by definition
$\mathbb{A} \cap \mathbb{B}$	intersection of set $\mathbb{A}$ and $\mathbb{B}$
$\mathbb{A} \cup \mathbb{B}$	union of set $\mathbb{A}$ and $\mathbb{B}$
$\mathbb{A} \Subset \mathbb{B}$	$\mathbb{A}$ is an open subset of $\mathbb{B}$
$\mathrm{Cov}\left[\cdot,\cdot\right]$	covariance of two stochastic variables
$\lceil\cdot\rceil$	ceiling
$\lfloor\cdot\rfloor$	floor
$\mathrm{E}\left[\cdot\right]$	expected value of a stochastic variable
mod	modulus operation
$\forall x$	for all elements x

$\langle \boldsymbol{U} \rangle_\mathrm{a}$	control-volume-averaged quantity for control volume a
$x \in \mathbb{A}$	x is an element of $\mathbb{A}$
$x \notin \mathbb{A}$	x is not an element of $\mathbb{A}$
$f : a \mapsto b$	f maps the element a to the element b
$f : \mathbb{A} \to \mathbb{B}$	f maps set $\mathbb{A}$ into the set $\mathbb{B}$
$\mathrm{Var}\left[\cdot\right]$	variance of a stochastic variable

Abbreviations

A.D.	*Anno Domini*
ADT	alternating digital tree
AGARD	Advisory group for aerospace research and development
BEM	blade element momentum
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CAD	computer aided design
CPU	central processing unit
CFL	Courant Friedrichs Lewy
CST	class function/ shape function transformation
DLR	*Deutsches Zentrum für Luft- und Raumfahrt*
DNS	direct numerical simulation
DOI	digital object identifier
dr.	doctor
e.g.	*exempli gratia*
elsA	*ensemble logiciel pour la simulation en Aérodynamique*
et al.	*et alii*
i.e.	*id est*
ir.	*ingenieur*
Fortran	The IBM Mathematical Formula Translating System
FPGA	field programmable gate array
GMRES	generalized minimal residual
GCL	geometric conservation law
GPU	graphical processing unit
IBM	International Business Machines corporation
ILU	incomplete lower-upper
INRIA	*l'Institut de recherche en informatique et en automatique*
ISBN	international standard book number
LES	large eddy simulation
LU	lower-upper
MHD	magnetohydrodynamics
MPI	message passing interface
MUSCL	monotone upstream-centred scheme for conservation laws
NACA	National advisory committee for Aeronautics
NASA	National aeronautical and space administration
NREL	National renewable energy laboratory
NURBS	non-uniform rational basis spline
ONERA	*Office national d'etudes et de recherches aerospatiales*

OpenCL	open computing language
PETSc	portable, extensible toolkit for scientific computation
PhD	doctor of philosophy
prof.	professor
RANS	Reynolds-averaged Navier-Stokes
RPM	recursive projection method
SNOPT	sparse nonlinear optimizer
Tapenade	tangent and adjoint penultimate automatic differentiation engine
viz.	*videlicet*
WENO	weighted essentially non-oscillatory

1

INTRODUCTION

> *"We can only see a short distance ahead, but we can see plenty there that needs to be done."* [189]
>
> — ALAN M. TURING (1912 – 1954)

THE power provided by the wind has been harnessed by mankind for millennia. First primarily for transport, such as in sailboats, later on it has also been used for mechanical work, like pumping water or milling grain. Nowadays, the principal application of wind power is for the production of electrical energy, by means of wind turbines; its use for transport is mainly of recreational nature.

Although the energy contained in the wind is free of charge, its collection and conversion to electrical energy are not. The construction and maintenance of wind turbines is a costly matter, which is reflected in the energy cost of the electrical energy produced by wind turbines — from here on referred to as 'wind energy'.

For a horizontal axis wind turbine — the wind turbine type considered in the present research — air flows over the blades of the rotor. Due to the geometry of the blade and its orientation with respect to the wind, a pressure difference is present between its lower and upper surface, resulting in a torque that makes the rotor rotate. The blades are connected to a hub, which is connected to the generator via a shaft. In most cases there is also a gearbox between the shaft connected to the rotor and the shaft connected to the generator. This gearbox increases the rotational speed of the generator axis, which allows for the design of a more compact generator. The generator subsequently converts the mechanical energy of the rotor to electrical energy. Because of the large capital cost associated with a wind turbine, it is important to design a rotor that operates efficiently, such that a large part of the kinetic energy available in the wind is extracted.

Wind conditions vary from site to site. Different wind conditions require a different wind turbine design. Since the design of a wind turbine is complicated, involving multiple disciplines, it is a tedious process to design wind turbines that satisfy the requirement of producing energy at a price competitive to the price of electrical energy from non-renewable energy sources. This statement is especially true for sites with non-optimal wind conditions, which are increasingly employed in order to meet the growing demand for renewable energy. This thesis presents an aerodynamic shape optimization method that can be used to facilitate the aerodynamic design of wind turbine rotor blades.

This chapter starts by stating the objective of the research presented in this thesis. Subsequently, the current common practice in wind turbine blade design is discussed and the approach taken in the present research is presented. Then, all aspects important in the development of an aerodynamic shape optimization method are introduced. This chapter ends with an overview of the work performed by others, related to the work treated in this thesis and finally, an outline of the present dissertation is given.

1.1 Objectives

The objective of this research is to investigate the potentials of an aerodynamic shape optimization method for improving the aerodynamic performance of a wind turbine rotor. Furthermore, the aim is to provide an optimization framework which can be used as a solid basis for a more extensive multidisciplinary optimization method for wind turbine rotors. For this purpose, the different aspects important in a shape optimization method are considered and a well informed choice is made with respect to each of these aspects. Therefore, in making these choices, the potential of extending the method is taken into account.

1.2 Wind turbine blade design

The current practice in the aerodynamic design of wind turbine blades is to design 2D sections of the blade at a number of different spanwise locations [86]. The 3D blade shape is then created by combining these 2D sections into a 3D geometry. For designing the rotor blade, it is common practice to use so-called engineering models — which are characterized by a low-fidelity modelling of the underlying physics at the gain of reduced computational requirements. Widely adopted engineering models used for blade design are based on the blade element momentum (BEM) theory [72, 157]. In this theory, the rotor blade is divided into different sections in radial direction. The flow around each section is assumed to be independent of each other. Moreover, the forces on the blade are determined solely by the lift and drag characteristics of the aerofoils used for each of the sections. In this approach, it is difficult to assess the 3D effects — that inherently exist in the flow field around a wind turbine blade, in particular in the tip and root region — and take these 3D flow features into account in the design. Therefore, in the research presented in this thesis, a different approach is taken. In this approach a method is employed which, instead of combining 2D aerofoil sections, uses a three-dimensional blade geometry as starting point. Based on results of the simulation of the flow around this three-dimensional geometry, the shape is adapted. For this purpose, an aerodynamic shape optimization method is developed. In this way, 3D flow features are automatically taken into account in the aerodynamic design of the blade. The different aspects considered in the development of the method are treated in the following sections.

1.3 Flow model

The flow around a wind turbine rotor can be modelled to different degrees of complexity. There are the potential flow models, which are most efficiently implemented using an integral boundary element method for incompressible flows. These methods only require the discretization of the boundary surfaces leading to the aforementioned efficiency. More elaborate methods belong to the class of field methods, which do require the discretization of the three-dimensional flow domain. This class includes models based on the Euler and

Navier-Stokes equations. The former do not include effects of viscosity, while the latter do. If effects of viscosity are included, the phenomenon turbulence and its modelling must be considered. The most accurate modelling approach is direct numerical simulation (DNS) of the flow. In this approach, all relevant length and time scales of the turbulent flow are resolved. For a wind turbine application however, the computational requirements of such a method are such that they cannot be met with currently available computational resources. Large-eddy simulation [167] gives the most accurate results, when the effect of turbulence is modelled. However, due to the inherently unsteady nature of this flow model and because of the quite severe requirements with respect to grid resolution, application of this model in an aerodynamic design optimization context is not feasible for currently available computational resources. One of its alternatives is the application of turbulence models based on the Reynolds-averaged Navier-Stokes (RANS) equations. In these models averaging of the flow variables is performed, in such a way that unsteady effects due to turbulence are eliminated from the result, but its influence on mean flow variables is modelled. Aerodynamic optimization methods have been developed for which the flow model is based on a RANS turbulence model [53, 115, 130]. For the present work however, the Euler equations are used to model the flow. This choice was made to alleviate the required computational resources for solving an optimization problem and to be able to focus on the other aspects that are important for an aerodynamic optimization method.

1.3.1 Discretization method

The spatial discretization of the partial differential equations that model the flow is carried out based on a cell-centred finite volume method. Second-order spatial accuracy can be achieved, either by using a central discretization of the convective flux with an added fourth-order artificial dissipation term [92] or by means of an upwind scheme [147] in combination with linear reconstruction of the state at the interface.

Only configurations for which a steady-state solution of the governing equations exists, will be considered in the optimization method. For that purpose, time-integration of the governing equations is used to reach the steady-state solution. This is achieved by a combination of explicit and implicit time-integration. First, an explicit scheme is used to determine a suitable solution which can be used as a starting point for the implicit time-integration method. This implicit method is subsequently used to accelerate — both in terms of CPU-time as well as in terms of number of iterations — the convergence to a steady-state solution.

1.4 Flow domain discretization

Solving the discretized equations that govern the flow requires the discretization of the flow domain. For this purpose multi-block structured grids are used. Unstructured grids have the advantage of a large geometrical flexibility and easy application for complex geometries. For structured grids on the other hand, it is simpler to extend the spatial discretization of the finite volume method to higher-order schemes — i.e. higher than second-order. It is also easier to apply efficient flow solution methods for structured grids. Furthermore, structured grids require fewer cells, compared to unstructured grids, to satisfy the requirement of having adequate grid resolution in one direction — to

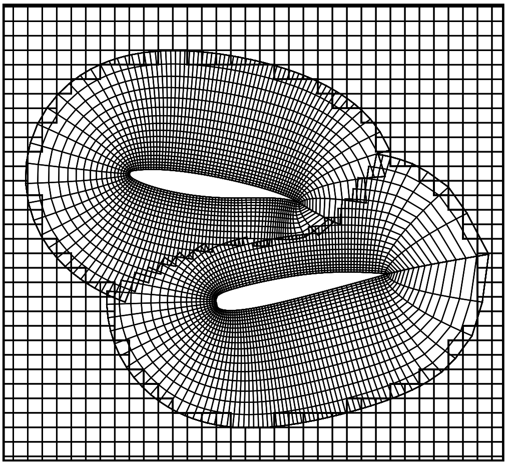


Figure 1.1: Composite overset grid consisting of body-fitted O-grids around two aerofoils and a Cartesian background grid — with overlap removed, i.e. only field cells shown.

accommodate large gradients of the flow solution in that direction — while retaining a much coarser spacing in the other direction. For a structured grid, grid quality can be maintained by using a smaller resolution of nodes in the direction along the surface at locations where variations in the flow solution are smooth — e.g. in the spanwise direction of the wind turbine blade. For an unstructured grid the same approach leads to a deterioration of the grid quality. Therefore, the choice is made to use structured grids for the discretization of the flow domain around the wind turbine blade.

Since the shape of the wind turbine blade changes during the aerodynamic optimization procedure, a new discretization is required for the resulting new flow domain around the modified blade geometry. The approach taken in the present research is to generate a body-fitted grid — instead of deforming the existing grid, which may result in an excessively distorted grid [29] — for each geometry that forms during the optimization procedure. For this purpose, a hyperbolic grid generation method has been implemented. Hyperbolic grid generation methods are known for being fast and providing high quality grids [81]. Generation of a new grid for each new flow domain, instead of the deformation of an initial discretization to match the modified domain, ensures that a high grid quality is maintained throughout the optimization, resulting in better flow solutions and a more stable algorithm. Better flow solutions in turn lead to a more accurate evaluation of the objective function used in the optimization. Moreover, a stable algorithm is of key importance for an effective optimization procedure.

To render the discretization of the flow domain less involved and to allow for flexibility of the optimization procedure, block overlap is allowed for the multi-block discretization of the flow domain. Grids for which overlap of the blocks occurs, such as in figure 1.1, are designated composite overset grids.

1.4.1 Overset grids

The conventional overset grid discretization method — in which multiple structured grids overlap in order to cover the flow domain — combines the advantages of structured grids with the geometrical flexibility of unstructured grids. The price to pay for the advantages inherent to an overset grid approach is reflected in the need to establish domain connectivity [120], which is required for enforcing boundary conditions in the flow solution method. Block connectivity information, for the multi-block discretization of a flow domain, is used to transfer the dependent variable information between components of the grid [120]. For point-matching grids this procedure is straightforward. However, if overlap of the blocks is allowed, more work needs to be done to determine a suitable block connectivity. The method used for this purpose is subject of chapter 4.

Apart from determining the domain connectivity for a composite overset grid, additional work is also required for the flow solution method to be able to handle overset grids. In regions of overlap, the dependent variables must be transferred by means of interpolation.

1.5 Optimization

Solving a shape optimization problem is equivalent to finding an optimum — either a maximum or a minimum — of a mathematical function in a multidimensional space spanned by the independent variables. These independent variables, denoted by $\boldsymbol{\chi} \in \mathbb{R}^{\mathrm{n}}$, $\mathrm{n} \in \mathbb{N}_1$, are composed of the parameters that define the geometry and parameters that specify the conditions of the optimization problem, e.g. the angle of attack for an aerofoil optimization. A general minimization problem is expressed as:

$$\begin{array}{ll} \text{minimize} & \mathcal{I}\left(\boldsymbol{\chi}\right) \\ \text{over} & \boldsymbol{\chi} \in \mathbb{R}^{\mathrm{n}} \\ \text{subject to} & \boldsymbol{C}\left(\boldsymbol{\chi}\right) \leq \mathbf{0} \end{array} \tag{1.1}$$

Where $I\left(\boldsymbol{\chi}\right) : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}$, $\mathrm{n} \in \mathbb{N}_1$ is the objective function and $\boldsymbol{C}\left(\boldsymbol{\chi}\right) : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}^{\mathrm{m}}$, $\mathrm{m} \in \mathbb{N}_1$ is the vector of constraint functions. Various strategies exist for solving such an optimization problem. The most suitable strategy depends on the optimization problem that is considered.

A general optimization procedure is depicted in figure 1.2. An optimization always starts with the definition of the optimization problem, which involves the definition of the objective function, the constraint functions and the design space. Subsequently, the design is specified — or multiple designs, depending on the optimization method. Then the performance in terms of the objective function of each design is analysed. Based on this performance, the design is considered to be optimal or not. If the design is optimal, the optimization procedure is terminated. Otherwise, the design parameters are updated and the procedure is repeated.

1.5.1 Optimization methods

Optimization methods can be divided in two different classes, viz. zeroth-order methods and first-order methods. Zeroth-order methods only require the value of the objective

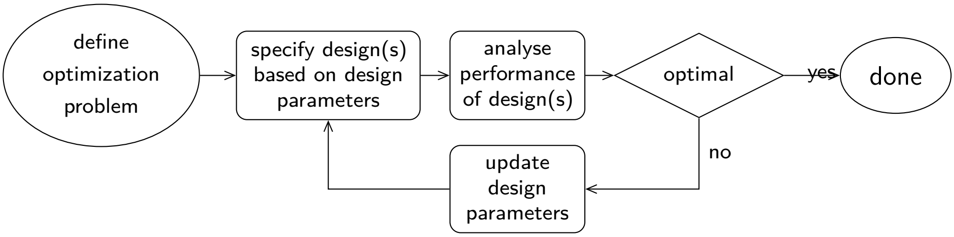

define optimization problem
specify design(s) based on design parameters
analyse performance of design(s)
optimal
yes
done
no
update design parameters

Figure 1.2: Flow chart of a general optimization procedure.

function and constraint functions. Since these methods do not require derivative information they are suitable for solving optimization problems with objective functions that are non-differentiable. An example of such an optimization problem involves the minimization of the sonic boom for supersonic aircraft [35, 43]. Moreover, zeroth-order methods are less sensitive to the existence of local optima in the design space. However, as the number of design variables increases, the number of function evaluations needed for reaching the optimum rapidly increases [113, 116].

First-order methods — also called gradient-based methods — do require derivative information during the optimization procedure. Therefore, they are not suited for application in optimization problems with non-differentiable objective or constraint functions. Moreover, when multiple (local) optima exist, the optimum found depends on the initial condition used to start the optimization procedure, which may not be the global optimum. The great strength of first-order methods comes from the use of the derivative information. This information allows to use a large number of design variables without the number of required function evaluations becoming prohibitively large.

For the optimization problem considered in the present research, the evaluation of the objective function requires at least one flow solution, i.e. a solution of the Euler equations. Therefore, it is expensive — in terms of required CPU-time — to evaluate the objective function. For this reason, the number of evaluations of the objective function should be limited. Furthermore, the accurate representation of the design space for a wind turbine blade optimization problem features a large number of design variables. Based on these two observations, a gradient-based optimization method is the method of choice for this research. Although this choice implies that the result of solving an optimization problem might yield a local optimum, results of solving aerodynamic design problems using a gradient-based approach indicate that the optimum obtained can be very close to the global optimum [94], in terms of the objective function[2].

1.5.2 Parametrization method

Solving a shape optimization problem requires a mathematical representation of the geometry to be optimized. This mathematical representation, also known as the parametrization method, must be able to represent the relevant part of the design space. Or in other words, the optimal design needs to be representable employing the chosen parametriza-

---

[2] An estimate of the value of the objective function for the global optimum can be obtained by considering physical bounds of the problem.

tion method. Moreover, the number of parameters required for this representation should be small, in order to limit the dimension of the design space and to limit the number of derivatives that needs to be computed. The dimension of the design space should be limited because the number of required optimization iterations to reach the optimum increases with the dimension of the design space [93].

In this research a non-uniform rational basis spline (NURBS) surface is used to represent the wind turbine blade geometry. A NURBS surface is used because it provides great flexibility for representing various shapes with a limited number of design variables. Moreover, NURBS surfaces are compatible with computer aided design (CAD) methods, which in the future will help to integrate the aerodynamic design process in the multidisciplinary wind turbine blade design procedure.

1.6 Sensitivity analysis

Sensitivity analysis refers to the estimation of the first directional derivative of one or more functions with respect to one or more independent variables [115]. For first-order optimization methods, the sensitivity analysis is an important aspect of the optimization procedure. Therefore, the computation of derivatives has to be carried out accurately. Several methods exist for the estimation of derivatives. These methods are treated in the following subsections and the advantages and disadvantages of their use are discussed.

1.6.1 Analytical differentiation

The most obvious method to estimate a derivative of a function is to differentiate the underlying function with respect to the independent variable of interest and subsequently compute the derivative by evaluation of the differentiated function. However, if the function to be differentiated becomes more involved, performing the differentiation also becomes more complicated. Therefore, this method is only feasible for functions that can be differentiated easily, which is generally not the case for the functions encountered in a flow method. Although, examples exist that this has been done for a 3D Euler method [31].

1.6.2 Finite-difference approximation

The most straightforward method to estimate the derivative is to use a finite-difference approximation of this derivative. The concept of this method can be explained by considering a Taylor series expansion of an analytic function $f(x) : \mathbb{R} \to \mathbb{R}$ to obtain $f(a+h)$ with $a, h \in \mathbb{R}$:

$$f(a+h) = f(a) + \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a h + \frac{1}{2}\left.\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right|_a h^2 + \frac{1}{3!}\left.\frac{\mathrm{d}^3 f}{\mathrm{d}x^3}\right|_a h^3 + \mathcal{O}\left(h^4\right). \tag{1.2}$$

The following first-order approximation of the first derivative can be obtained from this series expansion

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a = \frac{f(a+h) - f(a)}{h} + \mathcal{O}(h). \tag{1.3}$$

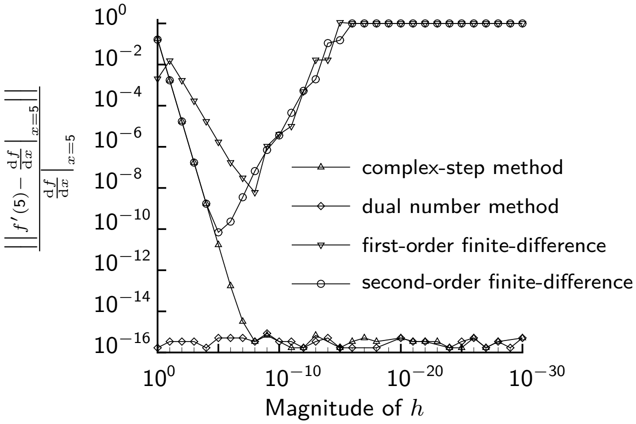


Figure 1.3: Relative error for various approximations of the derivative of the Bessel function of the first kind $J_0(x)$ evaluated at $x = 5$ for different step sizes h. The exact result is denoted by $\frac{\mathrm{d}f}{\mathrm{d}x}$ and the approximation by $f'(x)$. If the relative error is zero, no marker is present in the graph. Results show that the complex-step method reaches machine accuracy (employing double-precision arithmetic) for sufficiently small h and the accuracy of dual number method is independent of the step size. Results of both finite-difference methods illustrate the effect of truncation error for large h and cancellation error for too small h.

This expression indicates that an estimation of the derivative can be obtained by two evaluations of the function. Therefore, no explicit knowledge of the function that is evaluated is required to compute a finite-difference approximation of the derivative. This property is both an advantage of this method as well as a disadvantage. The advantage is that it can be applied as soon as a method for the evaluation of the function is available. The disadvantage is however, that the method requires additional evaluations of the function to estimate its derivative. Therefore, when evaluation of the function considered is expensive, computing the derivative using a finite-difference approximation is not a suitable method when a large number of derivatives must be determined. Moreover, the accuracy of the approximation depends on the magnitude of h. However, choosing h too small may result in the occurrence of subtractive cancellation errors, due to finite-precision arithmetic, which renders the result inaccurate, see figure 1.3. Accuracy of the finite-difference method can be increased by performing an additional function evaluation for $f(a-h)$. The central-difference approximation of the derivative then reads

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{a} = \frac{f(a+h) - f(a-h)}{2h} + \mathcal{O}\left(h^2\right). \tag{1.4}$$

This approximation has a second-order accuracy at the cost of one additional function evaluation — compared to the first-order finite-difference approximation — for each variable for which the derivative must be determined. Moreover, this method is still sensitive to subtractive cancellation errors, as illustrated in figure 1.3 as well.

1.6.3 Complex-step finite-difference method

A variation on the finite-difference concept is the complex-step finite-difference method [111, 175]. For an analytic function $f(z) : \mathbb{C} \to \mathbb{C}$ the Cauchy-Riemann equations apply [153]. Therefore, when f is expressed as $f(z) = u(z) + iv(z)$, $u, v : \mathbb{C} \to \mathbb{R}$ and $z \equiv x + iy$, $x, y \in \mathbb{R}$ the following equations must hold:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \tag{1.5}$$

Next, require $f(x) \in \mathbb{R} \; \forall \, x \in \mathbb{P}$ for the open subset $\mathbb{P} \Subset \mathbb{R}$. Then, consider the Taylor series expansion $f(z)$ for $z = a + ih$ with $a \in \mathbb{P}$, $h \in \mathbb{R}$

$$f(a + ih) = f(a) + i \left.\frac{\mathrm{d}f}{\mathrm{d}z}\right|_a h - \frac{1}{2} \left.\frac{\mathrm{d}^2 f}{\mathrm{d}z^2}\right|_a h^2 - \frac{i}{3!} \left.\frac{\mathrm{d}^3 f}{\mathrm{d}z^3}\right|_a h^3 + \mathcal{O}\left(h^4\right). \tag{1.6}$$

Using the definition of the derivative and considering that the direction in the complex plane from which zero is approached does not influence the result, $\epsilon \in \mathbb{R}$ can be chosen, such that

$$\left.\frac{\mathrm{d}f}{\mathrm{d}z}\right|_a = \lim_{\epsilon \to 0} \frac{f(a + \epsilon) - f(a)}{\epsilon} \equiv \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a. \tag{1.7}$$

Since $\mathbb{P}$ is an open set, $(a + \epsilon) \in \mathbb{P}$ for $\epsilon \to 0$. This observation means that both $f(a)$ and $f(a + \epsilon)$ are in $\mathbb{R}$, which also means that

$$\left.\frac{\mathrm{d}f}{\mathrm{d}z}\right|_a \equiv \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a \in \mathbb{R} \; \forall \, a \in \mathbb{P}.$$

Using this result and considering the Taylor series expansion of equation (1.6), it is found that the derivative of f with respect to x can be estimated by evaluating the expression

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a = \frac{\Im\left(f(a + ih)\right)}{h} + \mathcal{O}\left(h^2\right), \tag{1.8}$$

i.e. taking the imaginary part of the result — denoted by $\Im(\cdot)$ — for the function evaluated at $a + ih$, with ih an imaginary-valued disturbance of magnitude h and dividing this result by h. Since this expression does not involve a difference operation, the estimate obtained with this so-called complex-step derivative approximation [118] is not subject to subtractive cancellation errors. Therefore, the magnitude of h can be chosen extremely small — e.g. $\mathcal{O}\left(10^{-30}\right)$ — without losing accuracy due to finite-precision arithmetic, see figure 1.3. By choosing h sufficiently small this method effectively returns derivatives with an accuracy of machine precision, i.e. $\mathcal{O}\left(10^{-16}\right)$ for double-precision arithmetic [47] ; except when $\frac{\mathrm{d}f}{\mathrm{d}x}$ happens to tend to zero [118].

Like the finite-difference approximation, the disadvantage of the complex-step method is that it requires one additional evaluation of the objective function for each derivative that needs to be computed. Moreover, the execution of a function with complex arithmetic requires extra floating-point operations. Therefore, the complex-step method is more expensive than the finite-difference method, although it provides more accurate derivatives and is more stable. In contrast to the finite-difference method, the complex-step method requires intervention in the numerical method — in order to evaluate the functions with complex-valued numbers — making the application of the complex-step method more involved.

1.6.4 Dual number method

Yet another variation on the finite-difference concept is the dual number method. Before this method is presented, first the concept of dual numbers is explained. The set of dual numbers, $\mathbb{D}$, is defined as [54]:

$$\mathbb{D} := \left\{x + \varepsilon y \ : \quad x, y \ \in \ \mathbb{R}, \quad \varepsilon^2 \equiv 0\right\}. \tag{1.9}$$

In appearance, as well as in its properties, dual numbers, $x + \varepsilon y$, are quite similar to complex numbers, $x + iy$. One of the main differences is, however, that $\varepsilon^2 \equiv 0$, instead of $i^2 \equiv -1$ for complex numbers. This interesting property makes dual numbers very convenient to use in sensitivity analyses. For that purpose, require $f(x) \in \mathbb{R} \ \forall \ x \in \mathbb{E}$ for the analytic function $f(z) : \mathbb{D} \to \mathbb{D}$ and the open subset $\mathbb{E} \Subset \mathbb{R}$. Consider the Taylor series expansion of $f(z)$ for $z = a + \varepsilon h$ with $a \in \mathbb{E}$, $h \in \mathbb{R}$:

$$f(a + \varepsilon h) = f(a) + \varepsilon \left.\frac{\mathrm{d}f}{\mathrm{d}z}\right|_a h + \frac{\varepsilon^2}{2} \left.\frac{\mathrm{d}^2 f}{\mathrm{d}z^2}\right|_a h^2 + \mathcal{O}\left(\varepsilon^2\left(\varepsilon h^3\right)\right). \tag{1.10}$$

Since $\varepsilon^2 = 0$ by definition, this Taylor series expansion truncates at the first derivative. Analogous to the verification presented in the preceding subsection, it can be shown that

$$\left.\frac{\mathrm{d}f}{\mathrm{d}z}\right|_a \equiv \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a \in \mathbb{R} \ \forall \ a \in \mathbb{E}.$$

These interesting properties make dual numbers useful for computing derivatives, because the derivative can now be determined by evaluation of a function with a dual number and considering the non-real part only, i.e.

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_a \equiv \frac{\mathfrak{D}\left[f\left(a + \varepsilon h\right)\right]}{h}, \quad h \in \mathbb{R}. \tag{1.11}$$

Here $\mathfrak{D}(\cdot)$ indicates taking the non-real part of the dual number argument, equivalent to $\mathfrak{I}(\cdot)$ taking the imaginary part of a complex-valued number. Since all terms of the Taylor series expansion of order 2 and higher are exactly equal to zero, no truncation error occurs and therefore the choice of the value of h is arbitrary, which can be observed in figure 1.3. Choosing h equal to 1.0 is a convenient choice.

The advantage of the dual number method is that this method renders derivatives up to machine accuracy, just as the complex-step finite-difference method [60, 61]. Another similarity between this method and the other two finite-difference methods is the requirement of an additional function evaluation to compute a single derivative. Therefore, this method is also not very efficient for a large number of independent variables. As with the complex-step method, the dual number method requires intervention in the computational method in order to make it use and handle dual number arithmetic. More details on dual number arithmetic and the dual number method are given in section 6.1.

1.6.5 Algorithmic differentiation

Algorithmic differentiation [75] — also known as automatic differentiation — is considerably different from the methods discussed so far. This method is based on the systematic application of the chain rule of differentiation to each operation in a computer program [115].

Every computational method consists of a sequence of combinations of unary and binary operations[3], performed on the independent input parameters $\boldsymbol{x}$ and the subsequent intermediate results, leading eventually to a final result $\boldsymbol{f}$. Each of these operations can be differentiated, which is done by the method that performs the algorithmic differentiation. The result of applying an algorithmic differentiation method is a new algorithm that computes the derivatives of $\boldsymbol{f}$ with respect to the independent variables $\boldsymbol{x}$. This result can either be achieved by means of source code transformation or by so-called operator overloading. Both methods will be discussed briefly. Because each operation in the original program is differentiated exactly, the resulting differentiated method yields a derivative accurate up to machine accuracy.

Source code transformation

An algorithmic differentiation method that applies source code transformation to generate the algorithm that computes the derivatives first parses the original source. This approach is similar to that of a compiler, although a compiler generates machine code, while the algorithmic differentiation method generates a new source code which can be used to compute the derivatives.

Algorithmic differentiation by means of source code transformation distinguishes two different so-called modes. The forward mode produces source code that performs each step of the algorithm in the same order as the original source. The resulting new source computes the derivative of each output $\boldsymbol{f}$ with respect to a single input variable. On the other hand, the reverse mode, as the name implies, produces a new source that performs the steps of the original algorithm in the reverse order. The effect of performing the steps in the reverse order is that the resulting new function computes the derivative of a single output variable with respect to all input parameters $\boldsymbol{x}$. The consequence of computing the derivatives in the reverse direction is that intermediate results need to be stored in case these intermediate results are overwritten in the original source. This requirement for storing intermediate values makes the reverse mode more memory intensive than the forward mode. Which one of these methods is most efficient, depends on the ratio between the number of input and the number of output parameters. If a function has more output parameters than input parameters, the forward mode is more efficient, while the reverse mode is the better choice if the number of input parameters exceeds the number of output parameters.

This method generates new source code which computes the derivatives. A disadvantage of this approach is that the code that is generated can be quite unreadable [115], which is especially true when the reverse mode has been used. Therefore, maintenance of the code and performing modifications can be troublesome. An example of an algorithmic differentiation method that performs source code transformation is the program called Tapenade [80], developed at INRIA. This method can be applied to programs written in either Fortran or C.

Operator overloading

The other approach for generating an algorithmically differentiated code is by means of operator overloading. Operator overloading means that the unary and binary mathemati-

---

[3] Note, that in some programming languages the ternary operator `?:` also exists, which is essentially a conditional expression.

cal operations in an algorithm are redefined for a new user-specified data type. This data type is used to replace the data type representing real numbers in the original source. Redefinition or overloading of the operators is performed in such a way that when an operator is applied to a particular input, the operations to compute the derivative of the output with respect to the input are also directly applied. The result, i.e. both the value as well as the derivative, is then stored. To use this approach, the programming language that is employed needs to have the ability for the user to redefine these mathematical operations. Fortunately, this is true for a significant subset of modern programming languages [116], e.g. C++, Fortran and Python.

Since this method only requires the use of a new data type — for which the operators have been redefined — instead of the data type for real numbers, in order to realize the algorithm that computes the derivatives, this approach has less impact on the readability of the resulting source code. A disadvantage of operator overloading is that the resulting algorithm is generally less efficient than when source code transformation is used [144]. The difference in efficiency is because for operator overloading for all steps in the algorithm the derivative is computed, also for operations that do not influence the final result. With source code transformation, computing the derivative for these operations is omitted, resulting in fewer mathematical operations required to be performed to generate the final result.

1.6.6 Adjoint equation method

The methods discussed above can be generally applied in any situation in which derivatives need to be computed. For the method discussed in this section, i.e. the adjoint equation method, however, an additional requirement needs to be met, in order to be able to use it for computing sensitivities.

Consider the analytic functional $f\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right):\mathbb{R}^{\mathrm{m}}\times\mathbb{R}^{\mathrm{n}}\rightarrow\mathbb{R}$, $\boldsymbol{u}\left(\boldsymbol{x}\right):\mathbb{R}^{\mathrm{n}}\rightarrow\mathbb{R}^{\mathrm{m}}$, $\mathrm{m},\mathrm{n}\in\mathbb{N}_1$ and the vector of analytic constraint functionals $\boldsymbol{g}\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right):\mathbb{R}^{\mathrm{m}}\times\mathbb{R}^{\mathrm{n}}\rightarrow\mathbb{R}^{\mathrm{m}}$, which must satisfy $\boldsymbol{g}\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)=\boldsymbol{0}$. This additional requirement is not as restrictive as it may seem. Many mathematical problems are governed by partial differential equations (PDE). For these problems, functional $\boldsymbol{g}\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)$ can be viewed as a representation of these governing partial differential equations.

Since $\boldsymbol{g}\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)$ equals zero, a new functional can be formulated

$$F\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)=f\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)+\boldsymbol{\lambda}^T\boldsymbol{g}\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right), \tag{1.12}$$

which evaluates to the same result as the original functional $f\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)$, independent of $\boldsymbol{x}$, as long as the requirement for the vector of constraint functionals $\boldsymbol{g}\left(\boldsymbol{u}\left(\boldsymbol{x}\right),\boldsymbol{x}\right)$ is met. In this equation $\boldsymbol{\lambda}\in\mathbb{R}^{\mathrm{m}}$ is the vector of so-called Lagrange multipliers. The total derivative of functional F with respect to $\boldsymbol{x}$ reads

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}=\left[\frac{\partial f}{\partial\boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{x}}+\frac{\partial f}{\partial\boldsymbol{x}}\right]+\boldsymbol{\lambda}^T\left(\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{x}}+\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{x}}\right)+\frac{\mathrm{d}\boldsymbol{\lambda}^T}{\mathrm{d}\boldsymbol{x}}\boldsymbol{g}. \tag{1.13}$$

rearranging this equation, and considering that $\boldsymbol{g}\equiv\boldsymbol{0}$, gives

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}=\left[\frac{\partial f}{\partial\boldsymbol{u}}+\boldsymbol{\lambda}^T\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{u}}\right]\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{x}}+\left(\frac{\partial f}{\partial\boldsymbol{x}}+\boldsymbol{\lambda}^T\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{x}}\right). \tag{1.14}$$

The matrix $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{x}}$ can be eliminated from this expression, by requiring the part between the square brackets to be equal to the null vector. This requirement is met, when $\boldsymbol{\lambda}$ satisfies

$$\left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\right]^T \boldsymbol{\lambda} = -\left[\frac{\partial f}{\partial \boldsymbol{u}}\right]^T . \tag{1.15}$$

When $\boldsymbol{\lambda}$ is determined, by solving equation (1.15), the total derivative of f is computed by evaluation of

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}} \equiv \frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{x}} + \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \tag{1.16}$$

This method requires the solution of the system of linear equations of equation (1.15) in order to find $\boldsymbol{\lambda}$ and subsequently compute the derivatives. However, this procedure thereby eliminates requiring the computation of $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{x}}$, which can be a computationally intensive procedure; especially, when $\boldsymbol{u}$ is governed by a set of partial differential equations.

The adjoint equation method provides derivatives up to machine accuracy. Moreover, the computational requirements of the method are largely independent of the dimension of $\boldsymbol{x}$. In contrast to the methods discussed in sections 1.6.2 to 1.6.5, for which the computational requirements are proportional to the dimension of $\boldsymbol{x}$ — except for analytical differentiation and to some extent algorithmic differentiation. A major drawback of the adjoint equation method is, however, that the implementation is considerably more complicated [115] than in particular the finite-difference method. The reason for it is that considerable knowledge of how both f and $\boldsymbol{g}$ are computed is required in order to compose equation (1.15), while for instance for the finite-difference method the evaluation of both functions can be treated as a 'black box'.

Different approaches exist on how the adjoint equations are handled. For the continuous adjoint equation method, the constraint functions are first linearized. These linearized equations are subsequently discretized, in order to solve them. On the other hand, for the discrete adjoint equation method, discretization of the constraint functions is performed before the linearization is carried out. Each approach has its own implications regarding the implementation of the method, which is discussed in more detail in section 6.2.2.

1.6.7 Approach

Various methods for the sensitivity computation have been presented. Each of these methods has its advantages and disadvantages. The approach taken in the present research is to combine these methods for the computation of the sensitivities of the objective function with respect to the design parameters. In this way, full advantage is taken of the strength of each method. The optimization problems considered permit the use of the adjoint equation method, which is the most efficient method provided the number of design parameters is large compared to the number of non-linear functions in the problem considered. The partial derivatives, appearing in the adjoint equations are computed using the dual number method. In this way, the partial derivatives are computed accurately, resulting in an accurate gradient of the objective function. A more detailed discussion of the approach is presented in chapter 6.

1.7 Implementation on parallel computers

Solving an aerodynamic shape optimization problem requires multiple flow solutions. Therefore, in order to realize a feasible method, the starting point has been that the method must be executed in parallel, such that no sequential bottle-necks are present. In that way the algorithm can be executed efficiently on a cluster of computers. For this purpose the software library Message Passing Interface (MPI) [182] is used.

For an efficient parallelization of the method, it is of key importance that: (i) the amount of work is distributed evenly over the available processor cores, (ii) the number of communications between the processor cores is limited and (iii) the amount of data being communicated between the processor cores is limited as well. However, these factors depend on the task that is to be performed. For instance, to determine the overset block connectivity, the amount of work is proportional to the number of control volumes for which a donor needs to be found, while for obtaining a flow solution, the amount of work is proportional to the number of control volume faces over which the flux needs to be computed. Similar differences exist for the amount of data that needs to be communicated and the number of communications that needs to be performed for the different tasks.

As obtaining the flow solution has a major contribution to the overall computation time, the focus has been on the efficiency of the parallelization of the flow solution method. Therefore, the blocks used for the discretization of the flow domain are split[4] and distributed in such a way that the amount of work for obtaining a flow solution is spread evenly over the available cores — assuming the task being performed on a homogeneous cluster, i.e. a cluster consisting of compute nodes that have the same properties in terms of clock speed and other performance related parameters. Such a block, assigned to a single core is denoted a *compute block*. Moreover, splitting is done such that the required amount of communication is minimized too. The resulting decomposition is also used for the other tasks in the optimization procedure; except for the grid generation. For the grid generation it is in general not possible to use the same decomposition as for the flow solution method, because in the hyperbolic grid generation procedure, the previous layer of the grid is required to determine the next layer. Therefore, a different decomposition is used for the grid generation. In that case, the number of grid blocks that must be generated is distributed evenly over the available processor cores without splitting them; and one block is always generated by a single processor.

1.8 Related work

Numerical aerodynamic shape optimization started with the work of Vanderplaats et al. [192]. Vanderplaats et al. used a gradient-based optimization method for the optimization of aerofoils. Gradients were computed using a finite-difference approximation. Pironneau [140,141] suggested a different approach for computing the sensitivities, i.e. by means of the adjoint equations. This approach renders computing the sensitivities to be largely independent of the number of design parameters. Later on, the same approach was adopted by Jameson [90], using a continuous adjoint equation method. Since these pioneers introduced numerical aerodynamic shape optimization, much research effort has

---

[4] Assuming that the number of cores exceeds the number of blocks.

been devoted to the subject and many researchers made their contribution to this fascinating research area. In this section, the contributions are presented in some detail, since large portions of the work presented in this thesis have been inspired on these contributions.

Nielsen implemented the discrete adjoint equation method in an unstructured RANS method [130]. The solution of the adjoint equations was subsequently used for determining the sensitivities for various optimization problems. Results of aerodynamic shape optimization problems have been presented for, amongst others, the drag reduction of a wing in transonic flow — considered both using a viscous as well as an inviscid flow model — starting with an ONERA M6 wing as initial design. Since the initial implementation, the optimization method has been extended to solve aerodynamic optimization problems for unsteady flow as well [131, 132].

Another discrete adjoint equation method was implemented by Carpentieri [31]. In this case an inviscid flow model was considered. Unstructured grids were used for the discretization of the flow domain. Distinctive features of the method presented are the implicit time-integration and the possibility to compute the adjoint vector for different non-linear constraint functions simultaneously. This approach increases the efficiency of the method in terms of required CPU-time, with respect to handling the adjoint vectors sequentially. The method was tested for a similar case as the drag minimization problem of a wing in transonic flow presented by Nielsen [130], also using an ONERA M6 wing as starting geometry.

The research presented by Marta features a novel method to compute partial derivatives for the adjoint equations [115]. In the method, an algorithmic differentiation method — by means of source code transformation — is applied to large portions of the original procedures that are used to compute the flow solution. Utilizing this approach it is possible to greatly reduce the time required to develop an adjoint equation method for an arbitrary numerical method. The particular implementation was only suitable for using non-geometrical design variables, such as angle of attack. The method was subsequently tested by solving several magnetohydrodynamics (MHD) optimization problems.

The use of the adjoint equations for performing gradient-based aero-structural optimization was investigated by Martins [116]. Focus of the research was on coupling of the adjoint equations for the two different disciplines. The procedure was tested by performing an aero-structural optimization of a supersonic business jet. The work also presents an elaborate and clear description on the implementation of the complex-step finite-difference approximation for computing partial derivatives, for which the concept has been discussed briefly in section 1.6.3.

The effect of applying approximations to some of the terms involved in the adjoint equations on the resulting sensitivities was investigated by Dwight [53]. The research also investigated the effect of the approximations on the result of the optimization. For this purpose 2D cases in turbulent flow were considered. Focus of the research was, however, on the improvement of the efficiency of the flow solution method, in terms of CPU-time, by implementing an implicit solution method.

Hicken used an Euler based discrete adjoint equation method to investigate the potential of using an inviscid flow model for minimization of induced drag [82]. This investigation was performed for different wing configurations, including configurations with one or multiple winglets. The adjoint equations were solved using a specially developed Newton-Krylov method.

Other subjects, not directly related to gradient-based optimization, but which have been of great importance for the successful implementation of the optimization method presented in this thesis are hyperbolic grid generation and composite overset grid connectivity. With respect to the application of composite overset grids notable work has been performed by Zahle, who implemented an overset method in the RANS method for incompressible flow EllipSys3D [206]. Due to the nature of the incompressible flow model, special care needed to be taken in transferring the pressure between the different overlapping grids. The method was used to simulate the flow around a wind turbine rotor and to investigate the effect of the presence of the tower on the flow and on the performance of the rotor.

A similar feat as that of Zahle was achieved by Schwarz, who had a major contribution in the implementation of an overset method in the TAU code of DLR [160]. In the method presented, special care was taken to ensure the transfer of the correct flow solution in regions near curved solid walls. Flow simulations were performed for, amongst others, the flow around an aircraft in landing configuration.

The development of the implicit hole cutting — used in the present research for the overset block connectivity — can largely be attributed to Lee [107], who applied the method to rotorcraft aerodynamics. A significant contribution to developing the concept of zipper grids — used for the accurate evaluation of surface integrals — was done by Chan [37]. Chan also carried out much research on the subject of hyperbolic grid generation [36, 39]. That work was based on the work of Steger [176, 177], who introduced the concept of hyperbolic grid generation.

1.9 Thesis outline

In the remainder of this thesis a more thorough consideration is given of each of the aspects important in an aerodynamic shape optimization method. The first aspect considered is the shape parametrization method to be used for representing the wind turbine blade geometry, which is discussed in chapter 2. Subsequently, in chapter 3 the method used for the discretization of the flow domain is presented. Chapter 4 treats the method used for determining the block connectivity of the composite overset grid. The flow model and the corresponding solution method are subject of chapter 5; in this chapter the discretization of the governing equations — both in space and time — is discussed in more detail. Moreover, results of the case considered to verify the correct implementation of the flow model is presented in this chapter as well. Chapter 6 is dedicated to sensitivity analysis, providing, amongst others, more details on the concept of dual numbers. The discrete adjoint equation method is also discussed in this chapter. The approach taken in computing the partial derivatives involved in these equations is outlined as well and the accuracy of the derivatives computed by this method is verified. In the subsequent chapter, chapter 7, results are presented of the optimization method. First, the general concept of the optimization method is discussed. Subsequently, some test cases are considered to demonstrate the capabilities of the optimization method. The final chapter of this thesis states the conclusions drawn from the present investigation. Also recommendations for extension and improvement of the method are pointed out.

2

SHAPE PARAMETRIZATION

> *"If they would, for Example, praise the Beauty of a Woman, [...] they describe it by Rhombs, Circles, Parallelograms, Ellipses, and other geometrical terms [...]."*
>
> — JONATHAN SWIFT (1667 – 1745), *Gulliver's Travels*

THE choice of the parametrization method for the geometry in an aerodynamic shape optimization problem is a key aspect in the effectiveness and efficiency of the optimization procedure [65, 114]. This chapter considers the desired characteristics for the parametrization in a general gradient-based aerodynamic optimization problem. Based on this specification, a suitable choice is made for the parametrization method of the wind turbine blade geometry. The method chosen is elaborated on in more detail, especially regarding the mathematical formulation of the particular parametrization method. Subsequently, an investigation is performed on the number of design variables required to accurately represent a wind turbine blade geometry. In addition, it has been determined how this number relates to the number of design variables that is required for a sufficiently accurate representation of the design space for solving a model aerodynamic shape optimization problem.

2.1 Introduction

This section formulates the requirements for the parametrization method of the geometry in an aerodynamic shape optimization problem. A brief overview is given on the parametrization methods that have been employed — and presented in the literature — for this kind of problems. Finally, the parametrization method is considered that will be applied in the present research for the parametrization of the wind turbine blade geometry.

2.1.1 Requirements

Before the choice of a method for the parametrization of the wind turbine blade geometry can be made, it is important to consider the key aspects and requirements for the parametrization employed for solving an aerodynamic shape optimization problem.

First and foremost, the parametrization method must be able to span the complete design space: 'complete' in the sense that any geometry with a topology similar to the topology of the initial geometry must have a representation in the parametrization method that is considered. This requirement is important, since it would be very undesirable if the optimal geometry could not be obtained by applying the optimization method, because it cannot be represented by that parametrization method.

As a second requirement, the number of design variables required to satisfy the first requirement should be limited. This statement is not very precise, because in the present context it is hard to give an absolute quantification of the term 'limited'. However, this requirement can be used to measure the suitability of one parametrization method relative to another. Limiting the number of design variables is important for an efficient optimization procedure, because an increase in the number of design variables increases the computational cost per iteration. Moreover, it results in an increase in the number of optimization iterations that must be performed to reach the optimum. The latter statement can be explained by considering the dimension of the design space in which the optimum is to be searched for. In general, a search space of a high dimensionality requires more optimization iterations before the actual optimum is obtained than a search space with fewer dimensions. Results of an investigation of a model optimization problem [93] confirm this idea. Secondly, a large number of design variables may restrict the choice of the gradient-based optimization method to be used. A Newton-like method is the preferred choice, because of the low number of design iterations needed to obtain an optimum. However, for a large number of design variables, the matrix involved in the Newton-like method becomes prohibitively large and one is forced to apply less efficient gradient-based optimization strategies [71], requiring a larger number of design iterations. Thirdly, it must be noted that for a gradient-based optimization, the larger the number of design variables, the more sensitivities have to be determined. Even for the adjoint equation method, which is employed because its characteristics do not depend on the number of design variables, it is advantageous to use a compact parametrization. Because the adjoint equation method requires the evaluation of a number of partial derivatives. The evaluation of these partial derivatives may depend on the number of design variables, depending on the particular implementation. Therefore, the computational cost per iteration increases with the number of design variables.

Another requirement of the parametrization method is that varying the design variables should result in local changes of the geometry. In this way, the geometry can be modified locally without affecting regions that do not need to be changed any more in order to render the optimal shape.

The final requirement concerns the behaviour of the parametrization during the optimization procedure, this behaviour must be smooth [31]. This requirement is formulated to prevent the optimization procedure from failing, because of the occurrence of geometrical features — like sharp dents in the surface — that present problems for the discretization of the flow domain. Just like the requirement for a limited number of design variables, this requirement is not easily quantified. However, by envisaging the general characteristics of the parametrization method, the method can be qualified to display smooth behaviour or not; for example, by considering the degree of continuity of the parametrization employed.

2.1.2 Methods employed in aerodynamic shape optimization

In the literature there are many examples of parametrization methods used in aerodynamic shape optimization problems. An overview of the various existing methods adopted is presented below. However, this overview is not meant to be complete, but merely gives a good indication of what the options are.

For most optimization procedures presented in the literature, just one particular parametrization method is adopted, without explicitly considering the alternatives. One of the most straightforward means of the parametrization of an arbitrary shape is using a discrete representation of its geometry. The vertices of the surface mesh can be used for this purpose. In the work of both Jameson [90] and Mohammadi [125] this parametrization method was employed. The main advantage of this method is that it is easy to apply. However, its disadvantage is that the method is not inherently smooth, and therefore additional smoothing of the obtained geometry may be required. Furthermore, since the coordinates of the vertices of the surface mesh are the actual design variables, combined with that a considerable number of vertices is necessary for an accurate representation of a curved shape, the number of design variables becomes rapidly large for this method.

An alternative approach was taken by both Gauger [67] and Kim [97]. They employed so-called Hicks-Henne bump functions [84]. In this method, the actual shape is not parametrized, but the change in shape with respect to the initial geometry. This method requires less design variables than the mesh point approach to represent the geometry. Also this method results in smoother geometric shapes.

Carpentieri [31] employs a similar parametrization method, by using Chebychev polynomials instead of the Hicks-Henne bump functions. This approach has the additional advantage that the Chebychev polynomials are orthogonal, which could result in better convergence properties of the optimization process. Although Chebychev polynomials have originally been applied for 2D parametrization only, Carpentieri also presents an extension of the concept for the parametrization of 3D shapes. This method was subsequently used for the optimization of wings in inviscid flow.

Other widely used parametrization methods in aerodynamic optimization problems employ splines to describe the aerofoil or wing geometry. These methods generally provide a smooth parametrization and require only a limited number of design variables. For splines, control points are used to specify the shape of a curve. In this case, the location of these control points, and in some cases also the corresponding weights, represent the design variables. Furthermore, a collection of basis functions is used to convert the control point locations to the actual curve shape. Different kinds of splines are used for this same basic concept, such as the basis spline, successfully employed by Buckley [26] for the multi-point optimization of aerofoils. Another type of spline is known as the Bézier spline, which has been used for the optimization of transonic aerofoils [170]. The most general type of spline is the so-called non-uniform rational basis spline (NURBS) [186], which can be used for the parametrization of curves as well as of surfaces of almost any shape. This parametrization method has been used, for example, for the optimization of aircraft wings [109].

For 2D aerodynamic optimization problems the parametrization method PARSEC [169] is widely used. This method employs general geometrical characteristics of an aerofoil, such as the nose radius and trailing edge angle, to describe the geometry. These geometrical features are subsequently used to impose boundary conditions on a more general curve parametrization method, e.g. polynomials [135] or Bézier splines [50]. However, there is no generalization of this concept for the parametrization of 3D wing-like shapes and extension of the method to be able to describe a true 3D wing-like shape is not trivial. Therefore, it is not considered further in the present research. Moreover, Castonquay showed that only limited control over important regions, such as the leading edge and

Table 2.1: Summary of the most widely used parametrization methods in aerodynamic shape optimization problems and their most important characteristics. Favourable characteristics are indicated by a '+' and adverse characteristics by a '−'.

Method		Major characteristics
Mesh points	+	easy to apply
	−	not inherently smooth
	−	large number of design variables
Hicks-Henne bump functions/ Chebychev polynomials	+	smooth
	+	small number of design variables
	−	not a parametrization of the actual shape
Splines	+	inherently smooth
	+	local influence
	+	small number of design variables
PARSEC	−	limited to 2D
	+	small number of design variables
	+	smooth

trailing edge shape, is possible [33].

In the literature, various parametrization methods have been compared and assessed for their suitability for a particular optimization problem. Castonguay [33], for example, compares different methods for the optimization of aerofoils in transonic flow. For that particular optimization method, NURBS curves are considered the most suitable choice. Samareh [155] gives a more general overview of the various shape parametrization methods employed in aerodynamic shape optimization and also lists the reasons to use a particular method for a specific problem; this choice depends on the requirements of the problem considered. Another comparison for the optimization of aerofoils is presented in the research performed by Marinus [114], in which the parametrization by means of Bézier curves is compared with parametrization based on the use of basis splines. This research indicates that basis splines surpass Bézier curves in performance. Three different 3D parametrization methods are compared in the work by Mousavi [128], namely mesh points, basis spline surfaces and the so-called Class function/ Shape function Transformation [105] (CST) method. The latter method combines an analytical class function with a parametric shape function. The class function describes a basic class of shapes and the shape function describes the permutation around this basic shape [179]. The authors conclude that the mesh points show a slightly better performance than the basis spline surface for the test-cases performed. However, it is also noted that considering the number of design variables employed, the basis spline surface is the preferred choice. Additional investigations on the influence of shape parametrization on the optimization procedure can be found in the literature [135, 171, 204]. Table 2.1 summarizes the major parametrization methods used in aerodynamic optimization problems and lists their most important features.

2.1.3 Considerations

The choice has been made to use non-uniform rational basis spline surfaces to parametrize the wind turbine blade shape. This choice was made based on: (i) a NURBS surface parametrization can meet all the requirements mentioned in subsection 2.1.1; (ii) the experience and results presented in the literature; (iii) compatibility of the NURBS parametrization with computer aided design (CAD) methods. Because NURBS surfaces are widely used in CAD methods for representing arbitrary shapes, it is an expedient choice to use a NURBS surface for representing the wind turbine blade shape; since this approach helps to integrate the aerodynamic design process in the multi-disciplinary wind turbine blade design procedure.

When choosing a NURBS surface, there is one more choice that needs to be considered for the actual blade surface representation. The blade shape can be represented by the NURBS surface itself, such that the NURBS surface is equal to the blade surface. The alternative is to decompose the blade shape in a way that is analogous to the representation of an aerofoil by its camber line and thickness distribution, i.e. the method also used to describe the geometry of the well-known NACA aerofoils. The extension of this concept to 3D means that the thickness distribution is to be represented by a surface rather than a curve. This statement is also true for the camber line, which becomes a camber surface. Complementing this transformation, one additional curve is required to embody the twist distribution along the span of the blade. The direct parametrization of the blade surface has the advantage that it can also be employed directly in the generation of the surface grids. On the other hand, it was found by Marinus that, depending on the optimization method used, the indirect approach outperforms the direct parametrization regarding the required number of design iterations [114]. On the other hand, the indirect approach is incompatible with CAD methods, while the direct approach is. Furthermore, since the optimization procedure that led Marinus to this conclusion strongly differs from the planned approach in the present research, it can not be assumed to be also true in this case. Moreover, reusing the blade surface parametrization in the surface grid generation procedure is regarded a significant advantage of the direct approach. Therefore, the choice is made to employ a non-uniform rational basis spline surface for the direct parametrization of the surface of the blade of the wind turbine rotor. Nevertheless, it is interesting to investigate the performance of both methods, which, however, is not pursued in the present research.

2.2 Non-uniform rational basis spline surface

The parametrization method chosen to describe the wind turbine blade geometry is the non-uniform rational basis spline surface. In this section, the method is described, starting with the concept of the basis spline. Subsequently, this concept is extended to the rational basis spline. Thereafter, the extension from a curve to a surface is presented in section 2.2.3.

2.2.1 Basis spline curve

A basis spline curve is specified by a set of control points, with coordinate $\boldsymbol{P}_\mathrm{k} \in \mathbb{R}^3$, the order $\mathrm{p} \in \mathbb{N}_1 \geq 1$ of the spline and the so-called knot vector $\boldsymbol{\upsilon}$. For a basis spline curve of order p specified by $\mathrm{n} \in \mathbb{N}_1 \geq 2$ control points, the knot vector contains a sequence of $\mathrm{p}+\mathrm{n}$ non-decreasing real valued numbers:

$$\upsilon_0 \leq \upsilon_1 \leq \ \ldots \ \leq \upsilon_{\mathrm{p}+\mathrm{n}-1} \ , \quad \upsilon_{\mathrm{i}} \in \mathbb{R} \, \forall \, \mathrm{i} \in \mathbb{N}_0 : 0 \leq \mathrm{i} \leq \mathrm{p}+\mathrm{n}-1.$$

Without loss of generality it can be assumed that $0 \leq \upsilon_{\mathrm{i}} \leq 1$. When all the components of the knot vector and the coordinate of all control points are defined, the shape of the basis spline curve is given by

$$\boldsymbol{C}^{\mathrm{p}}(u) := \sum_{\mathrm{i}=0}^{\mathrm{n}-1} \left[\boldsymbol{P}_{\mathrm{i}} N_{\mathrm{i}}^{(\mathrm{p}-1)}(u) \right] \ , \quad \upsilon_0 \leq u \leq \upsilon_{\mathrm{p}+\mathrm{n}-1}, \tag{2.1}$$

where $N_{\mathrm{i}}^{\mathrm{q}}$ is a polynomial basis function of degree $\mathrm{q} \in \mathbb{N}_0$. These polynomial basis functions are given by the Cox-de Boor recursion formula [139]:

$$N_{\mathrm{i}}^{\mathrm{q}}(u) := \frac{u - \upsilon_{\mathrm{i}}}{\upsilon_{\mathrm{i}+\mathrm{q}} - \upsilon_{\mathrm{i}}} N_{\mathrm{i}}^{(\mathrm{q}-1)}(u) + \frac{\upsilon_{\mathrm{i}+\mathrm{q}+1} - u}{\upsilon_{\mathrm{i}+\mathrm{q}+1} - \upsilon_{\mathrm{i}+1}} N_{\mathrm{i}+1}^{(\mathrm{q}-1)}(u) \ , \quad \mathrm{q} \geq 1. \tag{2.2}$$

The recursive relation starts for $\mathrm{q} \equiv 0$ with a piece-wise constant function, also known as the boxcar function [23], defined as:

$$N_{\mathrm{i}}^{0}(u) := \begin{cases} 1 & \text{if} \quad \upsilon_{\mathrm{i}} \leq u < \upsilon_{\mathrm{i}+1} \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

This equation concludes the mathematical model to represent curves in $\mathbb{R}^3$ by means of a basis spline curve. Some general properties of basis functions can be identified, which are listed below.

Consider basis functions of degree q for a curve specified by n control points:

1. sum equal to one, i.e. $\sum_{\mathrm{i}=0}^{\mathrm{n}-1} \left[N_{\mathrm{i}}^{\mathrm{q}}(u) \right] \equiv 1 \ , \quad \upsilon_0 \leq u \leq \upsilon_{\mathrm{q}+\mathrm{n}}$;
2. are non-negative, i.e. $N_{\mathrm{i}}^{\mathrm{q}}(u) \geq 0 \ , \quad \forall \, u \in \mathbb{R}$;
3. equal zero outside their range of influence, i.e. $N_{\mathrm{i}}^{\mathrm{q}}(u) \equiv 0$ if $u \notin [\upsilon_{\mathrm{i}}, \upsilon_{\mathrm{i}+\mathrm{q}+1}]$.

Figure 2.1 illustrates basis functions of a different degree and with different knot vectors. Non-uniformity of the knot vector is treated next.

Knot vector

The knot vector can either be uniform or non-uniform. The latter provides greater flexibility for the curve parametrization [186], since the knot vector — in combination with the order of the curve — determines the amount and range of influence of a control point. A special case of non-uniformity in the knot vector is the occurrence of a repeated knot value, also known as knot multiplicity. A direct consequence of the occurrence of a repeated knot value in the knot vector is the reduction of the parametric continuity of the curve by one order for each repetition of a single knot. To be more precise, for a basis spline curve of order p with a knot vector with a maximum knot multiplicity k, the parametric continuity is of the order $\mathrm{p}-\mathrm{k}$. Knot multiplicity can, for instance, be

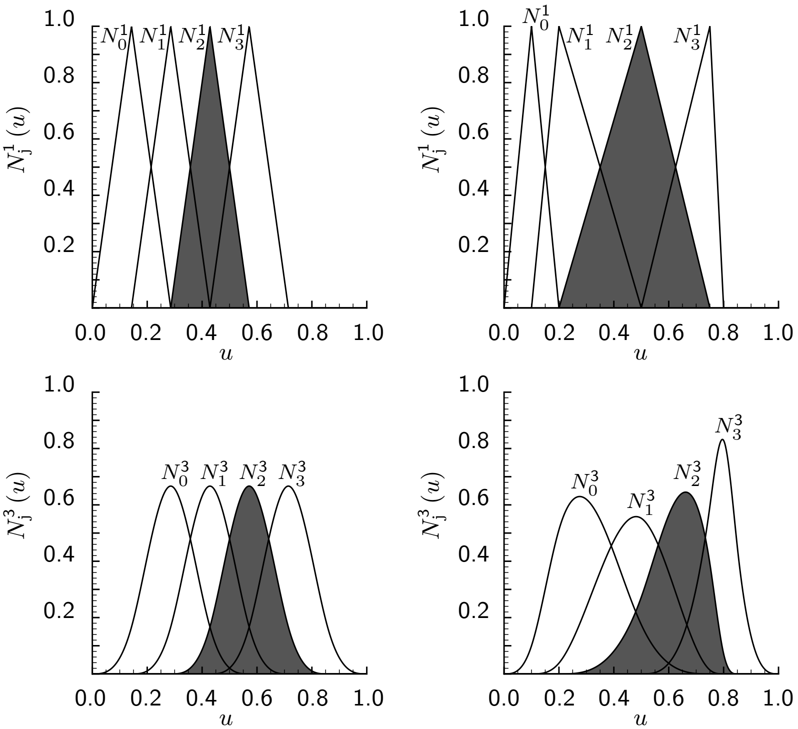


Figure 2.1: Basis functions of degree 1 (upper two) and 3 (lower two) for a spline defined by 4 control points. Basis functions depicted on the left employ a knot vector with uniform spacing of $\frac{1}{7}$. For basis functions depicted on the right, the non-uniform knot vector $\left(0, \frac{1}{10}, \frac{1}{5}, \frac{1}{2}, \frac{3}{4}, \frac{4}{5}, \frac{17}{20}, 1\right)^T$ has been employed. In each graph the area underneath basis function $N_2^{\mathrm{p}}(u)$ is shaded.

used to introduce a sharp corner, i.e. a discontinuity in the slope in the curve; see for instance figure 2.2 on page 24. In this figure the solid curve represents an aerofoil and it is specified using 8 control points — indicated by the open circles, which are connected by the control polygon indicated by the dashed line — and has basis functions of degree four. A knot multiplicity of 5 is applied at the trailing edge, resulting in a curve that has locally only a C^0-continuity. However, when two or more elements of the knot vector are of equal value, the problem of division by zero arises in the evaluation of equation (2.2). To cope with this difficulty, in such cases the following convention is applied:

$$\frac{0}{\upsilon_{\mathrm{i+q}} - \upsilon_{\mathrm{i}}} := 0 \quad \text{if} \quad \upsilon_{\mathrm{i+q}} \equiv \upsilon_{\mathrm{i}}.$$

In general, the control points of a basis spline curve are not located on the curve itself. However, if such an intersection is required for control point $\boldsymbol{P}_{\mathrm{i}}$ of equation 2.1, it can be achieved by the application of a knot multiplicity equal to the order p of the curve for the elements of the knot vector ranging from element i to $\mathrm{i} + \mathrm{p} - 1$. This approach is customary for the first and final control point that define the curve. In this way, the beginning and end of the curve are fixed, as illustrated by the aerofoil depicted in figure 2.2.

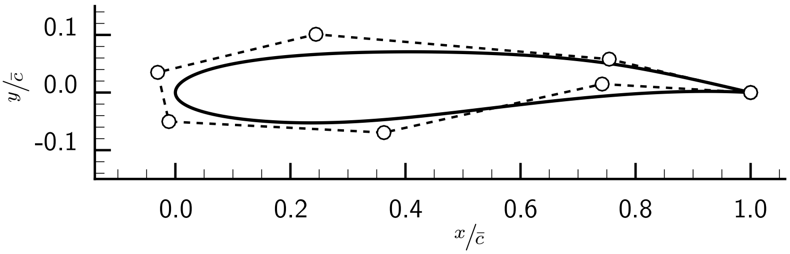


Figure 2.2: Aerofoil represented by a NURBS curve, with 8 control points, depicted by the open circles, and basis functions of degree 4. Control polygon, connecting consecutive control points, represented by the dashed line. Note, that at the trailing edge there are two control points that coincide. Dimensions are scaled by the chord length $\bar{c}$.

2.2.2 Rational basis spline curve

The concept of the basis spline can be extended by the introduction of a weighting factor w_k linked to each of the control points. To render the method coordinate invariant after the introduction of this weighting factor, a normalization factor needs to be introduced as well. Otherwise, the weighting factor effectively only scales the distance of the control point from the origin of the coordinate system, which means that the parametrization method is not coordinate invariant; this characteristic is undesirable. Including the normalization factor, the expression for the resulting so-called rational basis spline becomes

$$\boldsymbol{C}^\mathrm{p}(u) = \frac{1}{\sum\limits_{\mathrm{j}=0}^{\mathrm{n}-1}\left[N_\mathrm{j}^{(\mathrm{p}-1)}(u)\,w_\mathrm{j}\right]}\sum_{\mathrm{i}=0}^{\mathrm{n}-1}\left[w_\mathrm{i}\boldsymbol{P}_\mathrm{i}N_\mathrm{i}^{(\mathrm{p}-1)}(u)\right], \quad \upsilon_0 \le u \le \upsilon_{\mathrm{p}+\mathrm{n}-1},$$

with p again the order of the curve and n the number of control points used to specify the curve. In this equation the fraction in front of the summation sign represents the normalization factor. By rearranging this equation and the introduction of the rational basis function

$$R_\mathrm{i}^\mathrm{q}(u) := \frac{N_\mathrm{i}^\mathrm{q}(u)\,w_\mathrm{i}}{\sum\limits_{\mathrm{j}=0}^{\mathrm{n}-1}\left[N_\mathrm{j}^\mathrm{q}(u)\,w_\mathrm{j}\right]}, \tag{2.4}$$

the rational basis spline curve can be expressed as

$$\boldsymbol{C}^\mathrm{p}(u) := \sum_{\mathrm{i}=0}^{\mathrm{n}-1}\left[\boldsymbol{P}_\mathrm{i}R_\mathrm{i}^{(\mathrm{p}-1)}(u)\right], \quad \upsilon_0 \le u \le \upsilon_{\mathrm{p}+\mathrm{n}-1}. \tag{2.5}$$

2.2.3 NURBS surface

From the above description of the NURBS curve, an extension can be made to the 2D equivalent of the curve, i.e. the NURBS surface. This extension is achieved by the introduction of a second parametric variable v, an additional knot vector $\boldsymbol{\nu}$ and the use

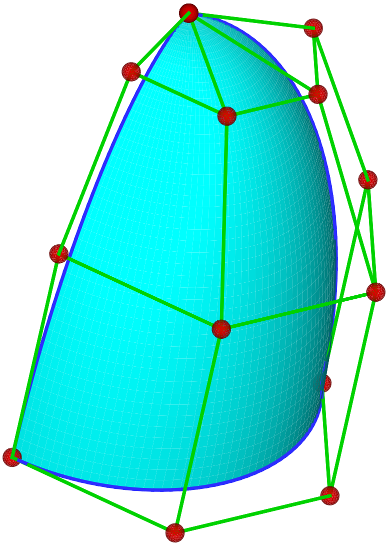


Figure 2.3: Part of an ellipsoid represented by a NURBS surface in cyan, together with the corresponding grid of control points represented by red spheres.

of a rectangular grid of control points, instead of a single array. If this grid consists of $\mathrm{n} \times \mathrm{m}$ control points and the order of the NURBS surface equals p in the direction that corresponds to variable u and q in the other direction, the shape of the surface is given by

$$\boldsymbol{S}^{\mathrm{pq}}\left(u,v\right) := \sum_{\mathrm{i}=0}^{\mathrm{n}-1}\left(\sum_{\mathrm{j}=0}^{\mathrm{m}-1}\left[\boldsymbol{P}_{\mathrm{ij}} R_{\mathrm{ij}}^{(\mathrm{p}-1)(\mathrm{q}-1)}\left(u,v\right)\right]\right), \quad \begin{cases} \upsilon_0 \leq u \leq \upsilon_{\mathrm{p}+\mathrm{n}-1}, \\ \nu_0 \leq v \leq \nu_{\mathrm{q}+\mathrm{m}-1}. \end{cases} \tag{2.6}$$

This expression is quite similar to the equation that describes a NURBS curve, equation (2.5). Now, the rational basis spline for a NURBS surface is:

$$R_{\mathrm{ij}}^{\mathrm{pq}}\left(u,v\right) := \frac{N_{\mathrm{i}}^{\mathrm{p}}\left(u\right) N_{\mathrm{j}}^{\mathrm{q}}\left(v\right) w_{\mathrm{ij}}}{\sum\limits_{\mathrm{k}=0}^{\mathrm{n}-1}\left(\sum\limits_{\mathrm{l}=0}^{\mathrm{m}-1}\left[N_{\mathrm{k}}^{\mathrm{p}}\left(u\right) N_{\mathrm{l}}^{\mathrm{q}}\left(v\right) w_{\mathrm{kl}}\right]\right)}. \tag{2.7}$$

The aforementioned features and characteristics of the NURBS curve also apply to the NURBS surface. Figure 2.3 shows part of the surface of an ellipsoid, represented by a NURBS surface, together with the corresponding grid of control points.

2.3 Aerofoil parametrization with NURBS curve

Before the parametrization of the blade of a wind turbine rotor is investigated with respect to the number of control points required for the representation of the design space, a less involved case is considered, namely the parametrization of an aerofoil using

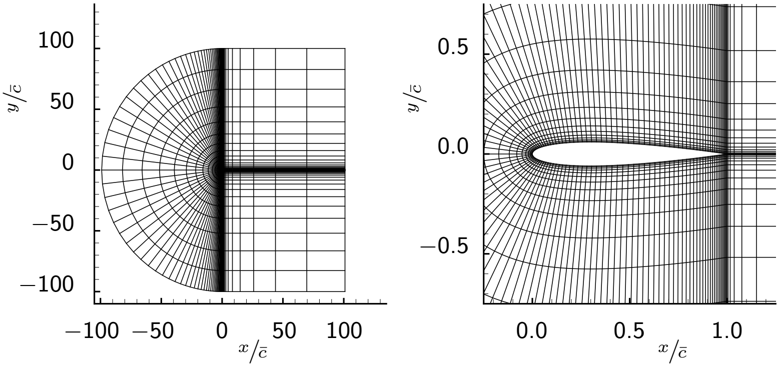
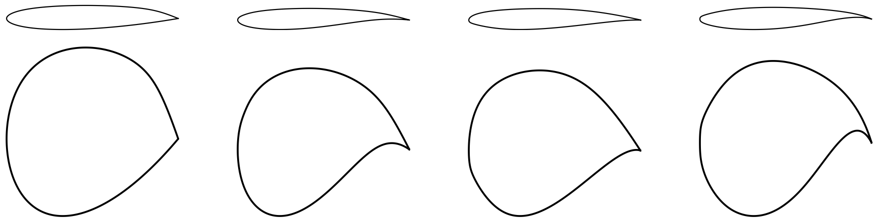


Figure 2.4: Flow domain used for solving 2D optimization problem, discretized using a C-grid consisting of 640×112 cells. For clarity, only every 4th grid line is displayed. Right picture shows detail of the grid around NACA 0012 aerofoil, used as initial geometry. Dimensions are scaled by chord length $\bar{c}$.

a NURBS curve. This test case is expected to give an indication of the number of design variables required for an aerodynamic optimization problem using non-uniform rational basis splines as a parametrization method. An investigation of the number of control points required for an accurate fit was performed by Lépine et al. [108, 109, 188]. In that research it was found that an arbitrary aerofoil can be represented sufficiently accurately, in terms of the accuracy required for a flow simulation, using 13 control points for the direct representation of the aerofoil shape. This result is adopted as a starting point for addressing the research question of how the number of control points required for an accurate fit of the geometry relates to the number of design variables necessary for an adequate representation of the design space for an aerodynamic optimization problem. This investigation is performed by solving an actual aerodynamic optimization problem, using a NURBS curve as the shape parametrization method. The optimization problem used for this purpose is the minimization of the wave drag of an aerofoil subject to the constraint of fixed lift, in a transonic inviscid flow. Even though the flow conditions considered in this test case are not particularly representative for the flow around a wind turbine blade, the transonic wave drag minimization problem is considered to sufficiently reflect the problem of the wind turbine blade optimization. The results will indicate the number of design variables required for an adequate representation of the design space in relation to the number of control points necessary for an accurate fit of the geometry.

2.3.1 Aerofoil optimization

This subsection gives a concise description of the optimization method employed for the 2D test case. For a more detailed treatise of the procedure, the reader is referred to the work of Hartkamp [79]. The compressible Euler equations are used to model the flow. The sensitivities needed for the optimization have been computed using the discrete adjoint equation method [90]. Furthermore, the optimization is performed using SNOPT [71], i.e. a library for performing gradient-based constrained optimization which employs a BFGS quasi-Newton method [143], developed at Stanford University. The flow equations have been discretized using a cell-centred finite volume scheme with an upwind



n = 6 n = 9 n = 12 n = 13

Figure 2.5: Results of the minimization of $\mathcal{I} \equiv c_\text{d}$ for fixed $c_\text{l} = 0.5$ in inviscid flow at a free-stream Mach number of $\mathrm{M}_\infty = 0.75$ using a variable number of control points for the parametrization of the aerofoil. Geometrical constraints imposed are: $\bar{\kappa}_\text{max}\bar{c} \leq 150$, $\bar{t}_\text{min}/\bar{c} \geq 0.12$ and $\tau_\text{te} \geq 10^\circ$. Results shown for — from left to right — a parametrization using 6, 9, 12 and 13 control points, respectively. For the version in the bottom row, the vertical scale is multiplied by a factor of $^1/_7$ to highlight the difference in shape.

convective flux discretization, using Roe's approximate Riemann solver [147]. Second-order spatial accuracy is achieved utilizing MUSCL-type reconstruction [193] via the Van Albada limiter [5]. The choice has been made to optimize an aerofoil in a transonic flow at a free-stream Mach number of $\mathrm{M}_\infty = 0.75$. The drag coefficient acts as the objective function in this optimization problem, i.e. $\mathcal{I} := c_\text{d}$ in equation (1.1), which defines the optimization problem. Besides some geometrical constraints, in order to obtain more or less realistic aerofoil shapes — for example in terms of the minimum trailing edge angle τ_te, the minimum thickness of the aerofoil $\bar{t}_\text{min}$ and the maximum allowable curvature $\bar{\kappa}_\text{max}$ — a constraint is imposed on the lift; the lift coefficient is set equal to 0.5. The NACA 0012 aerofoil is selected as the initial geometry to start the optimization procedure. For the specified condition and constraint[5] on the lift, the value of c_d equals $1.77 \cdot 10^{-2}$ for the NACA 0012 aerofoil. This result is obtained on a C-grid consisting of 640 cells along the aerofoil contour and wake line and 112 cells in the direction normal to the aerofoil. Moreover, the far-field is situated at 100 chord lengths away from the aerofoil. This domain — with its corresponding discretization, depicted in figure 2.4 — is also employed in the optimization procedure. To perform the actual optimization, a NURBS curve representation of this aerofoil is required. This representation has been obtained by performing a fit of the aerofoil shape, with the specified number of control points and degree of the splines, as described by Hartkamp [79]. Moreover, note that the curve fitting was performed using a fixed knot vector, with an initial uniform interior knot distribution. The choice was made to use a NURBS curve representation of the NACA 0012 aerofoil, modified to have a sharp trailing edge, with the number of control points ranging from 5 to 13. The final shapes resulting from solving this optimization problem for 6, 9, 12 and 13 control points are presented in figure 2.5. This picture clearly shows the diversity in shape for the different minima obtained. If on the other hand the value of the objective function of the final result is considered, listed in table 2.2, it can be observed that for the range considered, the minimum value of c_d is more or less independent of the number of control points used. This observation indicates that

---

[5] For the dimensions of the domain and the grid used, the required angle of attack — for the NACA 0012 aerofoil to satisfy the lift constraint, i.e. $c_\text{l} \equiv 0.5$ — equals 2.32855°.

Table 2.2: Results of the minimization of $\mathcal{I} \equiv c_{\mathrm{d}}$ for fixed $c_{\mathrm{l}} = 0.5$ in inviscid flow at a free-stream Mach number of $\mathrm{M}_{\infty} = 0.75$ — subject to geometrical constraints: $\bar{\kappa}_{\mathrm{max}}\bar{c} \leq 150$, $\bar{t}_{\mathrm{min}}/\bar{c} \geq 0.12$ and $\tau_{\mathrm{te}} \geq 10^{\circ}$ — using a variable number of control points for the parametrization of the aerofoil, listed in column one. The second column provides the number of design variables used in the optimization. The third column shows the accuracy of the geometric fit, by means of the quadratic mean of the error, of the NACA 0012 aerofoil for the corresponding NURBS curve representation, which was used as an initial guess for the optimization procedure. The fourth column lists the attained minimum resulting from solving the optimization problem. The three remaining columns provide the value of the geometrical constraint functions for the geometry obtained.

n	$\mathrm{n}_{\mathrm{design}}$	rms-error of fit	$\mathcal{I}_{\mathrm{min}}$	$\bar{t}_{\mathrm{min}}/\bar{c}$	τ_{te}	$\bar{\kappa}_{\mathrm{max}}\bar{c}$
5	10	$1.17 \cdot 10^{-4}$	$2.1321 \cdot 10^{-4}$	12.5	48.1°	66.7
6	13	$8.75 \cdot 10^{-5}$	$2.2141 \cdot 10^{-4}$	13.9	32.0°	66.4
7	16	$2.76 \cdot 10^{-5}$	$2.0840 \cdot 10^{-4}$	12.0	12.8°	82.6
8	19	$1.34 \cdot 10^{-5}$	$1.9616 \cdot 10^{-4}$	12.0	10.0°	67.0
9	22	$3.81 \cdot 10^{-6}$	$1.9381 \cdot 10^{-4}$	12.0	10.0°	66.9
10	25	$2.47 \cdot 10^{-6}$	$1.9362 \cdot 10^{-4}$	12.0	10.8°	68.6
11	28	$1.09 \cdot 10^{-6}$	$1.9020 \cdot 10^{-4}$	12.0	10.0°	66.2
12	31	$7.21 \cdot 10^{-7}$	$1.9529 \cdot 10^{-4}$	12.0	10.0°	68.3
13	34	$3.30 \cdot 10^{-7}$	$1.8388 \cdot 10^{-4}$	12.8	10.0°	76.7

from the aerodynamic shape optimization point of view, the number of control points required for an accurate fit — accurate in terms of the accuracy required for a flow simulation [188] — can be regarded as an upper limit for the number of control points needed for an adequate representation of the design space. However, it must be noted that the nature of this model optimization problem differs from the 3D blade optimization problem in the sense that the present model problem inherently exhibits multiple global minima due to the used inviscid flow model. Since the flow is subcritical, the exact value of c_{d} is equal to zero. For the optimized aerofoils the drag coefficient is not equal to zero due to the occurrence of numerical dissipation and the approximate evaluation of the integration of the surface pressure to compute the force coefficient. Moreover, the relative effect of the treatment of the far-field boundary condition or the location of the far-field boundary can be quite considerable [195].

From the results presented in table 2.2, it can be concluded that a NURBS curve representation of an aerofoil with five control points is sufficient to be able to obtain at least one of the many global minima present in the design space for this optimization problem. On the other hand, it is not known a priori if similar characteristics could be expected for the design space corresponding to the wind turbine blade optimization problem. Therefore, the safe approach is to choose the number of control points employed in the wind turbine blade optimization to be equal to the number of control points required for an accurate geometric fit. The parametrization of the wind turbine blade is discussed in the next section.

2.4 Parametrization of the wind turbine rotor blade

This section discusses the procedure employed to determine a NURBS surface representation of the geometry of the blade of a wind turbine rotor.

A discrete representation of the blade surface is required that has a fixed number of points representing each cross-section. The blade can, for example, be represented by $\mathrm{n}_1 \in \mathbb{N}_1$ points per cross-section for $\mathrm{n}_2 \in \mathbb{N}_1$ cross-sections. Once this representation has been obtained, a least-square fit of the discrete representation of the blade is performed.

From the 2D experiments of fitting aerofoils with a NURBS curve, it was found that specifying a knot vector with a uniform interior knot distribution, did not have a limiting effect on the achievable accuracy of the resulting parametrization. Based on this experience, the choice has been made to apply the same procedure for both knot vectors of the 3D NURBS surface used in the least-square fit procedure. The x, y and z-component of the coordinate as well as the weight of the control points are used as design variables in the fit procedure. To further improve the fit, the values of the parametric variables u and v for which the spline surface is evaluated are also used as design variables. Moreover, the actual fitting procedure is split in two parts. In the first step, the four boundaries of the reference surface are fitted as NURBS curves. Upon having obtained a NURBS curve representation of the edges of the blade surface, a fit of the remainder of the discrete blade surface is performed. The decision to split the fitting procedure into these two parts has been made to improve the accuracy of the fit at the edges of the blade surface. In this way the resulting fit will more accurately represent the trailing edge of the blade as well as its tip. Moreover, to account for the relative error being larger in the tip region, when carrying out a standard least square minimization where only the absolute error is considered, also a weighting of the error has been applied. This approach results in a more accurate representation of the blade in the tip region. Weighting has been implemented by dividing the error by the value Ξ_j of the perimeter of the local blade contour of the discrete version of the original blade. Applying this adjustment, the objective function for fitting the blade now reads

$$\mathcal{I}\left(\boldsymbol{x}\right) := \left(\text{weighted error}\right)^2_{\text{rms}} \equiv \frac{1}{\mathrm{n}_1\mathrm{n}_2}\sum_{\mathrm{j}=0}^{\mathrm{n}_2-1}\left[\frac{1}{\Xi_{\mathrm{j}}}\sum_{\mathrm{i}=0}^{\mathrm{n}_1-1}\left(\left|\left|\boldsymbol{x}\left(u_{\mathrm{i}}, v_{\mathrm{j}}\right) - \boldsymbol{x}'_{\mathrm{ij}}\right|\right|^2\right)\right], \qquad (2.8)$$

where $\boldsymbol{x}'_{\mathrm{ij}}$ represents a point of the original geometry and $\boldsymbol{x}\left(u_{\mathrm{i}}, v_{\mathrm{j}}\right)$ is the corresponding point of the fitted surface. Note, that the quadratic mean squared is used as objective function instead of just the quadratic mean, because it results in a smoother design space.

The fitting procedure has been used to determine the number of control points required to accurately represent the wind turbine blade surface. This investigation was done for two different wind turbine blade geometries. As discussed in the preceding section, an accurate representation of an arbitrary aerofoil using a NURBS curve for the direct parametrization of the shape requires about 13 control points. This finding is used as a starting point for the investigation of the number of control points required for the accurate representation of the shape of the wind turbine blade using a NURBS surface. Furthermore, some preliminary tests showed that 13 control points is also a suitable choice for the number of control points in the spanwise direction of the blade. The simple shape in that direction — simple relative to the shape of the cross-section —

Table 2.3: Results of fitting the geometry of the blade of a wind turbine rotor using 13 control points for the cross-section of the blade and also 13 control points for representing the shape in the spanwise direction. The geometrical fit is performed for two typical blades of different span, denoted by the name of the rotor for which they are used. The second column lists the span of the blade. The last two columns list the unweighed error of the fit by means of the quadratic mean and the ℓ_∞-norm, non-dimensionalized using the length of the blade and the wetted surface area.

Blade	Span $[\mathrm{m}]$	$(\text{error})_{\text{rms}}$	$\|\|\text{error}\|\|_\infty$
Suzlon S88	43.25	$8.21 \cdot 10^{-4}$	$6.39 \cdot 10^{-3}$
Suzlon S64	29.50	$5.05 \cdot 10^{-4}$	$4.53 \cdot 10^{-3}$

might give the impression that fewer control points would also be sufficient. However, it was found that using fewer control points in the spanwise direction results in an inaccurate representation of the blade in the region near the root, especially in the region where the shape of the cross-section changes significantly. The degree of the basis functions for the NURBS surface was chosen to equal four in both directions. This choice was made to have a sufficiently differentiable surface, which, for example, is required to be able to determine the local curvature. Besides that, the use of basis functions of a higher degree results in a wider range of influence on the shape of the surface of a control point. Therefore, to retain the local controllability — considering the third property of basis functions, listed on page 22 — the degree of the basis functions has been limited to four. The results of the fitting procedure for two different blades are summarized in table 2.3. In the errors listed in this table, the weighting by the perimeter of local blade contour is not taken into account. These results show that both geometries can be represented with a similar accuracy using a grid of 13 × 13 control points. The accuracy of the fit can be improved by increasing the number of control points in one or both directions. Furthermore, it is possible that a more elaborate fitting procedure, for example the method suggested by Becker [15], would provide a more accurate fit for the same number of control points. However, the accuracy achieved with the current approach and for the given number of control points is considered adequate, based on the aim of this investigation and the aforementioned results and remarks presented in the preceding section. Therefore, a NURBS surface representation by a grid of 13 × 13 control points should be used for the parametrization of the wind turbine blade in the optimization procedure.

When in the optimization all three components of the coordinate, as well as the corresponding weight are used as design variables, it results in a total number of $13 \times 13 \times 4 = 676$ design variables. However, in practice the number of design variables used for solving an optimization problem will be lower than this number, for several reasons: (i) the blade must be closed at the trailing edge, which reduces the number of design variables by $4 \cdot \mathrm{n}_2$; (ii) the weights may not be used — solving 2D problems indicated that similar results can be obtained without using weights — which reduces the number of design variables by $\mathrm{n}_1 \cdot \mathrm{n}_2$; (iii) the coordinates of not all control points need to be fully independent; (iv) the coordinate of some control points must be fixed, to control the absolute position of the blade.

2.5 Summary

Various parametrization methods have been investigated by reviewing publications on parametrization methods that have been employed for solving aerodynamic shape optimization problems. Based on this investigation, the choice has been made to use a NURBS surface for the direct parametrization of the geometry of the surface of the wind turbine blade. Subsequently, the mathematical description of the basis spline, the rational basis spline and the non-uniform rational basis spline has been presented, followed by the equations to describe a NURBS surface. Next, the results were presented of an investigation performed to determine the number of design variables required for the accurate representation of the design space for a typical aerodynamic shape optimization problem. This investigation was carried out using a model aerodynamic shape optimization problem, viz. the minimization of the wave drag for an aerofoil in transonic inviscid flow subject to the constraint of fixed lift. This research showed that the number of design variables required for an accurate representation of the geometry of an aerofoil can serve as an upper limit for the number of design variables required to represent the design space for solving an aerodynamic shape optimization problem. Finally, the number of control points required to represent a wind turbine blade using a NURBS surface was investigated. It has been found that a grid of 13×13 control points is sufficient for this task for a NURBS surface with basis functions of degree four.

3

Flow domain discretization

> *"I saw before me a vast multitude of small Straight Lines [...] interspersed with other Beings still smaller and of the nature of lustrous points [...]."*
>
> — Edwin A. Abbott (1838 – 1926), *Flatland*

The nature of the equations that describe the flow of a fluid is such that they cannot be solved analytically for a non-trivial flow configuration. If, however, a solution of these equations is required, one has to accept an approximate solution of the equations, which can be obtained by solving the flow equations numerically [29, 200]. For this purpose, a discrete representation of the geometric configuration is required, for which the flow equations need to be solved. This chapter treats the methods employed in the present research for the discretization of the geometry and the corresponding flow domain around the configuration. First, the requirements for the grid generation method are considered, focused on solving an aerodynamic wind turbine rotor blade shape optimization problem. Subsequently, a choice is made regarding the discretization method to be used in the present research. Related to the choice to use a hyperbolic grid generation method, composite overset grids are used to facilitate the block connectivity. Thereafter, the surface grid generation is treated. Two methods are discussed for the discretization of curves bounding the surface. Then, two different methods for the discretization of the parametric surface are treated. The resulting surface grid is used as an initial condition for the generation of the volumetric hyperbolic field grid; this method is presented in section 3.3. The chapter ends with a summary and an example of field grid that is generated using the methods presented in this chapter.

3.1 Introduction

The generation of a discretized flow domain consists of three steps. The first step is the definition of the flow domain; this definition refers to the specification of the surfaces bounding the flow domain. Subsequently, these surfaces need to be discretized, to serve as a basis for the volume grid. The generation of this volume grid is the third and final step in the discretization of the flow domain. For all three steps different methods exist. The most prominent advantages and disadvantages are discussed of the methods considered.

3.1.1 Requirements

In order to choose a suitable method for the discretization of the flow domain for solving a wind turbine optimization problem, first the requirements are discussed. The prerequisite follows directly from the choice for the flow solver that is employed to solve the flow equations. As discussed in chapter 1, a multi-block structured grid flow solver is used. Therefore, the flow domain discretization method must provide a grid of this type. The second requirement concerns the efficiency of the generation of the grid. Since a new grid is required for each design iteration, the generation of the grid must not be very costly in terms of the CPU-time required, relative to the CPU-time requirements for obtaining a flow solution. This demand also brings up the next requirement, that is, the generation of the grid needs to be fully automatic. The need for intervention of the user to generate a new grid for each iteration, will render the optimization method impractical. Moreover, the grid generation procedure must be robust, i.e. the grid generation must be able to generate a grid for each feasible blade, from an aerodynamic perspective. Otherwise, the optimization procedure can abort prematurely and user intervention is required to continue the optimization. Another requirement concerns the grid quality. As discussed in subsection 1.3, the Euler equations are used to model the flow. The hyperbolic nature of the Euler equations for unsteady flow has certain implications with respect to the discretization of the flow domain. Because a hyperbolic operator in general does not provide any smoothing effect [81], a poor quality of a grid is directly reflected in the discrete solution of the Euler equations. Furthermore, physical effects, such as refraction or reflection of waves, can occur due to a non-physical cause, like non-smooth features of the discretized flow domain. Moreover, a poor grid quality negatively affects convergence and stability of the flow solution method. Based on these requirements, a choice for a suitable discretization method is made in the next section.

3.1.2 Considerations

Different methods exist for the generation of a structured grid that can be used for solving the equations that govern the flow. Two of these methods can satisfy the requirement of providing a good grid quality[6]. The first method is called elliptic grid generation [174]. In this case *elliptic* refers to the type of partial differential equation that is solved to determine the location of the vertices of the grid. Due to the nature of the underlying partial differential equations, the surface grid of all six faces bounding the part of the volume that needs to be discretized must be specified. The second method that provides a volume grid of good quality is also named after the class of partial differential equations that must be solved in this method, i.e. hyperbolic equations [177]. This type of equations has also certain implications for the particular grid generation procedure. A hyperbolic grid generation method requires the specification of the surface grid of only one of the faces bounding the volume for which a discrete representation is required. This surface provides the initial condition for the hyperbolic partial differential equation. The surface grids of the remainder of the faces bounding the discretized volume follow from the solution of the hyperbolic equations and the boundary conditions enforced; they are effectively the result from marching the specified surface grid into the domain. A

---

[6] Note, that for specific cases it is also possible to realize a good grid quality, using algebraic grid generation methods. However, the grid generation method should be universal. Therefore, an algebraic method is less suitable, because generally it provides grids of lower quality.

consequence of this approach is, that the remainder of the boundaries cannot be specified. An advantage of the hyperbolic grid procedure is that this method is considerably faster than an elliptic method [20, 81]. Therefore, the choice is made to use a hyperbolic grid generation method for the generation of the field grid in the optimization procedure, since the computational efficiency of the method is a strict requirement.

To alleviate the problem of not being able to specify all boundaries of the volume grid explicitly, the choice is made to use a composite overset grid approach, see section 1.4.1 on page 5. With this approach, the block connectivity can be achieved when there is sufficient overlap between the different blocks. With a hyperbolic grid generation method, this requirement is easily met. The method to determine the block connectivity for the composite overset grids method is discussed in more detail in chapter 4. However, this choice has also certain implications for the flow solution method itself, i.e. it needs to be able to handle composite overset grids. Another advantage of the use of overset grids is that it simplifies the domain decomposition in different blocks; this task can be quite cumbersome when point matching of the blocks is used as the sole method for the block connectivity.

The generation of a volume grid requires a surface grid as a boundary or initial condition. Because the quality of the surface grid is reflected in the volume grid, especially when a hyperbolic volume grid generation method is used, the surface grid needs to be of good quality. Therefore, the choice is made to use two different methods for the generation of the surface grid. The first method is surface grid generation by means of linear transfinite interpolation and the second method is elliptical surface grid generation. The first method is the preferred choice because it is more efficient in generating a surface grid. However, situations exist for which linear transfinite interpolation cannot provide adequate grid quality. For these cases, the computationally more intensive elliptical surface grid generation method is used. Both methods are presented in the next section.

3.2 Surface grid generation

The generation of a surface grid is analogous to the generation of a volume grid. Therefore, the grid generation procedure requires a discrete representation of the entity bounding the domain to be discretized. In case of a surface, the bounding entity consists of curves. Two methods for the discretization of the curves bounding the surface are treated in subsection 3.2.1. Subsequently, the two different methods for the generation of the actual surface grid are discussed, viz. linear transfinite interpolation in subsection 3.2.2 and elliptical grid generation in subsection 3.2.3.

3.2.1 Vertex distribution on boundary curves

Consider a surface $\mathcal{S} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ bound by four parametric curves $\mathcal{C}_{\mathrm{k}}\left(t\right) : \mathbb{R} \rightarrow \mathbb{R}^3$, $\mathrm{k} \in \{0, 1, 2, 3\}$. The distribution of the vertices on these boundary curves serves as a boundary condition for the surface grid. Therefore, it is important to note that requirements for the discretization of the domain must already be taken into account in the distribution of the vertices on the boundary curves. Smoothness of the variation in grid spacing is the first requirement. Another requirement concerns clustering of vertices in regions of rapid spatial variation of the flow variables. This rapid variation can be due to geometrical features like highly curved surfaces, or it can find its origin in the nature of the flow. The method discussed in the next subsection takes the curvature of the bounding curve into account in determining the vertex distribution. In situations that other factors than the geometrical feature curvature are decisive for the required vertex distribution, the algebraic stretching function by Vinokur [198] is used, which is also presented.

Curvature based vertex distribution

This method considers the smoothness of the spacing of the vertices and the vertex density in regions of high curvature for determining the vertex distribution. A smooth spacing means that the distance from one vertex to its neighbouring vertices in both directions must not differ too much. For vertex i on curve $\mathcal{C}_\mathrm{k}\left(t\left[\xi\right]\right)$, $\xi \in \mathbb{R}$, with coordinate $\boldsymbol{r}_\mathrm{i} \equiv \mathcal{C}_\mathrm{k}\left[t\left(\mathrm{i}\right)\right] \equiv \left(x_\mathrm{i}, y_\mathrm{i}, z_\mathrm{i}\right)^T$ this requirement can be quantified as

$$\frac{\left|\boldsymbol{r}_{\mathrm{i}+1} - \boldsymbol{r}_\mathrm{i}\right|}{\left|\boldsymbol{r}_\mathrm{i} - \boldsymbol{r}_{\mathrm{i}-1}\right|} < \bar{\alpha}, \quad \bar{\alpha} \in \mathbb{R}, \quad \mathrm{i} \in \mathbb{N}_1, \tag{3.1}$$

where $\bar{\alpha}$ is the factor specifying the maximum allowable relative distance between consecutive vertices. For a good quality grid, the upper bound yields a value of $\bar{\alpha} = 1.3$ [36].

The arc length $s_\xi \equiv s\left(t\left[\xi\right]\right)$ along curve $\mathcal{C}_\mathrm{k}\left(t\left[\xi\right]\right)$ from the beginning of the curve at $t\left(0\right)$ to vertex i can be determined by the evaluation of the integral

$$s_\mathrm{i} = \int\limits_{t(0)}^{t(\mathrm{i})} \left|\frac{\mathrm{d}\mathcal{C}_\mathrm{k}\left(\tau\right)}{\mathrm{d}\tau}\right| \mathrm{d}\tau. \tag{3.2}$$

Furthermore, the local curvature $\kappa_\xi \equiv \kappa\left(t\left[\xi\right]\right)$ for a parametric curve in $\mathbb{R}^3$ can be computed from

$$\kappa_\mathrm{i} = \left.\frac{\left|\frac{\mathrm{d}\mathcal{C}_\mathrm{k}(t)}{\mathrm{d}t} \times \frac{\mathrm{d}^2\mathcal{C}_\mathrm{k}(t)}{\mathrm{d}t^2}\right|}{\left|\frac{\mathrm{d}\mathcal{C}_\mathrm{k}(t)}{\mathrm{d}t}\right|^3}\right|_{t(\mathrm{i})}. \tag{3.3}$$

A vertex distribution along a curve to be discretized can be obtained by solving

$$\left.\frac{\mathrm{d}^2 s\left(t\left[\xi\right]\right)}{\mathrm{d}\xi^2}\right|_\mathrm{i} = f\left(s, \kappa\right), \tag{3.4}$$

subject to boundary conditions $s_0 = 0$ and $s_{\mathrm{n}-1}$ equal to the length of the curve, for $t \ \forall\ \mathrm{i} \in \mathbb{N}_1 : \mathrm{i} < \left(\mathrm{n}-1\right)$, where $\mathrm{n} \in \mathbb{N}_1$ is the number of vertices used for the discretization of the curve. Function $f\left(s, \kappa\right)$ can be used to take into account specific features of the curve for determining the vertex distribution. Solving equation (3.4) for $f\left(s, \kappa\right) = 0$ results in an equidistant vertex distribution along the curve. To take into account the effect of curvature, the following expression for $f\left(s, \kappa\right)$ is formulated

$$f\left(s, \kappa\right) = \begin{cases} \left(\frac{1-\bar{\alpha}}{\bar{\alpha}}\right)\left(s_{\mathrm{i}+1} - s_\mathrm{i}\right)\tanh\left(\lambda\left[\theta - 1\right]\right) & \text{if } \Upsilon \geq 1, \\ -\left(\frac{1-\bar{\alpha}}{\bar{\alpha}}\right)\left(s_{\mathrm{i}+1} - s_\mathrm{i}\right)\tanh\left(\lambda\left[\theta - 1\right]\right) & \text{if } \Upsilon < 1, \end{cases} \tag{3.5}$$

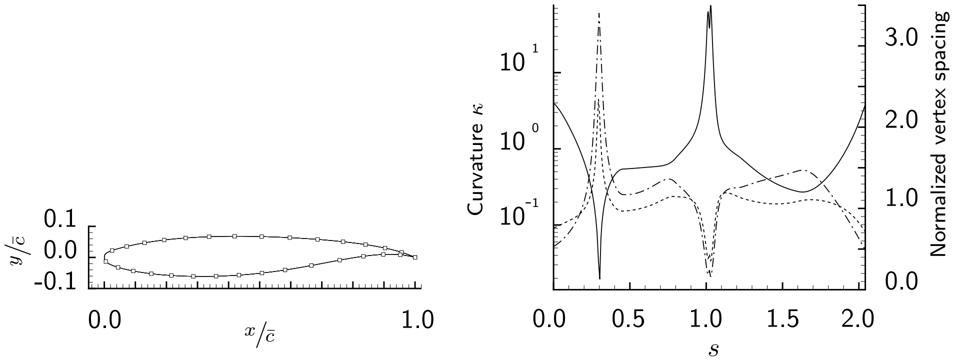


Figure 3.1: The aerofoil in the left picture has 31 vertices distributed along its circumference. Parameters used to obtain this distribution are $\bar{\alpha} = 1.2$ and $\lambda = 1.0$; dimensions are scaled by chord length $\bar{c}$. The right picture shows the characteristics of the vertex distribution of 513 vertices around the aerofoil. The arc length, along the x-axis, starts at the trailing edge and runs from the lower side to the upper side. The solid line depicts the curvature. The dashed line represents the vertex spacing obtained for $\lambda = 1.0$, the line with the dash-dot pattern is obtained for $\lambda = 2.0$. Both results have been normalized by the average vertex spacing and use parameter $\bar{\alpha} = 1.2$.

where $\lambda \in \mathbb{R}$ is a parameter which can be used to specify the relative importance of the curvature and $\theta \in \mathbb{R}$ is the ratio of the curvature-weighted arc length of two subsequent elements of the curve, specified as

$$\theta = \begin{cases} \Upsilon & \text{if } \Upsilon \geq 1, \\ \Upsilon^{-1} & \text{if } \Upsilon < 1, \end{cases} \qquad \text{with } \Upsilon := \frac{\kappa_{\mathrm{i}} \left(s_{\mathrm{i}+1} - s_{\mathrm{i}} \right)}{\kappa_{\mathrm{i}-1} \left(s_{\mathrm{i}} - s_{\mathrm{i}-1} \right)}. \tag{3.6}$$

The distribution of vertices along the curve is obtained by solving equation (3.4) iteratively for each discrete value of ξ. This iterative procedure is stopped when the average change in parametric value, for all vertices, between two subsequent iterations is below $5 \cdot 10^{-5}$. This threshold is typically reached in less than a thousand iterations. The left picture of figure 3.1 shows the result of the vertex distribution, on a curve representing an aerofoil, that was obtained with the method outlined above for $\mathrm{n} = 31$. Furthermore, figure 3.1 shows the characteristics of the curve by means of its curvature plotted against the arc length along the aerofoil, starting at the trailing edge on the lower side, represented by the solid line. The dashed line in this graph represents the vertex spacing obtained for $\lambda = 1.0$ and $\bar{\alpha} = 1.2$ for a vertex distribution of 513 vertices along the curve. The result is normalized by the average spacing. This figure shows how the method decreases the vertex spacing in regions of high curvature. To show the influence of parameter λ, the procedure is repeated for $\lambda = 2.0$. This result is represented by the line with the dash-dot pattern. It clearly shows the tendency to decrease the vertex spacing even more than for $\lambda = 1.0$ in regions of high curvature and vice versa for regions of low curvature.

Vinokur's stretching function

In situations for which the curve, for which the vertex distribution needs to be determined, shows no large variations in curvature or when other factors are more important for the grid spacing, the algebraic vertex distribution function provided by Vinokur [198] is used. For the particular function employed, the required vertex spacing at both ends of the curve must be specified, i.e. $(s_1 - s_0)$ and $(s_{\mathrm{n}-1} - s_{\mathrm{n}-2})$. For this purpose, define the following parameters

$$A := \frac{\sqrt{s_{\mathrm{n}-1} - s_{\mathrm{n}-2}}}{\sqrt{s_1 - s_0}}, \qquad B := \frac{1}{(\mathrm{n}-1)\sqrt{(s_1 - s_0)(s_{\mathrm{n}-1} - s_{\mathrm{n}-2})}}.$$

Note, that s_0 usually equals zero. Depending on the value of B, the stretching factor δ is computed by solving

$$\begin{aligned} \frac{\sin(\delta)}{\delta} &= B, \qquad \text{for } B < 1 \\ \frac{\sinh(\delta)}{\delta} &= B, \qquad \text{for } B > 1 \end{aligned}$$

for δ. Stretching factor δ is subsequently used to determine $t(\xi)$

$$t(\xi) = \frac{1}{2}\left(1 + \frac{\tanh\left(\left[\frac{\xi}{\mathrm{n}-1} - \frac{1}{2}\right]\delta\right)}{\tanh\left(\frac{\delta}{2}\right)}\right). \tag{3.7}$$

Note, that for the limiting case of $B \equiv 1$, intermediate variable t is uniformly distributed. This intermediate variable can then be used to determine arc length $s(t[\xi])$ according to

$$s(t[\xi]) = \frac{t(\xi)}{A + [1 - A]\, t(\xi)}, \tag{3.8}$$

which completes the mathematical formulation for this particular algebraic stretching function.

3.2.2 Transfinite interpolation

Again, consider a parametric surface $\mathcal{S}(\boldsymbol{u}) : \mathbb{R}^2 \to \mathbb{R}^3$ bound by four parametric curves $\mathcal{C}_{\mathrm{k}}(t) : \mathbb{R} \to \mathbb{R}^3$, $\mathrm{k} \in \{0, 1, 2, 3\}$. The vertex distribution on each of these parametric curves is used as a starting point for the discretization of surface $\mathcal{S}$. The parametric curves that are topologically opposite to each other need the same number of vertices. Therefore, the discretization of the surface by $\mathrm{n}_1 \times \mathrm{n}_2$ vertices, requires curves $\mathcal{C}_0$ and $\mathcal{C}_1$ to be discretized by $\mathrm{n}_1 \in \mathbb{N}_1$ vertices and curves $\mathcal{C}_2$ and $\mathcal{C}_3$ by $\mathrm{n}_2 \in \mathbb{N}_1$ vertices. Then, the boundary vertex distributions in the parametric space of the surface parametrization are obtained by solving

$$\begin{aligned} \mathcal{S}(\boldsymbol{u}) &= \mathcal{C}_0(t[\mathrm{i}]) \quad \text{for} \quad \boldsymbol{u}_{\mathrm{ij}}\ \forall\, \mathrm{i} \in \mathbb{N}_0 \ : \ \mathrm{i} < \mathrm{n}_1 \,,\ \mathrm{j} \equiv 0 \\ \mathcal{S}(\boldsymbol{u}) &= \mathcal{C}_1(t[\mathrm{i}]) \quad \text{for} \quad \boldsymbol{u}_{\mathrm{ij}}\ \forall\, \mathrm{i} \in \mathbb{N}_0 \ : \ \mathrm{i} < \mathrm{n}_1 \,,\ \mathrm{j} \equiv \mathrm{n}_2 - 1 \\ \mathcal{S}(\boldsymbol{u}) &= \mathcal{C}_2(t[\mathrm{j}]) \quad \text{for} \quad \boldsymbol{u}_{\mathrm{ij}}\ \forall\, \mathrm{j} \in \mathbb{N}_0 \ : \ \mathrm{j} < \mathrm{n}_2 \,,\ \mathrm{i} \equiv 0 \\ \mathcal{S}(\boldsymbol{u}) &= \mathcal{C}_3(t[\mathrm{j}]) \quad \text{for} \quad \boldsymbol{u}_{\mathrm{ij}}\ \forall\, \mathrm{j} \in \mathbb{N}_0 \ : \ \mathrm{j} < \mathrm{n}_2 \,,\ \mathrm{i} \equiv \mathrm{n}_1 - 1, \end{aligned} \tag{3.9}$$

where $\boldsymbol{u}_{\mathrm{ij}} \equiv (u_{\mathrm{ij}}, v_{\mathrm{ij}})^T$. Next, define the set $\mathbb{U}_{\mathsf{b}}$ to contain the value of the parametric variable vector $\boldsymbol{u}$ for all boundary vertices, i.e. the solution of equation (3.9):

$$\begin{aligned}\mathbb{U}_{\mathsf{b}} := &\{\boldsymbol{u}_{\mathrm{ij}} : (\mathrm{i} \in \mathbb{N}_0 : \mathrm{i} < \mathrm{n}_1), \mathrm{j} \equiv 0\} \cup \{\boldsymbol{u}_{\mathrm{ij}} : (\mathrm{i} \in \mathbb{N}_0 : \mathrm{i} < \mathrm{n}_1), \mathrm{j} \equiv \mathrm{n}_2 - 1\} \cup \\ &\{\boldsymbol{u}_{\mathrm{ij}} : (\mathrm{j} \in \mathbb{N}_0 : \mathrm{j} < \mathrm{n}_2), \mathrm{i} \equiv 0\} \cup \{\boldsymbol{u}_{\mathrm{ij}} : (\mathrm{j} \in \mathbb{N}_0 : \mathrm{j} < \mathrm{n}_2), \mathrm{i} \equiv \mathrm{n}_1 - 1\}.\end{aligned} \tag{3.10}$$

Subsequently, interpolation is performed to obtain the coordinate in parameter space of the inner vertices of the surface grid, i.e. finding the set

$$\mathbb{U}_{\mathsf{i}} := \left\{ u(\xi,\eta),\ v(\xi,\eta)\ :\ \begin{array}{l} \xi \in \{\mathbb{R} \cap \mathbb{N}_1\ :\ \xi < \mathrm{n}_1 - 1\}, \\ \eta \in \{\mathbb{R} \cap \mathbb{N}_1\ :\ \eta < \mathrm{n}_2 - 1\} \end{array} \right\}. \tag{3.11}$$

Then, the parametric coordinate is used to determine the actual coordinate of the vertices in physical space. For clarification of the concept parameter space and computational space, see figure 3.2.

The interpolation procedure used to find $\mathbb{U}_{\mathsf{i}}$ is outlined next, following the method described by Khamayseh and Kuprat [96]. The method starts with performing a scaling of the computational coordinate, defined as

$$\sigma_\xi := \frac{\xi}{\mathrm{n}_1 - 1} \qquad \text{and} \qquad \tau_\eta := \frac{\eta}{\mathrm{n}_2 - 1}, \tag{3.12}$$

such that σ_ξ and τ_η both range from 0 to 1. Then the inner vertices of the surface grid are obtained by evaluating

$$\boldsymbol{u}_{\mathrm{i,j}} = \boldsymbol{u}^{\sigma}_{\mathrm{i,j}} + \boldsymbol{u}^{\tau}_{\mathrm{i,j}} - \boldsymbol{u}^{\sigma\tau}_{\mathrm{i,j}}, \quad 0 < \mathrm{i} < \mathrm{n}_1 - 1, \quad 0 < \mathrm{j} < \mathrm{n}_2 - 1, \tag{3.13}$$

where $\boldsymbol{u}^{\sigma}_{\mathrm{i,j}}$ stands for an interpolation in σ-direction between the lower and upper boundary in ξ-direction. Equation (3.14) provides an expression for σ, if linear interpolation is used for this purpose. The interpolation in τ-direction is represented by $\boldsymbol{u}^{\tau}_{\mathrm{i,j}}$, which in turn is an interpolation between the upper and lower boundary in η-direction, for linear interpolation expressed by equation (3.15). Moreover, $\boldsymbol{u}^{\sigma\tau}_{\mathrm{i,j}}$ stands for the combined interpolation in both directions. First, an interpolation in τ-direction is performed between the four corners of the grid, which gives an interpolated upper and lower boundary in η-direction. Subsequently, an interpolation is performed in the σ-direction between the interpolated upper and lower boundary obtained in the previous step. When linear interpolation is also used for this purpose, the procedure can be expressed by equation (3.16). Besides linear interpolation, also higher-order interpolation methods exist. However, in the present research, only linear interpolation is applied, for which the corresponding equations are given below:

$$\boldsymbol{u}^{\sigma}_{\mathrm{i,j}} = ([1 - \sigma_{\mathrm{i}}], \sigma_{\mathrm{i}}) \cdot \begin{pmatrix} \boldsymbol{u}_{0,\mathrm{j}} \\ \boldsymbol{u}_{\mathrm{n}_1-1,\mathrm{j}} \end{pmatrix}, \tag{3.14}$$

$$\boldsymbol{u}^{\tau}_{\mathrm{i,j}} = (\boldsymbol{u}_{\mathrm{i},0}, \boldsymbol{u}_{\mathrm{i},\mathrm{n}_2-1}) \cdot \begin{pmatrix} [1 - \tau_{\mathrm{j}}] \\ \tau_{\mathrm{j}} \end{pmatrix}, \tag{3.15}$$

$$\boldsymbol{u}^{\sigma\tau}_{\mathrm{i,j}} = ([1 - \sigma_{\mathrm{i}}], \sigma_{\mathrm{i}}) \cdot \begin{bmatrix} \boldsymbol{u}_{0,0} & \boldsymbol{u}_{0,\mathrm{n}_2-1} \\ \boldsymbol{u}_{\mathrm{n}_1-1,0} & \boldsymbol{u}_{\mathrm{n}_1-1,\mathrm{n}_2-1} \end{bmatrix} \begin{pmatrix} 1 - \tau_{\mathrm{j}} \\ \tau_{\mathrm{j}} \end{pmatrix}. \tag{3.16}$$

Once the interpolation is performed, the set $\mathbb{U}_{\mathsf{i}}$ is obtained. This result is combined with $\mathbb{U}_{\mathsf{b}}$ to arrive at the representation of the surface grid in parametric space — an

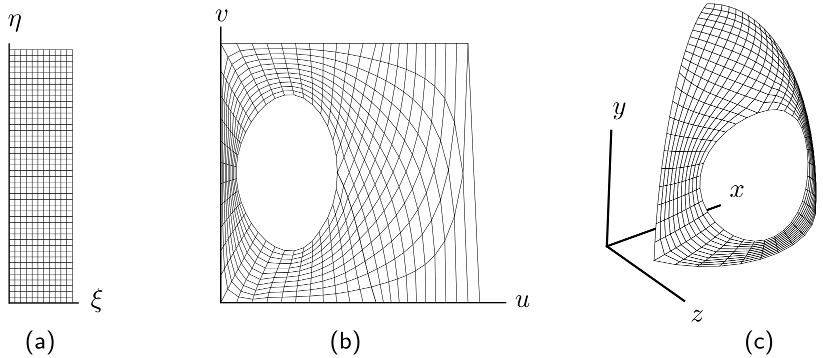


Figure 3.2: Mapping from: (a) computational space, with coordinate $(\xi,\eta)^T$, to (b) parameter space, with coordinate $(u,v)^T$, which in turn is mapped to (c) physical space, with coordinate $(x,y,z)^T$.

example of such a representation can be seen in the centre picture of figure 3.2. If the parametric surface $\mathcal{S}(\boldsymbol{u})$ is represented by a NURBS surface $\boldsymbol{S}^{\mathrm{pq}}(u,v)$, the surface grid in physical space is subsequently obtained by evaluating $\boldsymbol{S}^{\mathrm{pq}}(u,v)$ for all distinct $\boldsymbol{u}$ in parameter space, i.e.

$$\mathbb{X}_{\mathsf{s}} = \left\{\boldsymbol{S}^{\mathrm{pq}}(\boldsymbol{u}) \ \forall\, \boldsymbol{u} \in \{\mathbb{U}_{\mathsf{b}} \cup \mathbb{U}_{\mathsf{i}}\}\right\}. \tag{3.17}$$

3.2.3 Elliptical surface grid generation

For non-complex shapes, surface grid generation by means of the above linear transfinite interpolation provides good grid quality. However, situations exist for which this quality is not satisfactory, such as for the surface grid on the spinner of the wind turbine rotor. In these cases, a different surface grid generation method must be employed. One such method is the so-called elliptical surface grid generation. As the name indicates, an elliptical partial differential equation is solved for the location of each vertex of the surface grid. This grid generation technique starts with a surface grid that acts as an initial guess for the elliptical surface grid generation method. This initial grid can be determined using a different grid generation method, such as the method described in the preceding section.

In the elliptical surface grid generation procedure, just as with the transfinite interpolation, the parametric value corresponding to each vertex value is first determined. Subsequently, the parametrization used for the surface is used to determine the corresponding physical location of each of the vertices, resulting in the actual surface grid. For a parametric surface $\mathcal{S}(\boldsymbol{u})$ the elliptical surface grid generation equations can be written as [96]:

$$\left[\frac{\partial\mathcal{S}}{\partial\eta}\right]^2\left(\frac{\partial^2\boldsymbol{u}}{\partial\xi^2}+P\frac{\partial\boldsymbol{u}}{\partial\xi}\right)-2\left[\frac{\partial\mathcal{S}}{\partial\xi}\frac{\partial\mathcal{S}}{\partial\eta}\right]\frac{\partial^2\boldsymbol{u}}{\partial\xi\partial\eta}+ \\ \left[\frac{\partial\mathcal{S}}{\partial\xi}\right]^2\left(\frac{\partial^2\boldsymbol{u}}{\partial\eta^2}+Q\frac{\partial\boldsymbol{u}}{\partial\eta}\right)=J^2\Delta_2\boldsymbol{u}, \tag{3.18}$$

where P and Q are the so-called control functions, J is the determinant of the Jacobian matrix of the transformation from computational to parametric space, i.e. $J = \frac{\partial u}{\partial \xi}\frac{\partial v}{\partial \eta} - \frac{\partial u}{\partial \eta}\frac{\partial v}{\partial \xi}$, and $\Delta_2 \boldsymbol{u}$ is defined as

$$\Delta_2 \boldsymbol{u} := \overline{J} \begin{pmatrix} \frac{\partial}{\partial u}\left[\frac{\left(\frac{\partial \mathcal{S}}{\partial v}\right)^2}{\overline{J}}\right] - \frac{\partial}{\partial v}\left[\frac{\frac{\partial \mathcal{S}}{\partial u}\frac{\partial \mathcal{S}}{\partial v}}{\overline{J}}\right] \\ \frac{\partial}{\partial v}\left[\frac{\left(\frac{\partial \mathcal{S}}{\partial u}\right)^2}{\overline{J}}\right] - \frac{\partial}{\partial u}\left[\frac{\frac{\partial \mathcal{S}}{\partial u}\frac{\partial \mathcal{S}}{\partial v}}{\overline{J}}\right] \end{pmatrix}.$$

In this equation $\overline{J}$ is given by

$$\overline{J} := \sqrt{\left(\frac{\partial \mathcal{S}}{\partial u}\right)^2 \left(\frac{\partial \mathcal{S}}{\partial v}\right)^2 - \left(\frac{\partial \mathcal{S}}{\partial u}\frac{\partial \mathcal{S}}{\partial v}\right)^2}.$$

Control functions P and Q are used to control the characteristics of the surface grid, such as the spacing of the grid lines and their orthogonality. When the equations are solved for both P and Q equal to zero, a grid with uniform spacing of the grid lines is obtained [96]. However, when some of the characteristics of the initial surface grid need to be retained, P and Q can be specified in such a way that this aim is achieved. The control functions used in the present research realize an improved orthogonality of the grid lines, while maintaining the grid spacing of the initial grid in the direction normal to the boundaries. For this purpose, the value of the control functions is obtained by solving the following system of linear equations:

$$\begin{bmatrix} \left(\frac{\partial \mathcal{S}}{\partial \eta}\right)^2 \frac{\partial u}{\partial \xi} & \left(\frac{\partial \mathcal{S}}{\partial \xi}\right)^2 \frac{\partial u}{\partial \eta} \\ \left(\frac{\partial \mathcal{S}}{\partial \eta}\right)^2 \frac{\partial v}{\partial \xi} & \left(\frac{\partial \mathcal{S}}{\partial \xi}\right)^2 \frac{\partial v}{\partial \eta} \end{bmatrix} \begin{pmatrix} P \\ Q \end{pmatrix} = \bar{\boldsymbol{R}} \tag{3.19}$$

for each vertex in the surface grid. In this equation $\bar{\boldsymbol{R}}$ is defined as

$$\bar{\boldsymbol{R}} := J^2 \Delta_2 \boldsymbol{u} + 2\left(\frac{\partial \mathcal{S}}{\partial \xi}\frac{\partial \mathcal{S}}{\partial \eta}\right)\frac{\partial^2 \boldsymbol{u}}{\partial \xi \partial \eta} - \left(\frac{\partial \mathcal{S}}{\partial \eta}\right)^2 \frac{\partial^2 \boldsymbol{u}}{\partial \xi^2} - \left(\frac{\partial \mathcal{S}}{\partial \xi}\right)^2 \frac{\partial^2 \boldsymbol{u}}{\partial \eta^2}.$$

Before P and Q are actually used in equation (3.18), they are smoothed by applying

$$\begin{aligned} P_{\mathrm{i,j}} &= \frac{1}{2}\left(P_{\mathrm{i,j+1}} + P_{\mathrm{i,j-1}}\right) \qquad \forall\, \mathrm{i}, \quad 1 < \mathrm{j} < \mathrm{n}_2 - 2 \quad \text{and} \\ Q_{\mathrm{i,j}} &= \frac{1}{2}\left(Q_{\mathrm{i+1,j}} + Q_{\mathrm{i-1,j}}\right) \qquad \forall\, \mathrm{j}, \quad 1 < \mathrm{i} < \mathrm{n}_1 - 2 \end{aligned}$$

for a fixed number of iterations. The result is that P, which controls the spacing in ξ-direction, is only smoothed in the η-direction and Q, which controls the spacing in η-direction is only smoothed in the ξ-direction. In this way, the grid spacing of the initial grid in the direction normal to the boundaries is maintained, while the orthogonality of the grid lines is improved with respect to the initial grid. Alternatives for the specification of the control functions, e.g. when different characteristics than smoothness and orthogonality are important, can be found in the literature [96, 174].

Equation (3.18) is discretized, in order to solve it numerically. For this purpose, the derivatives of the parametric variables with respect to ξ and η are replaced by a

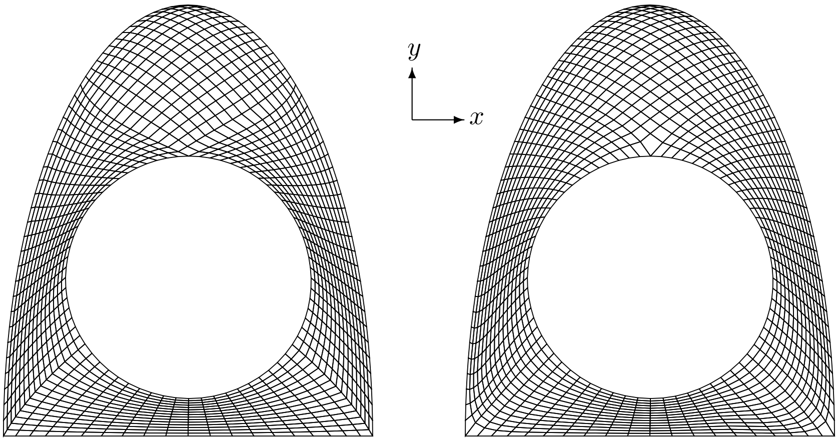


Figure 3.3: Surface grid constructed for the spinner of a wind turbine rotor. Left grid, constructed by means of linear transfinite interpolation, is used as starting point for the elliptical grid generation method, of which the result is depicted on the right. Initial grid shows large differences in spacing and regions with only C^0-continuity of grid lines.

second-order central-difference finite-difference approximation. Since $\Delta\xi = \Delta\eta \equiv 1$, this substitution results in the following discrete version of the elliptical surface grid equations:

$$\begin{aligned}&\left[\frac{\partial \mathcal{S}}{\partial \eta}\right]^2 \left([\boldsymbol{u}_{\mathrm{i+1,j}} - 2\boldsymbol{u}_{\mathrm{i,j}} + \boldsymbol{u}_{\mathrm{i-1,j}}] + \frac{P_{\mathrm{i,j}}}{2} [\boldsymbol{u}_{\mathrm{i+1,j}} - \boldsymbol{u}_{\mathrm{i-1,j}}] \right) - \\ 2&\left[\frac{\partial \mathcal{S}}{\partial \xi}\frac{\partial \mathcal{S}}{\partial \eta}\right] (\boldsymbol{u}_{\mathrm{i+1,j+1}} - \boldsymbol{u}_{\mathrm{i-1,j+1}} - \boldsymbol{u}_{\mathrm{i+1,j-1}} + \boldsymbol{u}_{\mathrm{i-1,j-1}}) + \\ &\left[\frac{\partial \mathcal{S}}{\partial \xi}\right]^2 \left([\boldsymbol{u}_{\mathrm{i,j+1}} - 2\boldsymbol{u}_{\mathrm{i,j}} + \boldsymbol{u}_{\mathrm{i,j-1}}] + \frac{Q_{\mathrm{i,j}}}{2} [\boldsymbol{u}_{\mathrm{i,j+1}} - \boldsymbol{u}_{\mathrm{i,j-1}}] \right) = J^2 \Delta_2 \boldsymbol{u}.\end{aligned} \tag{3.20}$$

The partial derivatives of the parametric surface with respect to ξ and η can be expanded, using the chain rule of differentiation. Subsequently, the partial derivatives of $\mathcal{S}$ with respect to u and v are determined analytically, by differentiating the expression for the NURBS surface, presented in chapter 2, i.e. equation (2.6) on page 25. For the remaining partial derivatives, those of u and v with respect to ξ and η, a finite-difference approximation is employed. With this approach, the remaining terms in equation (3.20) — i.e. $\frac{\partial \mathcal{S}}{\partial \xi}$, $\frac{\partial \mathcal{S}}{\partial \eta}$ and $J^2\Delta_2\boldsymbol{u}$ — are evaluated for $\boldsymbol{u}_{\mathrm{i,j}}$. The resulting equation is solved iteratively using Gauss-Seidel [150, 196] relaxation $\forall \boldsymbol{u} \in \mathbb{U}_{\mathrm{i}}$. When a certain convergence criterion is met, a surface grid, created by means of the elliptical surface grid generation method, is obtained. An example of a surface grid that is created using this method, is depicted in figure 3.3.

Subsequently, the surface grid — obtained using either the linear transfinite interpolation or the elliptical surface grid generation method — is used as an initial condition for the generation of the hyperbolic field grid. This method is presented in the next section.

3.3 Field grid generation

As discussed in section 3.1.2, a hyperbolic grid generation method is used for the discretization of the flow domain. This section treats the particular method by first presenting the equations that need to be solved to generate a field grid. Subsequently, the numerical solution method used for solving the set of partial differential equations is discussed. Thereafter, details on the implementation of the method are presented.

3.3.1 Hyperbolic field grid equations

Consider a grid for which the coordinate $\boldsymbol{r} \equiv (x, y, z)^T \in \mathbb{R}^3$ of the vertices of the grid are obtained from a mapping of the computational space to the physical space. The computational coordinate is represented by $(\xi, \eta, \zeta)^T \in \mathbb{R}^3$, where ξ and η are the in-plane directions and ζ is the marching direction, i.e. the direction perpendicular — perpendicular in the computational space — to the surface. For a grid of good quality, the grid lines must be close to orthogonal. This requirement of orthogonality can be expressed as [176]:

$$\frac{\partial \boldsymbol{r}}{\partial \xi} \cdot \frac{\partial \boldsymbol{r}}{\partial \zeta} = 0, \tag{3.21}$$

$$\frac{\partial \boldsymbol{r}}{\partial \eta} \cdot \frac{\partial \boldsymbol{r}}{\partial \zeta} = 0. \tag{3.22}$$

Apart from orthogonality of the grid lines, a good grid quality also requires a smooth variation in the spacing of the grid lines. The spacing between the grid lines can be enforced through the specification of the volume of the grid cells for the subsequent grid layer, which is achieved by requiring

$$\frac{\partial \boldsymbol{r}}{\partial \zeta} \cdot \left(\frac{\partial \boldsymbol{r}}{\partial \xi} \times \frac{\partial \boldsymbol{r}}{\partial \eta} \right) \Delta\xi \Delta\eta \Delta\zeta = V \tag{3.23}$$

for the volume $V \in \mathbb{R}$ of a particular cell. Note, that there is always a unit increment in one direction, between the computational coordinates of two neighbouring vertices, hence $\Delta\xi = \Delta\eta = \Delta\zeta \equiv 1$. To determine the volume that needs to be specified to satisfy a specific grid spacing in ζ-direction, the area of the cell face of the current layer is used. Equations (3.21) to (3.23) compose the equations for hyperbolic grid generation. This set of partial differential equations needs to be discretized, in order to be able to solve them numerically. The method used for the discretization of these equations is presented next.

3.3.2 Spatial discretization

To discretize the partial differential equations given in equations (3.21), (3.22) and (3.23), the equations are first linearized around a given state $(\cdot)_\mathrm{k}$. Linearization is achieved by substitution of

$$\boldsymbol{r} = \boldsymbol{r}_\mathrm{k} + \boldsymbol{\Delta r}$$

in equations (3.21) to (3.23). Upon dropping the higher-order terms, these equations can be written as a system of linear equations, as follows:

$$\underline{\underline{A}}_{\mathrm{k}} \frac{\partial}{\partial \xi} \left(\boldsymbol{r} - \boldsymbol{r}_{\mathrm{k}}\right) + \underline{\underline{B}}_{\mathrm{k}} \frac{\partial}{\partial \eta} \left(\boldsymbol{r} - \boldsymbol{r}_{\mathrm{k}}\right) + \underline{\underline{C}}_{\mathrm{k}} \frac{\partial}{\partial \zeta} \left(\boldsymbol{r} - \boldsymbol{r}_{\mathrm{k}}\right) = \left(0, 0, V - V_{\mathrm{k}}\right)^T , \tag{3.24}$$

where V_{k} represents the volume of the cell from the previous grid layer. Moreover, matrices $\underline{\underline{A}}$, $\underline{\underline{B}}$ and $\underline{\underline{C}}$ — in equation (3.24) evaluated for $\boldsymbol{r}_{\mathrm{k}}$ — are given by

$$\underline{\underline{A}} := \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \\ 0 & 0 & 0 \\ \left(\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) & \left(\frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta}\right) & \left(\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\right) \end{bmatrix}, \tag{3.25}$$

$$\underline{\underline{B}} := \begin{bmatrix} 0 & 0 & 0 \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \\ \left(\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \zeta}\right) & \left(\frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \xi}\right) & \left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi}\right) \end{bmatrix}, \tag{3.26}$$

$$\underline{\underline{C}} := \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \xi}\right) & \left(\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta}\right) & \left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right) \end{bmatrix}. \tag{3.27}$$

The system of linear equations of (3.24) can be rewritten to

$$\underline{\underline{A}}_{\mathrm{k}} \frac{\partial \boldsymbol{r}}{\partial \xi} + \underline{\underline{B}}_{\mathrm{k}} \frac{\partial \boldsymbol{r}}{\partial \eta} + \underline{\underline{C}}_{\mathrm{k}} \frac{\partial \boldsymbol{r}}{\partial \zeta} = \boldsymbol{f}, \tag{3.28}$$

for which the right hand side vector reads

$$\boldsymbol{f} = \left(0, 0, V + 2V_{\mathrm{k}}\right)^T , \tag{3.29}$$

since

$$\underline{\underline{A}}_{\mathrm{k}} \left.\frac{\partial \boldsymbol{r}}{\partial \xi}\right|_{\boldsymbol{r}_{\mathrm{k}}} + \underline{\underline{B}}_{\mathrm{k}} \left.\frac{\partial \boldsymbol{r}}{\partial \eta}\right|_{\boldsymbol{r}_{\mathrm{k}}} + \underline{\underline{C}}_{\mathrm{k}} \left.\frac{\partial \boldsymbol{r}}{\partial \zeta}\right|_{\boldsymbol{r}_{\mathrm{k}}} \equiv \left(0, 0, 3V_{\mathrm{k}}\right)^T .$$

Equation (3.28) can be used to obtain an expression for $\frac{\partial \boldsymbol{r}}{\partial \zeta}$, i.e.

$$\frac{\partial \boldsymbol{r}}{\partial \zeta} = \left(\underline{\underline{C}}_{\mathrm{k}}\right)^{-1} \left[\boldsymbol{f} - \left(\underline{\underline{A}}_{\mathrm{k}} \frac{\partial}{\partial \xi} + \underline{\underline{B}}_{\mathrm{k}} \frac{\partial}{\partial \eta}\right) \boldsymbol{r}\right] . \tag{3.30}$$

From the linear combination of equations (3.21) to (3.23) it follows that $\frac{\partial \boldsymbol{r}}{\partial \zeta}$ can also be expressed as

$$\left.\frac{\partial \boldsymbol{r}}{\partial \zeta}\right|_{\boldsymbol{r}_{\mathrm{k}}} = \left(\underline{\underline{C}}_{\mathrm{k}}\right)^{-1} \begin{pmatrix} 0 \\ 0 \\ V_{\mathrm{k}} \end{pmatrix} , \tag{3.31}$$

for grid layer k. Next, the partial derivatives of $\boldsymbol{r}$ with respect to ξ and η in equation (3.28) are discretized by replacing them by finite-difference operators, respectively denoted by δ_ξ and δ_η. The semi-discrete equation for grid layer $\mathrm{k}+1$ then reads

$$\left(\underline{\underline{A}}_{\mathrm{k}} \delta_\xi + \underline{\underline{B}}_{\mathrm{k}} \delta_\eta\right) \boldsymbol{r}_{\mathrm{k}+1} + \underline{\underline{C}}_{\mathrm{k}} \left.\frac{\partial \boldsymbol{r}}{\partial \zeta}\right|_{\boldsymbol{r}_{\mathrm{k}+1}} = \boldsymbol{f}_{\mathrm{k}+1} . \tag{3.32}$$

Using equations (3.30) and (3.31), this expression can be written as

$$\left(\underline{\underline{C}}_{\mathrm{k}}\right)^{-1}\left[\underline{\underline{A}}_{\mathrm{k}}\delta_{\xi}+\underline{\underline{B}}_{\mathrm{k}}\delta_{\eta}\right]\left(\boldsymbol{r}_{\mathrm{k+1}}-\boldsymbol{r}_{\mathrm{k}}\right)+\left.\frac{\partial\boldsymbol{r}}{\partial\zeta}\right|_{\boldsymbol{r}_{\mathrm{k+1}}}=\left(\underline{\underline{C}}_{\mathrm{k}}\right)^{-1}\begin{pmatrix}0\\0\\V_{\mathrm{k+1}}\end{pmatrix}. \tag{3.33}$$

Following the method first proposed by Kinsey et al. [98] for 2D hyperbolic grids and later on extended by Chan [36, 39] to 3D hyperbolic grids, the partial derivative with respect to ζ is discretized using an implicit weighted average method. After approximate factorization and the introduction of numerical dissipation to both the implicit part as well as the right hand side, the discrete equations read

$$\begin{aligned}\left[\underline{\underline{I}}+\left(1+\bar{\theta}\right)\underline{\underline{C}}_{\mathrm{k}}^{-1}\underline{\underline{B}}_{\mathrm{k}}\delta_{\eta}-2\epsilon_{\eta}\delta_{\eta}^{2}\right]\left[\underline{\underline{I}}+\left(1+\bar{\theta}\right)\underline{\underline{C}}_{\mathrm{k}}^{-1}\underline{\underline{A}}_{\mathrm{k}}\delta_{\xi}-2\epsilon_{\xi}\delta_{\xi}^{2}\right]\left(\boldsymbol{r}_{\mathrm{k+1}}-\boldsymbol{r}_{\mathrm{k}}\right)=&\\ \underline{\underline{C}}_{\mathrm{k}}^{-1}\begin{pmatrix}0\\0\\V_{\mathrm{k+1}}\end{pmatrix}+\left[\bar{\epsilon}_{\xi}\bar{\delta}_{\xi}^{2}+\bar{\epsilon}_{\eta}\bar{\delta}_{\eta}^{2}\right]\boldsymbol{r}_{\mathrm{k}},&\end{aligned} \tag{3.34}$$

where δ_{ξ}^{2} and δ_{η}^{2} denote finite-difference operators for the second derivative with respect to ξ and η, respectively. The corresponding dissipation coefficients ϵ_{ξ} and ϵ_{η} are used for smoothing of the grid in ξ and η-direction. Moreover, note that an overbar is used to denote the dissipation coefficient in the explicit part of the equation. Similarly, an overbar is used for the second-order finite-difference operators, to accommodate the use of a different finite-difference stencil for the implicit and explicit part of the equations. The dissipation coefficients may also differ for each vertex in the grid; the method used for specifying the value is discussed in the following section. Implicit weighting factor $\bar{\theta}$ is used to cope with concave regions in the flow domain. If no concave regions exist, $\bar{\theta}=0$ is the default choice. Otherwise, a value between 0 and 5 can be used.

3.3.3 Implementation details

Before equation (3.34) can be solved, a number of terms appearing in this equation must be specified. Details of the treatment of these terms are presented below.

Finite-difference operators

Finite-difference operators appear in both the explicit part as well as in the implicit part of equation (3.34). For the explicit part, a central-difference stencil, with fourth-order spatial accuracy, is employed for vertices away from the boundary. Near the boundaries, a central-difference stencil with a lower order spatial accuracy is employed; except when the particular boundary condition allows for maintaining the fourth-order stencil, see section 3.3.4.

In the implicit part of the equations, a central-difference stencil with second-order spatial accuracy is used to represent the finite-difference operator. This choice was made to limit the complexity of constructing the matrix involved in the system of linear equations.

Cell volume and grid spacing

Specification of the grid spacing is enforced by prescribing the cell volume. The cell volume that is prescribed is computed based on the surface area of the face of the previous layer and the required spacing. In this computation, it is assumed that the face is planar and that the marching direction is normal to the plane of the face. The grid spacing that is prescribed is governed by two parameters: (i) the spacing between the first and second grid layer, $|\Delta \boldsymbol{r}|_0$ and (ii) the spacing between the before last and final grid layer, $|\Delta \boldsymbol{r}|_{\mathrm{n}_3-2}$. Based on these parameters, the spacing is defined by:

$$|\Delta \boldsymbol{r}|_{\mathrm{k}} = |\Delta \boldsymbol{r}|_{\mathrm{n}_3-2}\left[1-\tanh\left(\bar{\gamma}\left[1-\frac{\mathrm{k}}{\mathrm{n}_3-2}\right]\right)\right], \tag{3.35}$$

with

$$\bar{\gamma} := \operatorname{arctanh}\left(\frac{|\Delta \boldsymbol{r}|_{\mathrm{n}_3-2}-|\Delta \boldsymbol{r}|_0}{|\Delta \boldsymbol{r}|_{\mathrm{n}_3-2}}\right).$$

A favourable property of this spacing function is that the variation in grid spacing is smooth. Moreover, this approach allows for the explicit specification of both $|\Delta \boldsymbol{r}|_0$ and $|\Delta \boldsymbol{r}|_{\mathrm{n}_3-2}$, such that an adequate grid resolution can be achieved near the body as well as a compatible off-body spacing for regions of overlapping grids. The latter property does not apply to, for instance, a geometric stretching function, because for that method the off-body spacing is a direct consequence of the near-body grid resolution, the number of grid layers used and the chosen growth factor. Therefore, the present approach provides more flexibility. Moreover, note that a compatible off-body spacing in regions of overlapping grids is an important property, when composite overset grids are used to obtain a flow solution, because a compatible grid spacing in the region of overlap yields more accurate flow solutions [206] and better convergence properties of the numerical flow solution method. Figure 3.4 shows an example of this stretching function for $|\Delta \boldsymbol{r}|_0 = 0.05$, $|\Delta \boldsymbol{r}|_{\mathrm{n}_3-2} = 0.20$ and $\mathrm{n}_3 = 30$.

In order to compute the cell volume, the required spacing — determined using the stretching function — is multiplied by the face area of a cell. Subsequently, this result is transferred to the vertex by averaging the volume over the faces neighbouring the vertex. This value can be used in the construction of the system of linear equations. However, the smoothness of the grid can be enhanced by taking a weighted average of the volume computed for the neighbouring vertices, as [36]

$$\Delta \bar{V}_{\mathrm{i,j}} = (1-v_{\mathsf{a}})\,\Delta V_{\mathrm{i,j}} + \frac{v_{\mathsf{a}}}{4}\left(\Delta V_{\mathrm{i-1,j}} + \Delta V_{\mathrm{i,j-1}} + \Delta V_{\mathrm{i,j+1}} + \Delta V_{\mathrm{i+1,j}}\right), \tag{3.36}$$

where $\Delta \bar{V}_{\mathrm{i,j}}$ is the averaged result and $v_{\mathsf{a}} \in \mathbb{R}$ is a weighting factor for which a value of $^4/_{25}$ is used. This procedure can be repeated multiple times to further increase the amount of smoothing.

Dissipation coefficient

Smoothing is also achieved by the introduction of a numerical dissipation term in the equations, by means of a second-order finite-difference operator. Apart from increasing the smoothness of the resulting grid, the second-order finite-difference operator also prevents the occurrence of numerical instabilities due to odd-even decoupling. The amount

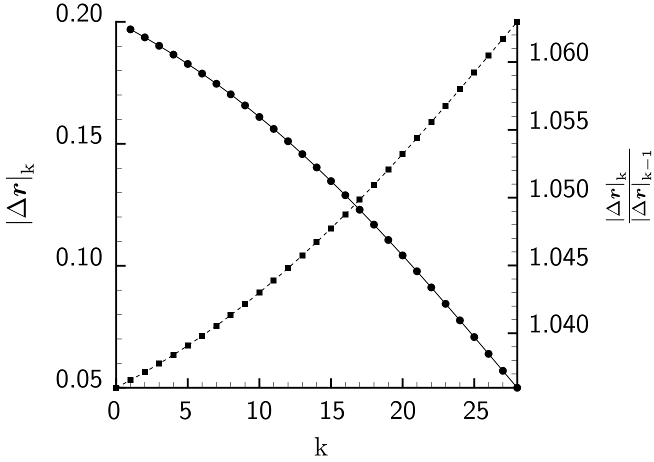


Figure 3.4: Example of stretching function defined by equation (3.35) for $|\Delta \boldsymbol{r}|_0 = 0.05$, $|\Delta \boldsymbol{r}|_{\mathrm{n}_3-2} = 0.20$ and $\mathrm{n}_3 = 30$. Dashed line with square shaped markers represents the grid spacing. Solid line with circular shaped markers represents the growth factor between two consecutive grid layers.

of dissipation is controlled by the dissipation coefficients. The required dissipation depends on local properties of the grid. The method described by Chan and Steger [39] is used to account for this observation. Following their approach, the dissipation coefficient for the implicit part can be expressed as

$$(\epsilon_\xi)_{\mathrm{i,j,k}} = \epsilon_{\mathrm{i}} f\left(\boldsymbol{r}, \left.\frac{\partial \boldsymbol{r}}{\partial \xi}\right|_{\mathrm{i,j,k}}, \left.\frac{\partial \boldsymbol{r}}{\partial \zeta}\right|_{\mathrm{i,j,k}}, \mathrm{k}\right) \quad \text{and} \tag{3.37}$$

$$(\epsilon_\eta)_{\mathrm{i,j,k}} = \epsilon_{\mathrm{i}} f\left(\boldsymbol{r}, \left.\frac{\partial \boldsymbol{r}}{\partial \eta}\right|_{\mathrm{i,j,k}}, \left.\frac{\partial \boldsymbol{r}}{\partial \zeta}\right|_{\mathrm{i,j,k}}, \mathrm{k}\right), \tag{3.38}$$

i.e. a constant $\epsilon_{\mathrm{i}} \in \mathbb{R}$ times a function f that depends on the derivative with respect to the direction considered, the derivative in ζ-direction and the index of the present layer. For the details on the exact formulation of this function, see appendix A or the original description [39]. The dissipation coefficient for the explicit part is subsequently computed, using the result for the implicit part, employing a different proportionality factor $\epsilon_{\mathrm{e}} \in \mathbb{R}$, i.e.

$$(\bar{\epsilon}_\xi)_{\mathrm{i,j,k}} = \frac{\epsilon_{\mathrm{e}}}{\epsilon_{\mathrm{i}}} (\epsilon_\xi)_{\mathrm{i,j,k}} \text{ and } (\bar{\epsilon}_\eta)_{\mathrm{i,j,k}} = \frac{\epsilon_{\mathrm{e}}}{\epsilon_{\mathrm{i}}} (\epsilon_\eta)_{\mathrm{i,j,k}}.$$

For the proportionality factor of the implicit part, a default value of 1.0 is used and 0.5 for the explicit part. These default values can be changed independently when more or less numerical dissipation is required.

Solution method

Considering equation (3.34), it is observed that a block tridiagonal matrix is encountered, for which a very efficient solution algorithm exists, i.e. the Thomas algorithm [127, 183].

However, the possibility of using different boundary conditions, which locally alters the structure of the matrix, prevents the straightforward application of the Thomas algorithm. Therefore, the choice was made to use a more generally applicable method for solving systems of linear equations, to compute the next grid layer. Given the generally limited dimension of the system of equations, which is dictated by the dimension of the surface grid, it is feasible to use a direct method, for instance utilizing an LU decomposition, to solve the problem. However, in the present implementation, a preconditioned GMRES method is used instead, because it was found to be more efficient than performing an LU factorization of the matrix and subsequently performing a forward and backward substitution. The particular iterative method employed for this purpose is discussed in more detail in section 5.6.2. Note on the other hand, that for computing grid sensitivities, the use of a direct method can be advantageous over an iterative method, see section 6.9 for more details. Moreover, with some additional work, it is still possible to employ the Thomas algorithm [164]. However, considering the very limited amount of CPU-time required for the grid generation, relative to for instance obtaining the flow solution, the effect of this improvement on the efficiency of the over-all optimization method would be minor and is therefore not pursued further in the present research.

3.3.4 Boundary conditions

For hyperbolic grid generation, one boundary — the ζ_{min} boundary at $\zeta = 0$ — is specified by the surface grid. The ζ_{max} boundary, at $\zeta = \mathrm{n}_3 - 1$, directly follows from solving the hyperbolic grid equations for the number of layers specified. Like for a physical model governed by partial differential equations, the equations for the grid generation require boundary conditions for the remaining four boundaries. The boundary conditions employed in the present research are: symmetry boundary conditions, periodic boundary conditions and so-called splay boundary conditions. The first two of these boundary conditions allow for maintaining the fourth-order central-difference approximation of the derivatives. For vertices near splay boundary conditions on the other hand, a second-order central difference stencil is used instead.

Symmetry

Symmetry boundary conditions are used to create a grid that is mirror-symmetric about a symmetry plane, represented by the boundary of the grid. This boundary condition is enforced by mirroring the vertices that occur in the finite-difference stencils about the symmetry plane, as

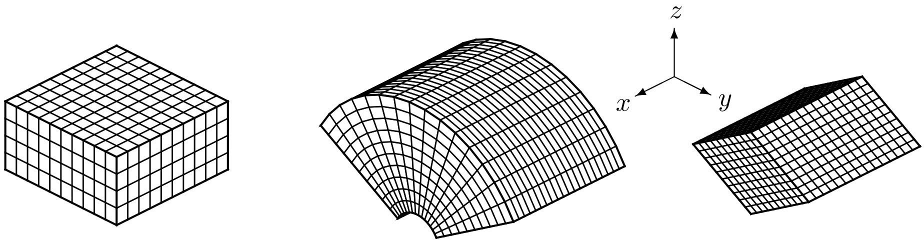
$$\boldsymbol{r}_{-\mathrm{i},\mathrm{j}} = 2\boldsymbol{r}_{0,\mathrm{j}} - \boldsymbol{r}_{\mathrm{i},\mathrm{j}} \tag{3.39}$$

for a symmetry boundary condition at the ξ_{min} boundary. For application of this boundary condition at a different boundary, the approach is similar.

Periodicity

Periodic boundary conditions are used for domains that exhibit spatial periodicity. Translational periodicity for the ξ_{min} is enforced by using

$$\boldsymbol{r}_{-\mathrm{i},\mathrm{j}} = \boldsymbol{r}_{0,\mathrm{j}} - \left(\boldsymbol{r}_{\mathrm{n}_1-1,\mathrm{j}} - \boldsymbol{r}_{\mathrm{n}_1-(\mathrm{i}+1),\mathrm{j}}\right), \tag{3.40}$$



(a) (b) (c)

Figure 3.5: Example of different background grids: (a) Cartesian background grid; (b) partial cylindrical background grid, 120°; (c) rhombus-shaped background grid.

in the finite-difference stencil. In this case, the same approach must be taken for the ξ_{max} boundary.

For enforcing rotational periodicity at the ξ_{min} boundary

$$\boldsymbol{r}_{-\mathrm{i,j}} = \boldsymbol{r}_{0,\mathrm{j}} - \underline{\underline{T}}\,(\vartheta)\left[\boldsymbol{r}_{\mathrm{n_1}-1,\mathrm{j}} - \boldsymbol{r}_{\mathrm{n_1}-(\mathrm{i}+1),\mathrm{j}}\right] \tag{3.41}$$

is used to obtain the appropriate coordinate to satisfy the finite-difference stencil at the boundary. In this equation $\underline{\underline{T}}\,(\vartheta)$ is the rotation matrix corresponding to the angle $\vartheta \in \left\{\frac{2\pi}{\mathrm{c}} \,:\, \mathrm{c} \in \mathbb{N}_1\right\}$ for which rotational periodicity is observed.

Splay

The so-called splay boundary conditions are used for ensuring sufficient overlap between different blocks, to accommodate overset block connectivity. In contrast to the previous two boundary conditions, this boundary condition is enforced by making the increment in $\boldsymbol{r}$ from one grid layer to the next of the boundary vertex linearly dependent on the increment in $\boldsymbol{r}$ for interior vertices, neighbouring the boundary vertex. This approach can be expressed as

$$\Delta\boldsymbol{r}_{0,\mathrm{j}} = \Delta\boldsymbol{r}_{1,\mathrm{j}} + \alpha_{\mathsf{splay}}\left(\Delta\boldsymbol{r}_{1,\mathrm{j}} - \Delta\boldsymbol{r}_{2,\mathrm{j}}\right); \tag{3.42}$$

again for the ξ_{min} boundary and where $0 \leq \alpha_{\mathsf{splay}} \leq 1$ is the extrapolation factor, which can be used to adjust the amount of splay.

3.4 Background grids

The composite overset grid method allows for using a different grid in the off-body region of the flow domain. Since these blocks, which are commonly referred to as background grids, do not need to conform with the body, the construction of these grids is straightforward. Usually an algebraic generation method provides satisfactory results. In this section the background grids available in the present grid generation method are briefly discussed.

3.4.1 Cartesian

Figure 3.5 (a) shows a uniform Cartesian background grid. Cartesian background grids can be generated with any arbitrary orientation. For the spacing in each direction either a uniform or a non-uniform spacing can be used. The non-uniform spacing can be employed to realize a compatible spacing in the region of overlap, while maintaining a coarser spacing near the far-field boundaries of the domain.

3.4.2 Cylindrical

Cylindrical background grids, depicted in figure 3.5 (b), are the obvious choice for the discretization of the off-body region of cylindrical domains. For the construction of a cylindrical background grid, the inner and outer radius must be specified. The inner radius should be chosen larger than zero, to prevent the occurrence of a singular grid line. For rotational periodic problems, a sector of the cylinder can be used, instead of the complete cylinder. The spacing in each of the directions can be chosen to be either uniform or non-uniform, depending on the requirements.

3.4.3 Rhombus

The rhombus-shaped background grid, shown in figure 3.5 (c), is similar to the Cartesian background grid, apart from the fact that the grid lines in the three directions are not mutually orthogonal. With this feature, the rhombus-shaped background grid can be used for the discretization of the rotationally periodic flow domain in the region near the axis of rotation of a wind turbine rotor with more than two blades. In this way the cylindrical grid does not need to be used in this region, avoiding the occurrence of a singular grid line.

3.5 Summary and results

The requirements for the grid and the grid generation method in an aerodynamic shape optimization problem have been considered. The most important requirements identified are that grid quality is maintained throughout the optimization procedure and that the grid can be generated efficiently. Based on these requirements, the choice was made to use a hyperbolic grid generation method for the discretization of the flow domain. Surface grids are generated using either linear transfinite interpolation or an elliptical surface grid generation method. For the purpose of generating a surface grid, first the edges of the surface grid need to be discretized. Two methods are available for this purpose. One method takes the local curvature into account for the distribution of the vertices, applying a higher vertex density in regions of high curvature. For the other method, the vertex density is user controlled.

The surface grid, generated on the geometry of the object in the flow domain, is used as a boundary condition for the hyperbolic field grid generation method, for which the equations have been presented. Discretization of these equations has been discussed and details on the implementation have been given.

Background grids are used in composite overset grids for discretization of the off-body regions of the flow domain. Three different background grid topologies available in the present grid generation method have been presented, viz. Cartesian, cylindrical and rhombus-shaped background grids.

To give an idea of the results that can be obtained by the method presented in this chapter, an example is provided here. Figure 3.6 shows part of the body-fitted grid with

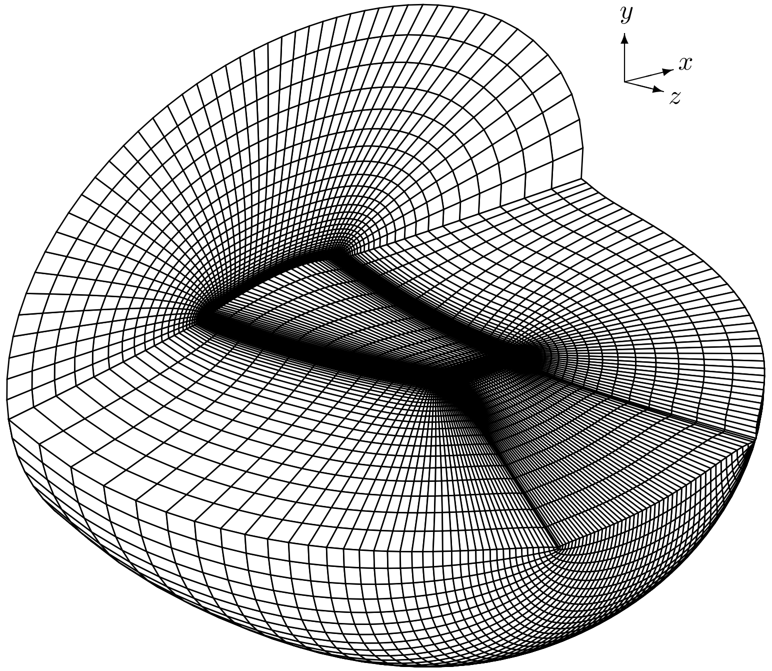


Figure 3.6: Example of a body-fitted grid with CO-topology around a wing. Note, that the x-direction is chordwise and the z-direction is spanwise.

a CO-topology around a wing. To also quantify the quality of the grid, the orthogonality of grid lines is investigated. As a measure for the orthogonality of the grid lines, the following quantity is defined

$$\Phi := \prod_{\mathrm{i}=0}^{\mathrm{i}=7} \left([\boldsymbol{u}_\mathrm{i} \times \boldsymbol{v}_\mathrm{i}] \cdot \boldsymbol{w}_\mathrm{i} \right) , \tag{3.43}$$

where $\boldsymbol{u}_\mathrm{i}$, $\boldsymbol{v}_\mathrm{i}$ and $\boldsymbol{w}_\mathrm{i}$ are unit vectors pointing from vertex i, each in a direction parallel to the edges of the cell, that intersect at the vertex. The order of vectors is chosen such that together they form a right-handed coordinate system. Considering the definition of Φ, the result will always have a positive value for valid cells. Moreover, the result is equal to one for perfectly orthogonal cells and close to zero for cells that are highly skewed.

Following the approach presented by Hicken [82], the range of possible values for Φ, ranging from 0 to 1, is divided in 100 equally sized bins. Subsequently, the orthogonality measure of equation (3.43) is computed for every cell and its corresponding bin is determined. This procedure results in an orthogonality distribution for the grid considered. For the grid shown in figure 3.6 the orthogonality distribution is depicted in figure 3.7. Considering this result, the peak near $\Phi = 1$ shows that the grid generation method is successful in realizing orthogonality of the grid lines for the majority of the cells. A striking detail in the orthogonality distribution graph is, however, the slight peek occurring

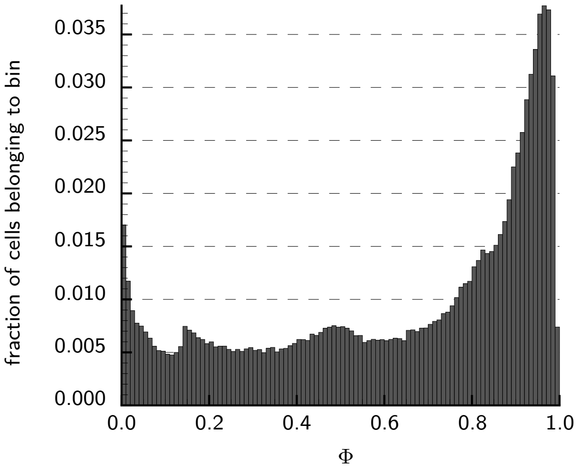


Figure 3.7: Distribution of the orthogonality, defined by equation (3.43), for the cells of the grid depicted in figure 3.6.

near $\Phi = 0$. This peak means that quite a number of cells exist for which orthogonality of the grid lines is not achieved. The cause for this peak is found in the use of a grid with an O-grid topology for a wing with a sharp trailing edge. Moreover, the grid depicted is quite coarse, i.e. coarser than what would be used for computing an accurate flow solution, which also has a negative effect on the quality of the grid.

4

OVERSET BLOCK CONNECTIVITY

> *"To horses beyond all mortal creatures cunning Nature has given a subtle mind and heart. [...] A horse carried above the clouds him that slew the Chimaera;"*
>
> — OPPIAN OF APAMEA (3^{rd} century A.D.), *Cynegetica*

USING composite overset grids for the discretization of the flow domain offers many opportunities. However, it also provides a number of challenges. One of the main challenges encompasses determining a suitable block connectivity for multiple overlapping grids. Moreover, the evaluation of surface integrals is also not straightforward, when the same part of the geometry is discretized by multiple surface grids. These matters, among others, are addressed in this chapter, together with the approach taken in the present research to deal with these challenges.

First, a concise overview of the history and current status of the composite overset grid technology is presented. Subsequently, the concept and terminology of overset grids is introduced. Then, the method used for efficiently performing a search procedure is treated in section 4.2. The particular method used is called an alternating binary tree search. Thereafter, the zipper grid method is discussed, which is employed to accurately evaluate surface integrals and to provide a non-overlapping closed surface grid. This non-overlapping surface grid is used by the ray-casting technique, explained in section 4.4, used to identify cells that reside outside the physical flow domain and are therefore not needed for solving the governing equations. Next, the actual method used for determining the block connectivity is presented. A so-called implicit hole cutting method is applied for this purpose. The chapter concludes with a summary.

4.1 Introduction

The concept of using composite overset grids for the discretization of the flow domain was first introduced by Atta [10, 11] in 1981. Subsequently, a lot of work on the development of overset grid technology was carried out at NASA by, among others, Benek [16] and Steger [178]. An excellent and complete overview of the advancement on this area of research throughout the years within NASA was given by Chan [38]. Since the introduction of the original concept, many different methods have been developed for determining the block connectivity for composite overset grids; a good overview of common methods currently in use has been given by Lee [107].

Research in this field has mainly been focused in the United States of America. Well known methods developed include the structured grid Navier-Stokes method Over-

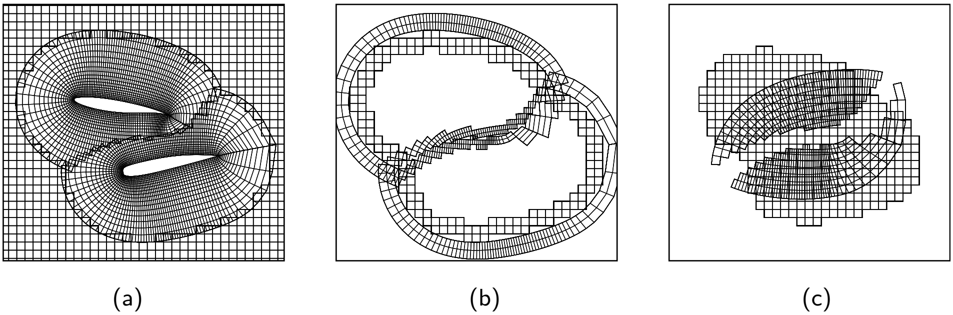

(a) (b) (c)

Figure 4.1: Composite overset grid consisting of two body-fitted O-grids around aerofoils and a Cartesian background grid. Cells with three possible qualifications are shown: (a) field cells; (b) fringe cells; (c) hole cells.

flow [129], developed at NASA and the unstructured-mesh flow solver Fun3D [18, 19], developed at aircraft manufacturer Boeing. However, driven by the benefits of the use of overset grids, the technique also started to emerge in other parts of the world. For instance, DLR has incorporated an overset grid capability in their unstructured Reynolds-averaged Navier-Stokes TAU code [159, 160, 162]. The same is true for the incompressible Reynolds-averaged Navier-Stokes method developed at Risø National Laboratory and the Danish Technical University, for which the overset capabilities were implemented by Zahle [206]. The French aerospace research centre ONERA also included a capability for handling composite overset grids in their flow solver elsA [27].

4.1.1 Concept and terminology

For solving a partial differential equation on a domain that is discretized by several mutually overlapping grids, boundary conditions need to be imposed on the boundary of each grid block discretizing the domain. The required boundary conditions for a certain block are obtained from the flow solution of the block that overlaps the block considered. The flow solution in the overlapped cell is in that case obtained by means of interpolation of the flow solution from cells of the overlapping block. Cells that obtain a solution by interpolation are designated *fringe cells*. On the other hand, cells for which the solution is obtained by solving the discretized partial differential equations are called *field cells*. Because the overlap between blocks is in general not only restricted to the region near the boundary, parts of blocks can be discarded and are therefore not required for obtaining the flow solution. Cells for which this condition applies are called *hole cells*. These are the three qualifications that can be given to cells used in a composite overset grid; figure 4.1 shows an example of each of these qualifications for a composite overset grid.

To specify the qualification of each cell, a so-called iblank-array is used. This iblank-array is an array of signed integers that can take three different values: 1, 0 and -1. The value 1 is used to indicate that a cell is a field cell. Zero is used for hole cells and consequently -1 designates a cell being a fringe cell.

Apart from the three above cell qualifications, a cell can also be qualified as *donor cell*. This qualification can only be assigned to field cells and as the name indicates, the flow solution of a donor cell is used for the interpolation of the dependent variables of a fringe cell.

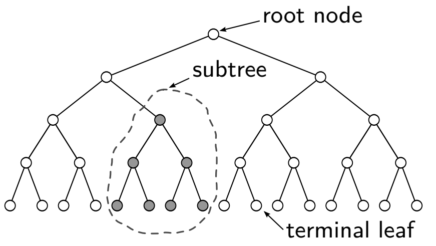


Figure 4.2: Schematic representation of a perfectly balanced binary tree with the corresponding terminology.

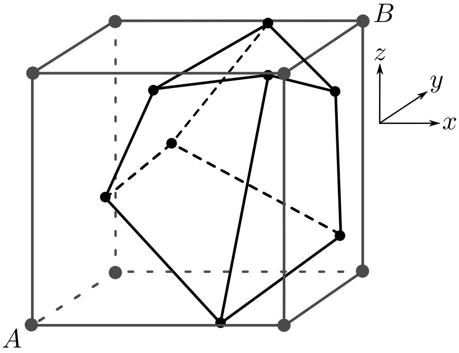


Figure 4.3: Arbitrary hexahedral cell with its corresponding bounding box, specified by the coordinate of vertex A and B.

4.2 Alternating binary tree search

For overset grids, a donor cell needs to be found for each fringe cell, which means that a considerable number of search operations needs to be performed in order to determine the block connectivity. To expedite the search procedure, a so-called alternating binary tree search — also known as an alternating digital tree (ADT) [22] search — is used. In this method, a tree — i.e. an undirected connected graph with no loops or cycles [76], see figure 4.2 for a schematic representation — is used to resemble each compute block of the grid used for the discretization of the flow domain. Each tree encompasses all the cells of the particular block that is represented by the tree, where each cell corresponds to a node in the tree.

4.2.1 Tree generation

The first step in the generation of the alternating binary tree is the determination of a Cartesian bounding box for each cell in the block for which the tree is generated. This bounding box is constructed such that it is just large enough to completely enclose its corresponding hexahedral cell, as illustrated in figure 4.3. The advantage of using a Cartesian bounding box is that it is fully specified by only two coordinates, viz. coordinate $\boldsymbol{a}$ of vertex A and coordinate $\boldsymbol{b}$ of vertex B in the bounding box in figure 4.3. Using a so-called point-based approach [57, 156] reduces the bounding box to a point in 6-dimensional space. Next, all cells are sorted and the resulting ordered set of bounding boxes is bisected. This bisection is performed based on the number of elements in the set and not on for instance, their geometric properties. With this approach the resulting tree is always well balanced, even for non-uniformly distributed data. The sorting is first performed based on the x-component of coordinate $\boldsymbol{a}$. The component of the coordinate for which this sorting is performed changes after each bisection, in a cyclic manner. Once all three components of coordinate $\boldsymbol{a}$ have been used for sorting the bounding boxes, coordinate $\boldsymbol{b}$ is used. For each bisection, a node in the tree is created. Then, one part of the bisected set is assigned to the left child node and the other part to the right child node. Both new sets are then sorted in the next coordinate direction and the above procedure is repeated. This procedure is performed, until a remaining set consists of three bounding boxes or fewer. In the case that there are two bounding boxes remaining, both bounding boxes are assigned to the present node and the node itself is qualified as a so-called terminal leaf, as the tree terminates at that node. If there are, on the other hand, three bounding boxes remaining, the left child node is created as a terminal leaf, containing two bounding boxes. Instead of creating a second child node, the remaining bounding box is then stored in the present node. The result of the whole process is that in the end, all bounding boxes of the current block have been assigned to a terminal leaf. Next, the nodes of the tree need to be assigned a key, to accommodate the search procedure. This key consists of a bounding box that is dimensioned such that it encloses all bounding boxes of the terminal leaves of the subtree that is considered. Once a bounding box is determined for each node in the tree, the generation of the tree is complete.

A separate tree is generated for each block. Trees are generated by the processor for every compute block that has been assigned to that particular processor. The root node of each tree is distributed among the processors taking part in the computation.

4.2.2 Search procedure

The containment search for a point P, with coordinate $\boldsymbol{p}$, starts by checking the containment of P by the bounding box assigned to the root node — which encloses all bounding boxes of the tree — of each tree. If a tree is encountered for which the bounding box of the root node encloses coordinate $\boldsymbol{p}$ and which corresponds to a block that resides on a different processor, the information of the point is transmitted to that processor. If the tree is on the same processor, no additional operations need to be performed. The processor that holds the block, and its corresponding binary tree, will always perform the actual tree search for a particular point.

For a tree that is identified to enclose P, the search procedure is as follows. The root node was already checked to contain P. Next, the bounding box of the left and right child node of the root node are checked if they contain P as well. If the bounding box corresponding to the node contains P, its child nodes are stored. After both nodes of the present level have been handled and at least one of the bounding boxes was found to contain P, the tree is traversed down one level. Then, all bounding boxes corresponding to the child nodes that have been stored, are checked for containment of P. This procedure is repeated until either the end of the tree is reached or none of the bounding boxes of the present level contains P. In the case that a terminal leaf is reached for which one of the bounding boxes contains P, a more accurate check — which is treated in subsection 4.5.2 — is performed that can identify whether or not the hexahedral cell corresponding to the bounding box also contains P. If the accurate check determines that the point is enclosed by the cell, the binary tree search is terminated for that tree, since no more than one cell of a single block can contain P. To elucidate the search procedure just described, a flow chart of the process is provided in figure 4.4.

4.2.3 Other entities

The two preceding sections describe the procedure for the generation of the binary tree and its subsequent use in the containment search procedure for a hexahedral cell of the volume grid. However, with minor modifications, the same framework can be used to

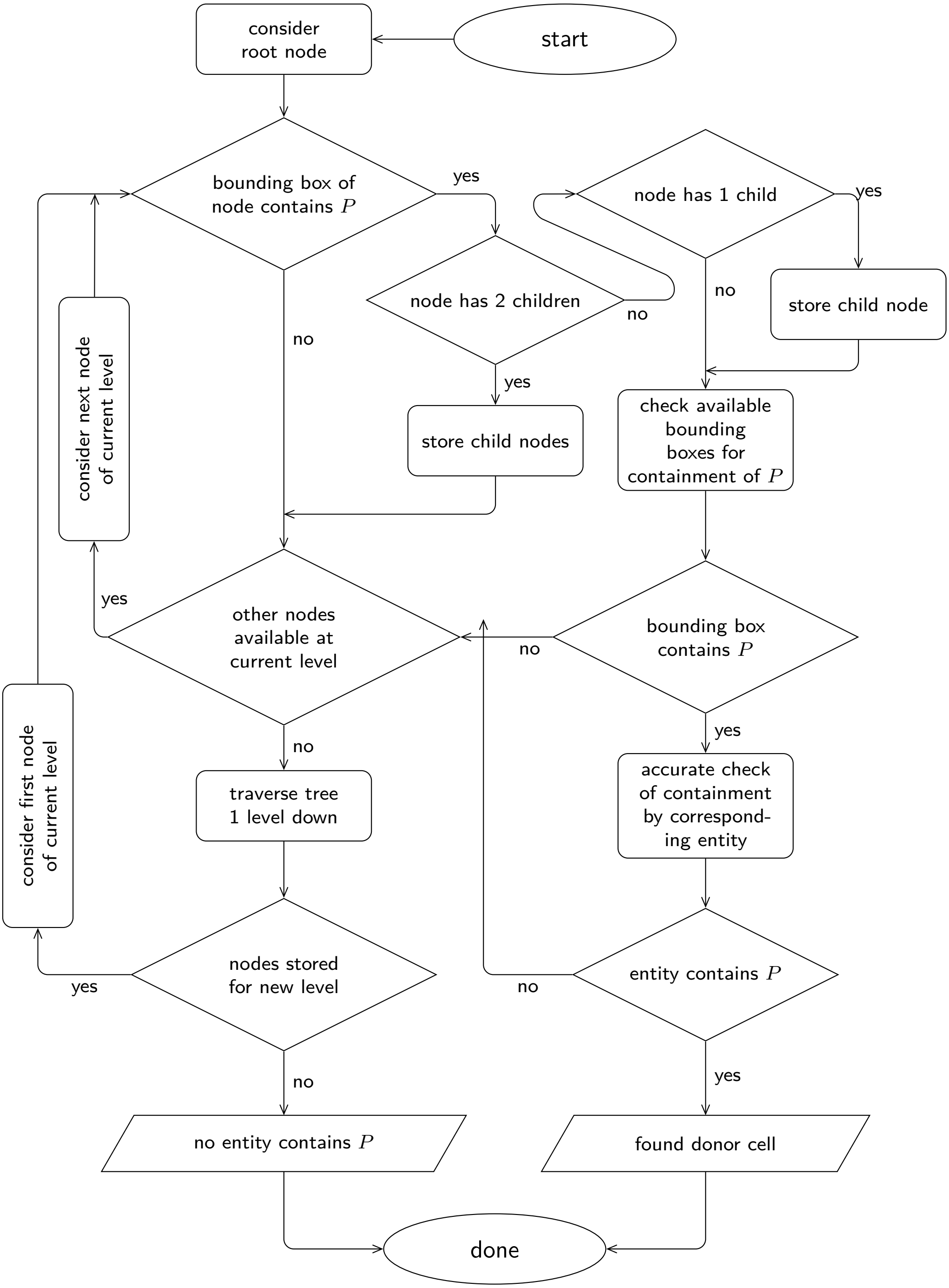

start
consider root node
bounding box of node contains P
yes
no
node has 2 children
yes
no
store child nodes
node has 1 child
yes
no
store child node
check available bounding boxes for containment of P
bounding box contains P
yes
no
accurate check of containment by correspond-ing entity
entity contains P
yes
no
found donor cell
other nodes available at current level
yes
no
consider next node of current level
traverse tree 1 level down
nodes stored for new level
yes
no
consider first node of current level
no entity contains P
done

Figure 4.4: Flow chart for the binary tree search for the containment search of point P.

facilitate the containment search for other entities. If, for instance, a surface grid is considered, instead of a field grid, the bounding boxes are constructed based on the quadrilateral faces of the surface grid. Moreover, the accurate check, that follows the bounding box containment check of a terminal leaf, must be altered. The rest of the procedure can remain exactly the same.

4.3 Evaluation of surface integrals

Overlap of grids is in general not restricted to the off-body part of the grid, which means that surface grids can also overlap. Occurrence of overlap on surface grids presents a number of challenges. One of these challenges is the accurate evaluation of surface integrals, required for instance to compute force coefficients. This problem is dealt with by application of so-called zipper grids [37]. A zipper grid consists of a single layer of triangles between two structured surface grids. First, the method used for the generation of these zipper grids is discussed, subsequently, the implementation is verified. Finally, the approach taken to use a zipper grid in the evaluation of a surface integral is discussed.

Requirements

Before the zipper grid generation method is explained, the requirements for the overlapping surface grids are stated. Although, in principle, there are no restrictions for the applicability of the zipper grid generation method, some restrictions have been imposed in the current implementation. These restrictions have been imposed to limit the amount of code to be implemented.

The current implementation of the zipper grid generation method requires the number of different surface grids in one region of overlap to be no more than two. In this way, only two surface grids need to be zipped together in a single region of overlap. The second requirement is that the resulting front of the surface grid forms a closed loop when overlapping faces have been removed from the surface grid.

For the current application, these requirements are easily met. However, to make the method more generally applicable these restrictions need to be alleviated. Suggestions on how this can be done have been presented by Chan [37].

4.3.1 Zipper grid generation method

Surface grid overlap identification

The first step in the generation of the zipper grid is the identification of the surface grids that require zipping. Zipping is only required for surface grids that represent a solid wall boundary, since this type of surface is typically used in the surface integrals to compute the force coefficients. This information is available, because the flow solution method also requires knowledge of surfaces that represent solid walls, to impose the appropriate boundary conditions. An alternating binary tree is created for each of these surfaces.

The alternating binary tree is first used to identify which of the corresponding surface grids are point matching. This information can be used later on to combine surface grids to satisfy the requirement of surface grids in the region of overlap to form a closed loop. Moreover, point matching surface grids do not need to be checked for overlap with

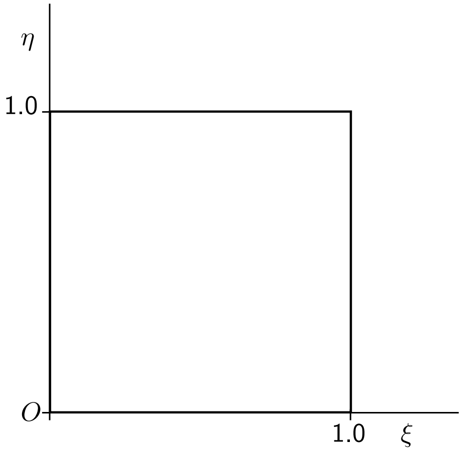


Figure 4.5: Quadrilateral cell in generalized coordinate system.

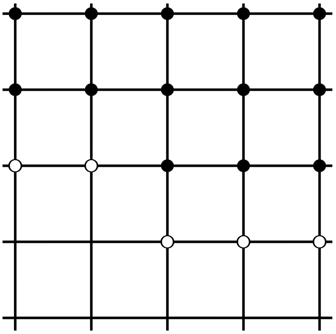


Figure 4.6: Schematic representation of a part of a partially blanked surface grid. The solid dots represent non-blanked vertices. The open circles represent vertices that have been identified as boundary vertices.

each other. Subsequently, non-point matching surface grids are checked for overlap by considering overlap of the bounding boxes of the root node of the binary tree. When these do not overlap, the corresponding surface grids will not overlap either. However, if root node bounding boxes are found to overlap, a tree search is performed. This search consists of a containment search of the vertices of one of the surface grids. Faces corresponding to a terminal leaf for which the bounding box contains the vertex are subjected to additional checks. First, the surface unit normal vector is checked for compatibility with the surface unit normal vector of the face corresponding to the boundary vertex, by considering the dot product of both vectors. A second check is performed if compatibility of both vectors is observed — i.e. if the result is larger than $(1 - \delta)$, for a small value of δ. In this second check, the minimum distance of the vertex to the face is determined, together with the coordinate of the vertex in the generalized coordinate system of the quadrilateral cell, see figure 4.5. If the coordinate satisfies

$$0 \leq (\xi, \eta) \leq 1$$

and if the minimum distance of the vertex to the face is smaller than a distance d, for which the value is based on the local grid dimensions, the vertex is qualified to be inside the face. If this condition is true for at least a small number of vertices, both surface grids are qualified to overlap with each other.

Closed loops

A single surface grid does not always form a closed loop by itself. For a closed loop to exist, the surface grid needs to have at least two vertices that share the same physical location. Moreover, a loop is only closed if the equation

$$\oint \left[\boldsymbol{\nabla}\phi\left(\boldsymbol{x}\right)\right] \cdot \mathrm{d}\boldsymbol{l} = 0 \tag{4.1}$$

is satisfied for an arbitrary path along the grid lines from one vertex to the other vertex sharing its coordinate. In this equation $\phi(\boldsymbol{x}) : \mathbb{R}^3 \to \mathbb{R}$ is an arbitrary scalar field that is at least once differentiable in each coordinate direction. Therefore, if equation (4.1) is not satisfied or if no matching vertices exist in a surface grid, the surface grid does not form a closed loop and needs to be combined with other point matching surface grids to do so. Note, that the point matching information was already acquired in the procedure to find overlapping surface grids and can therefore be reused here. In the process of combining two surface grids, the relative orientation of each grid is determined and then the grids are combined. Subsequently, the line integral of equation (4.1) is evaluated for the newly created surface grid. When the result satisfies the criterion for a closed loop, the surface can be used for the generation of a zipper grid; otherwise, the procedure is repeated until it is true for all surface grids that need to be considered in the zipper grid generation.

Zipping procedure

Consider two surface grids, grid S_{A} and grid S_{B}, that have been identified to require zipping and for which the zipper grid requirements have been met. The first step in the generation of the zipper grid is the identification of the faces of the surface grids for which the actual overlap exists. This identification is done in the same way as for the identification of the actual overlap. Faces in the region of overlap are removed from the surface grid by blanking out all vertices of the face for which at least one vertex has been found to be inside a face of the other surface grid. This process starts with the surface grid that has the least number of vertices in the region of overlap, because this grid is in general the coarsest of the two. If overlap still remains after the faces have been removed from the coarser grid, the same procedure is repeated for the finer grid.

Now that no overlap remains, the resulting gap must be closed. For that purpose, the vertices that are on the boundary of the surface grid in the region where the overlap existed need to be identified. This identification is done based on two criteria: (i) the vertex is blanked out; (ii) the vertex neighbours a vertex that is not blanked out. Vertices satisfying these criteria are collected and used in the generation of the zipper grid. Figure 4.6 shows an example of an already partly blanked surface grid for which the previously stated criteria have been employed to identify boundary vertices, indicated by the open circles. However, for the surface grid with the highest density of vertices in the region of overlap, which is therefore blanked after the first grid has been blanked, it often occurs that the whole boundary, or at least part of it, is not blanked at all. This situation exists, because blanking of the first grid already results in no remaining overlap between both grids. Hence, no faces have been removed from the second grid and therefore no blanked vertices exist. This plight is dealt with as follows. Since by requirement the surface forms a closed loop, the vertices of two of the boundaries of the surface grid must match. First, it is determined for which two boundaries matching occurs; this identification is done for each surface grid. For the two remaining boundaries, the average minimal distance to the boundary to which the grid is to be zipped is computed. The boundary for which this result is the smallest is the boundary that is closest and hence the boundary that will be zipped to the already blanked boundary of the other surface grid. Therefore, vertices belonging to this boundary are blanked and subsequently identified as boundary vertices for the zipper grid procedure.

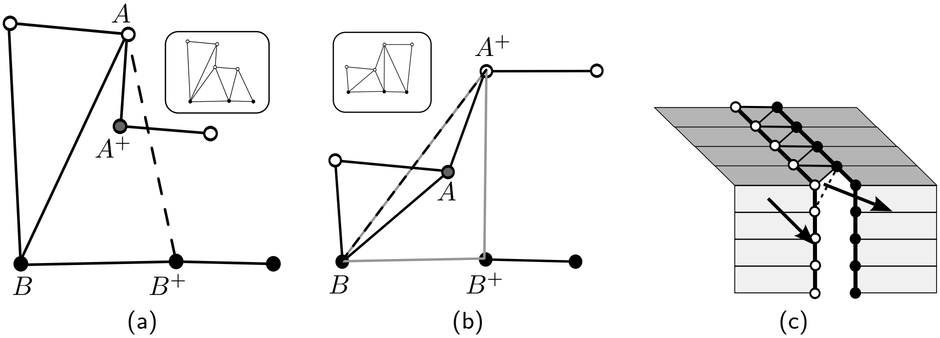


Figure 4.7: Examples of triangles that must not be created when a zipper grid is made. The boundary with the open circles, containing vertex A and A^+, must be zipped to the boundary with the solid dots, containing vertex B and B^+. The last valid edge created connects vertex A with vertex B. The proposed next edge is indicated by the dashed line. In subfigure (a), the resulting triangle encloses vertex A^+ and is therefore invalid. In subfigure (b), creating the proposed edge always results in the triangle created by placing a next edge to enclose vertex A and is therefore also invalid. Subfigure (c) shows a situation where the surface normal of the triangle, created by placing the proposed edge, is incompatible with the surface normal of the face with which the triangle shares an edge. The insets in subfigure (a) and (b) show the proper way of placing the edges, to create the zipper grid.

The next step in the zipping procedure is to create the triangles. First, it is investigated if valid triangles can be created between vertices of one boundary. In this case a triangle is considered valid if it does not enclose a vertex of either boundary. After that, triangles are created between the vertices of both boundaries. For the generation of these triangles, there are three criteria that need to be satisfied:

- a triangle must not enclose a vertex of one of the boundaries;
- the counterpart of a triangle must not enclose a vertex of either boundary, example of such a situation is shown in subfigure 4.7 (b);
- the surface normal vector should be compatible with that of the original geometry.

Situations for which placing an edge results in the construction of an invalid triangle are illustrated in figure 4.7. Furthermore, there is an additional requirement for the triangles to ensure a better quality of the zipper grid. If two possible triangles can be created, the triangle that realizes the most orthogonal propagation front is the preferred choice. An example of this situation is given in figure 4.8. In this case both triangles are valid based on the requirements stated above. However, the triangle on the right is preferred over the left one, because of the orthogonality requirement. This procedure of generating triangles is repeated, until all vertices of both boundaries are part of a triangle. When this condition is true, the complete gap between both surface grids is filled with triangles. The final step in the generation of the zipper grid is to order the vertices of each triangle in such a way that the direction of the surface normal vector is compatible with the surface normal vector of both surface grids that it connects. For this purpose,

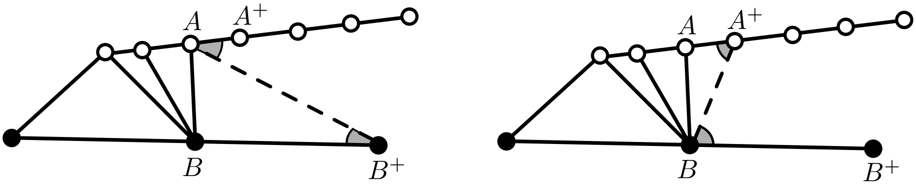


Figure 4.8: Both triangles resulting from creating the edge indicated by the dashed line are valid. The option on the right is preferred, because it gives the most orthogonal propagation front.

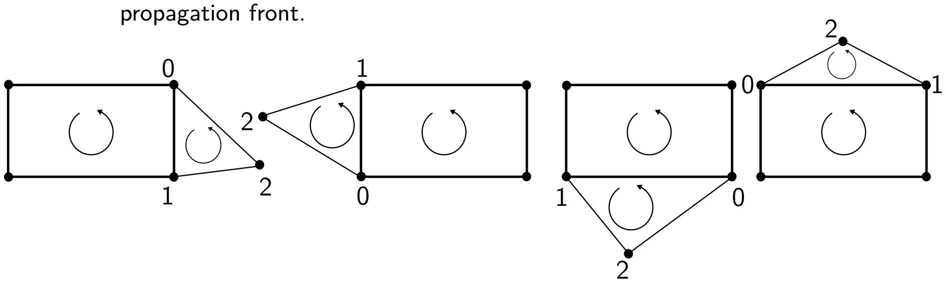


Figure 4.9: Ordering of the vertices of a triangle, depending on the edge it is connected to, to realize a compatible surface normal vector. The counter-clockwise directed arrow indicates a vector pointing outward of the paper.

the outward pointing surface unit normal vector of a triangle is defined as

$$\boldsymbol{n} := \frac{\boldsymbol{v}_{01} \times \boldsymbol{v}_{02}}{||\boldsymbol{v}_{01} \times \boldsymbol{v}_{02}||}, \tag{4.2}$$

where $\boldsymbol{v}_{01}$ is the vector pointing from vertex 0 to vertex 1 of the triangle, vector $\boldsymbol{v}_{02}$ is defined in an analogous way, see figure 4.9. The order of the vertices of the triangle that satisfies the requirement of a compatible surface normal is determined using information about the order of the vertices of the quadrilateral the triangle is connected to. In figure 4.9 the four possible ways to connect a triangle to a quadrilateral are depicted. For this quadrilateral the surface normal is pointing outward of the paper, indicated by the arrow directed counter-clockwise. The algorithm for the vertex ordering of the triangle determines to what side of the quadrilateral the triangle is connected and chooses the vertex order of the triangle for the corresponding situation. In this way, there is no need to perform floating-point operations, making the procedure more robust.

It should be noted that the procedure described above does not introduce new vertices. The method only uses existing vertices and connects them to create the zipper grid.

4.3.2 Verification

The integral of the outward pointing surface unit normal vector over a closed surface is equal to the null vector. This property of a closed surface is used to verify that the zipper grid method provides a closed surface for a configuration that originally has overlapping surface grids. Since the surface grid provides a discrete representation of the surface, the surface integral reduces to a summation over the faces used for the discretization. For a configuration consisting of surface grids S_A and S_B — for which the non-blanked quadrilateral faces are collected in the sets $\mathbb{Q}_\mathrm{A}$ and $\mathbb{Q}_\mathrm{B}$ — that are connected by a zipper

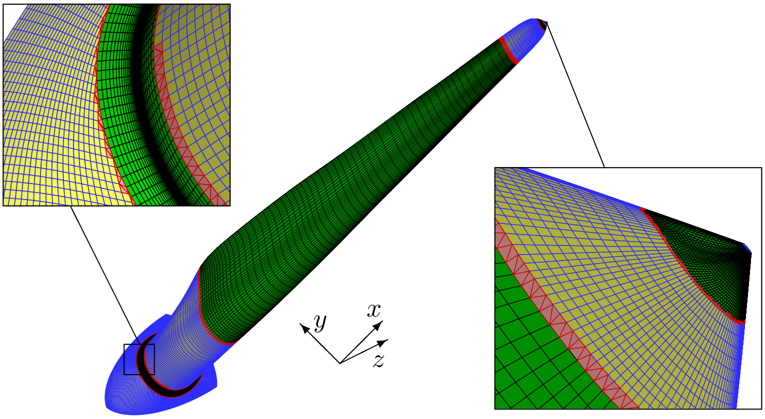


Figure 4.10: Surface grid on one blade and part of nose cone, including zipper grids to connect the different surface grids. Insets show details of zipper grid in region near tip and around collar grid.

grid for which the triangles are collected in set $\mathbb{T}$, this summation is expressed as

$$\sum_{F \in \mathbb{A}} \left(\boldsymbol{n}_F A_F \right), \tag{4.3}$$

with $\mathbb{A} = \mathbb{Q}_\mathrm{A} \cup \mathbb{Q}_\mathrm{B} \cup \mathbb{T}$, $\boldsymbol{n}_F$ the outward pointing surface unit normal vector of face F and A_F its corresponding surface area.

The configuration used to verify the zipper grid generation method consists of a three-bladed wind turbine rotor and a nose cone, with a rotor diameter of approximately 88 $[\mathrm{m}]$. To completely close the geometry, the open part of the nose cone, downstream of the rotor, is closed using triangles. These triangles are constructed by connecting two successive vertices on the edge of the cone with a common vertex at the axis of rotation of the rotor. The surface geometry of a single rotor blade is discretized using 8 different overlapping body-fitted surface grids. The nose cone is discretized using three point-matching body-fitted surface grids. A collar grid is used in the region where a blade intersects with the nose cone. The resulting zipper grid for part of this configuration is presented in figure 4.10. The result of the summation of equation (4.3) for the whole configuration is depicted in table 4.1. Considering the small values of this result and the finite-precision of the floating-point arithmetic used for the evaluation of the summation, the surface grid configuration is considered closed, indicating that the zipper grid method has been successful in creating a closed non-overlapping surface grid.

4.3.3 Use of zipper grid

Once a zipper grid has been generated, it can be used for the accurate evaluation of surface integrals, for instance to compute a power coefficient. However, since the triangles of the zipper grid have no explicit connection with the flow domain — which is the case for the quadrilateral faces of the surface grids — a flow solution must first be obtained for the triangles, in order to be able to evaluate the surface integral. The flow solution

Table 4.1: Result of the evaluation of equation (4.3) for the configuration of a surface grid for which a part is displayed in figure 4.10.

direction	value $\left[\mathrm{m}^2\right]$
x	$2.01 \cdot 10^{-15}$
y	$1.94 \cdot 10^{-13}$
z	$-1.36 \cdot 10^{-13}$

for the quadrilateral cells neighbouring a triangle is used for this purpose. The approach taken is as follows.

First, two faces are selected from each surface grid neighbouring the zipper grid. Selection is based on the smallest distance of the centroid of the face to the centroid of the triangle. The centroids of the quadrilateral faces are connected, to create a new quadrilateral face, see figure 4.11. Again, the transformation to a generalized coordinate system, this time for this new quadrilateral face, is used to determine the bi-linear coordinate of the centroid of the triangle. If this bi-linear coordinate indicates that the centroid of the triangle is inside the newly created quadrilateral, the result is used to determine the weighting factors for the interpolation of the flow solution from the quadrilateral faces of the surface grids. If, on the other hand, the centroid of the triangle is outside the quadrilateral, a different face of the surface grid is selected, based on the bi-linear coordinate found. This process is repeated, if required, until a quadrilateral can be constructed which encloses the centroid of the triangle.

When the interpolation weights have been determined for all triangles of every zipper grid involved in the evaluation of the surface integral, an accurate result can be computed. This result is obtained by neglecting the contribution of the quadrilateral faces of the surface grids which have been blanked for the construction of the zipper grid. Moreover, the contribution of the zipper grid to the surface integral is computed by means of interpolation of the flow solution, using the corresponding weights.

Note, that for the construction of a zipper grid that is involved in a surface integral, the blanking of the field grid must be taken into account when blanking the surface grid. Satisfying this requirement makes sure that no faces neighbouring a hole cell — for which the flow solution is not updated — are used for the interpolation of the flow solution of a triangle in the zipper grid.

4.4 Elimination of cells outside physical flow domain

Overlap occurs between different field grids, but it can also happen that part of a grid resides outside the physical flow domain and hence does not necessarily overlap other grids. This situation occurs when for example an off-body grid shares part of the domain with a geometrical entity, such as a wing or a wind turbine blade. A 2D example of such a situation is presented in figure 4.12. Cells that are inside this geometrical entity do not contribute to the flow solution and must therefore be removed from the active computational grid. Different methods exist to achieve this goal. Some examples are:

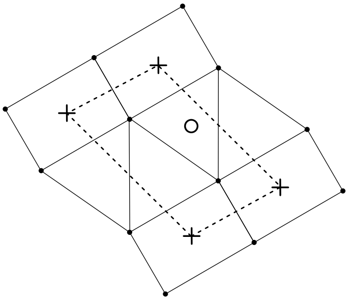


Figure 4.11: Schematic representation of a part of a zipper grid. The centroid of the triangle, depicted by a circle inside the dotted quadrilateral cell used for interpolation of the flow solution, required for the evaluation of the surface integral.

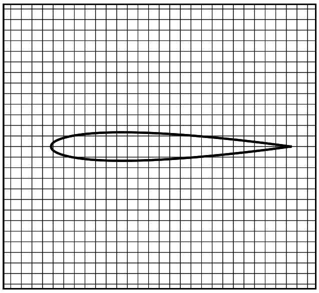


Figure 4.12: Aerofoil geometry and a uniform background grid. A number of cells from the background grid reside (partly) inside the aerofoil geometry.

the object x-ray technique [121], a Cartesian hole map [41] or a ray-casting method. The latter method is used in the present research and its concept and implementation are explained in the next subsection.

4.4.1 Ray-casting method

Concept

The concept of the ray-casting method is very simple. A ray is being cast from a certain location in an arbitrary direction and the number of physical surfaces crossed by the ray is counted. Depending on the result being odd or even, a qualification of the point the ray was cast from can be made regarding the status of that point. When the number of crossings found is odd, the point must be inside a geometrical entity and is therefore outside the physical flow domain. For an even number of crossings, the opposite is true.

Approach

Since the direction of the ray being cast is arbitrary, a convenient choice is to choose the casting direction collinear with one of the coordinate directions. Then, a single binary tree is constructed containing all non-blanked surface elements. In the case of surface grid overlap, a zipper grid must be generated first. In such an event, the tree also includes the triangular faces of the zipper grid.

Next, a ray is cast from the centre of mass of a cell — assuming a cell-centred discretization being used — in the casting direction chosen, for all cells used for the discretization of the flow domain. For each ray cast, a tree search is performed, looking for bounding boxes crossed by the line collinear with the ray. For bounding boxes found to be crossed by this line it is verified if they are also crossed by the actual ray, taking into account the origin and the direction of the ray. When this result is true, the actual face corresponding to the bounding box is checked for being crossed by the ray. The method used for this verification is treated in the next subsection. If the face is found to be crossed by the ray, it is determined if the ray enters or leaves the flow domain, considering the

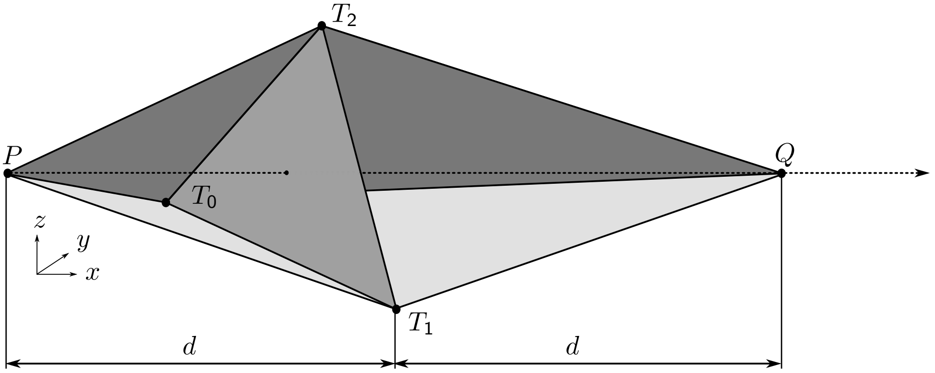


Figure 4.13: Ray, indicated by dashed line, is cast from P in positive x-direction. Point Q is placed at distance $2d$ from P in the casting direction. To investigate triangle $T_0T_1T_2$ being crossed, the signed-volume is computed for tetrahedron $T_0T_1T_2P$ and $T_0T_1T_2Q$. The triangle is crossed by the ray if both values have the opposite sign.

surface normal vector of the face. This analysis is done to identify situations for which a miscount of the number of crossings occurs, making the whole procedure more robust.

Note, that in this case, the tree search is not terminated after one match has been found, since more than one face of a single surface grid can be crossed by a ray.

Verification of face being crossed by ray

A bounding box found to be crossed by the ray can either correspond to a quadrilateral face or a triangular face. In the case a quadrilateral face is encountered, the face is split along the diagonal, such that two triangular faces arise. Then, for a triangular face, the following checks are performed. When the origin of the ray does not correspond with either of the vertices of the triangle, a signed-volume check [3] is performed. This signed-volume check is used to verify if the ray really crosses the face and is not just collinear with it.

The signed-volume check comprises of the computation of the signed-volume of two tetrahedral cells. The first tetrahedron is constructed by connecting each of the vertices of the triangle with the origin of the ray. For the construction of the second tetrahedron, a suitable end point of the ray must be selected. This end point is placed at a distance of two times the distance d from the origin of the ray in casting direction; where d is the distance in casting direction from the origin of the ray to the farthest vertex of the triangle, see figure 4.13. Subsequently, the signed-volume is computed. For a tetrahedron specified by the coordinates $\boldsymbol{v}_0$, $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ the signed-volume $\mathcal{V} : \mathbb{R}^3 \to \mathbb{R}$ is found by evaluating

$$\mathcal{V}\left(\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\right) = \frac{1}{3!} \det \begin{pmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{pmatrix}. \tag{4.4}$$

If point P with coordinate $\boldsymbol{p}$ and point Q with coordinate $\boldsymbol{q}$ lie on the opposite side of triangle T with its vertices at coordinates $\boldsymbol{t}_0$, $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$, the signed-volumes $\mathcal{V}\left(\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{p}\right)$

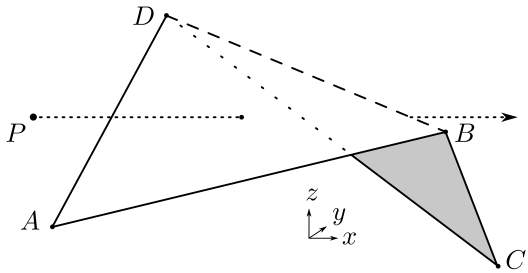


Figure 4.14: Schematic representation of a non-planar surface element $ABCD$, split in two triangles ABD and BCD, which are both crossed by the ray originating from point P cast in the positive x-direction.

and $\mathcal{V}(\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{q})$ have the opposite sign. In this situation the triangle being crossed by the ray is verified by considering the barycentric coordinate [124] of the origin of the ray with respect to the triangle, both projected on the plane perpendicular to the casting direction. For the purpose of limiting problems with finite-precision arithmetic, the origin of the coordinate system is chosen to coincide with one of the projected vertices of the triangle.

Any coordinate $\boldsymbol{p}$ within a triangle can be expressed as the linear combination of the coordinates of its vertices. For a triangle with vertex 2 at the origin of the coordinate system, this linear combination is expressed as

$$\boldsymbol{p} = \sum_{\mathrm{i}=0}^{\mathrm{i}=1} \left(\bar{\lambda}_{\mathrm{i}} \boldsymbol{r}_{\mathrm{i}}\right), \tag{4.5}$$

with weights $\bar{\lambda}_{\mathrm{i}}$ subject to the constraint

$$\sum_{\mathrm{i}=0}^{\mathrm{i}=2} \left(\bar{\lambda}_{\mathrm{i}}\right) = 1.$$

The weights are found by solving equation (4.5), using this constraint. For a coordinate within the triangle, the solution satisfies

$$0 \leq \bar{\lambda}_{\mathrm{i}} \leq 1 \quad \forall \mathrm{i} \in \{0, 1, 2\}.$$

This method is used to check if the projection of P is located in the triangle resulting from the projection of triangle T on the plane. If it is, then the triangle considered is crossed by the ray. When the bounding box corresponds to a quadrilateral face, crossing of the second triangle — originating from splitting the quadrilateral face — is verified using the same procedure. This action is also performed when the result for the first triangle indicates that it is crossed by the ray, because it can happen that a ray enters the quadrilateral via the first triangle and exits again through the second one — or vice versa — as shown in figure 4.14. This situation arises, when a non-planer surface is discretized and the resulting face is slightly twisted. Not checking both triangles would therefore result in an incorrect qualification of the status of the cell the ray was cast from.

4.5 Domain connectivity

After the ray-casting procedure has been performed, to get rid of cells outside the physical flow domain, the block connectivity needs to be determined. The method used for this purpose is the so-called implicit hole cutting method [107]. The advantage of this method compared to most of the other methods devised for determining the domain connectivity, is that this method does not require the manual specification of hole cutting surfaces. Not having this requirement is a significant advantage, considering the use in the optimization method; because user intervention must be absent, in order to realize a feasible optimization procedure.

4.5.1 Concept

The first step in the implicit hole cutting is the identification of overlapping cells between the different blocks of the field grid. For a cell-centred approach the containment of the centre of mass is employed to qualify the existence of overlap. For a vertex-centred approach, the vertex itself is the obvious choice. In regions of overlap, a choice is made regarding which cells are used for solving the governing equations and which cells are used to transfer the dependent variables from one block to another. This choice is based on three criteria, in order of importance:

(i) user defined block priority;

(ii) index distance to the wall;

(iii) cell volume.

The first criterion is only used in cases for which the other two criteria result in ambiguous results and therefore acts as a means to manually control the hole cutting procedure. For that reason, it is the first criterion to be considered. Cells of a block with high priority are used as field cells while a low block priority results in cells to become potential hole cells. If no block priority has been specified, or if the priority is the same for both blocks, the second criterion is considered.

This criterion considers the index distance to the wall, i.e. the number of cells a cell is away from a block boundary for which a wall boundary condition is imposed. A cell with a lower index distance to a wall is the preferred choice to act as a field cell over a cell with a higher index distance, which becomes a potential hole cell. For this criterion, a value can be specified to indicate what the maximum index distance is for which this criterion is relevant; typical values range from 5 to 10. When the first two criteria do not give a conclusive result, the third and final criterion is used.

The cell volume criterion considers the volume of the cells that overlap with each other. The cell with the smallest volume is used as field cell, while the other overlapping cells become potential hole cells.

After the potential hole cells have been identified, a choice must be made regarding which cells are used as fringe cells, to transfer the dependent variable information. Fringe cells are chosen from the cells that have been labelled as potential hole cell in such a way that for all field cells the stencil used for solving the governing equations can be satisfied. Therefore, cells that are near a block boundary for which no explicit boundary conditions are imposed — i.e. a so-called overset-outer-boundary — are automatically fringe cells.

Unless there is a cell farther away from that particular boundary, that is already used as a fringe cell to satisfy the stencil of the closest field cell. In that case the cell near the boundary becomes a hole cell.

The donor cell corresponding to a fringe cell is one of the cells that was found to overlap with the current fringe cell. Determining the corresponding interpolation stencil is treated in subsections 4.5.3 and 4.5.4.

4.5.2 In cell qualification

In section 4.2.2 the binary tree search procedure, used to find cells for which the Cartesian bounding box encloses a certain point, was explained. This method is used to identify the overlap between cells. Note, that for a vertex-centred scheme, the bounding boxes of the actual cells are used, while for a cell-centred approach, bounding boxes are constructed for the cells of the dual mesh instead. Once a bounding box has been found, a more accurate check must be performed to identify if the cell corresponding to the bounding box actually encloses the cell as well. This method is explained next.

The containment of a point P with coordinate $\boldsymbol{p}$ by a cell defined by vertices with coordinates $\boldsymbol{r}_{\mathrm{n}}$, $\mathrm{n} \in \mathbb{N}_0$, $\mathrm{n} < 8$ is divided into two steps. The first step considers the containment of the point by a different bounding box. However, this second bounding box has in general a different orientation than the Cartesian bounding box used in the binary tree. The new bounding box with the alternative orientation has been introduced to increase the robustness of the method. The accurate containment check requires the solution of a set of non-linear equations. A Newton algorithm is employed for this purpose. However, for high-aspect ratio cells, that are not very well aligned with the Cartesian axes directions, a point within the Cartesian bounding box is not necessarily very close or inside the cell itself. Applying the Newton algorithm in such a case can result in divergence of the solution procedure. Therefore, the alternatively oriented bounding box acts as a means to filter out points that are within the Cartesian bounding box, but which could present difficulties for the solution algorithm used for the accurate check.

The new bounding box is constructed in a transformed coordinate system. The corresponding coordinate transformation is determined as follows. The first coordinate axis of the transformed coordinate system, $\tilde{x}$, is chosen to be parallel to the axis of the hexahedron for which the dimension is the largest, i.e. the length of the vector connecting two opposite surfaces, see figure 4.15. The second axis is perpendicular to this axis in the direction of the vector which is second in length. For this purpose, the projection of the corresponding vector onto the normalized first coordinate direction is subtracted from the original vector. The direction of the third axis is then determined by taking the cross product of the vectors that define the first two axes. A Cartesian bounding box is constructed in the new $\tilde{x}\tilde{y}\tilde{z}$-coordinate system and the containment of P by the bounding box is determined.

If the search point is also enclosed by this second bounding box, a third check is performed, which establishes if the search point is actually in the cell itself. For this purpose, the tri-linear transformation $\mathbb{R}^3 \to \mathbb{R}^3$ is used of a unit cube to a general hexahedron with one of its corners at the origin, see figure 4.16. This transformation

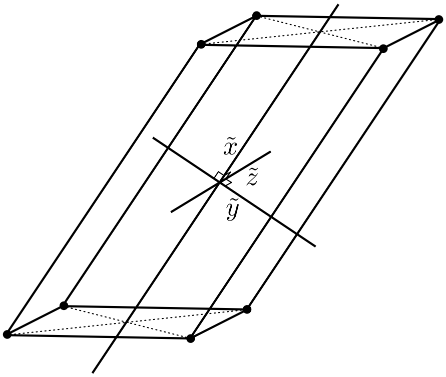


Figure 4.15: Definition of the new coordinate system based on shape and orientation of a hexahedral cell. The coordinate axes used to determine the second bounding box for the in-cell-qualification of a point.

reads

$$\boldsymbol{r}'(\boldsymbol{\xi}) = \frac{1}{8}\sum_{\mathrm{k}=0}^{\mathrm{k}=1}\left(\sum_{\mathrm{j}=0}^{\mathrm{j}=1}\left[\sum_{\mathrm{i}=0}^{\mathrm{i}=1}\left(\left[(-1)^{\mathrm{i}}(1-2\xi)+1\right]\left[(-1)^{\mathrm{j}}(1-2\eta)+1\right]\right.\right.\right.$$
$$\left.\left.\left.\left[(-1)^{\mathrm{k}}(1-2\zeta)+1\right]\boldsymbol{r}'_{\mathrm{n}}\right)\right]\right), \quad (4.6)$$

with index $\mathrm{n} = 4\mathrm{k} + 2\mathrm{j} + \mathrm{i}$, $\boldsymbol{r}'_{\mathrm{n}}$ the coordinate of the corners of the hexahedron and $\boldsymbol{\xi} \equiv (\xi, \eta, \zeta)^T$. Note, that in general an arbitrary hexahedron representing a cell of the grid will not have one of its vertices coincide with the origin. This requirement can however, simply be met, by performing a translation of the cell, such that vertex 0, with original coordinate $\boldsymbol{r}_0$, is at the origin; therefore $\boldsymbol{r}'_{\mathrm{n}} \equiv \boldsymbol{r}_{\mathrm{n}} - \boldsymbol{r}_0$. The purpose of first performing a translation, before the transformation is performed, is to limit the effect of using finite-precision arithmetic for performing the computations. Subsequently, containment of P is determined by solving

$$\boldsymbol{r}'(\boldsymbol{\xi}) - (\boldsymbol{p} - \boldsymbol{r}_0) = \boldsymbol{0}, \quad (4.7)$$

for $\boldsymbol{\xi}$. Based on the solution of equation (4.7), it can be determined if P is inside the hexahedron, i.e. when

$$0 \leq (\xi, \eta, \zeta) \leq 1.$$

Otherwise, the point is considered to be outside the cell.

4.5.3 Interpolation coefficients

When a cell is qualified as fringe cell and the corresponding donor cell has been identified — based on the procedure explained in subsection 4.5.1 — the interpolation coefficients must be determined. For this purpose, the result of solving equation (4.7) for $\boldsymbol{\xi}$ is

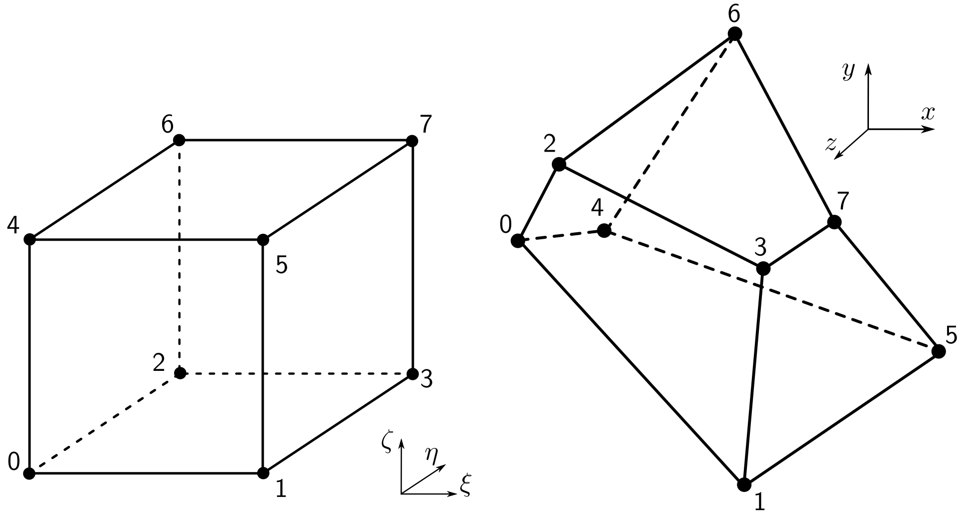


Figure 4.16: Schematic representation of the tri-linear transformation of a unit cube on the left to an arbitrary general hexahedral cell on the right.

used. The weights $\boldsymbol{\omega}$ are then given by

$$\boldsymbol{\omega} := \begin{pmatrix} (1-\xi)(1-\eta)(1-\zeta) \\ \xi \quad (1-\eta)(1-\zeta) \\ (1-\xi) \quad \eta \quad (1-\zeta) \\ \xi \quad \eta \quad (1-\zeta) \\ (1-\xi)(1-\eta) \quad \zeta \\ \xi \quad (1-\eta) \quad \zeta \\ (1-\xi) \quad \eta \quad \zeta \\ \xi \quad \eta \quad \zeta \end{pmatrix}, \tag{4.8}$$

where the i^{th} component of this vector corresponds to vertex v_{i} of the donor cell; the order of the vertices according to figure 4.16. These weights are used to determine the value of the dependent variables for the fringe cell, based on the flow solution corresponding to the vertices of the donor cell. Note, that for a cell-centred discretization, these vertices of the dual mesh correspond to the cell centres of the primal mesh. Therefore, it can happen with the cell-centred approach that one or more of the donors of a fringe cell is a fringe cell itself. For the purpose of preventing inconsistencies in the interpolation, which can result in, for instance, hampering convergence of the flow solution method, the interpolation stencil of a fringe cell must be an explicit expression in terms of field cells only. The approach taken to achieve this purpose is presented in the next subsection.

4.5.4 Explicit expression in terms of field cells

To obtain an interpolation stencil for a fringe cell, that is an explicit expression in terms of field cells only, first fringe cells are considered for which the flow solution can be computed using field cell information only. When such a fringe cell, referred to as cell A, also provides donor information for a different fringe cell, referred to as cell B, the donor information of cell A can be transferred to cell B by multiplying the weight corresponding to cell A by the weight of each of the donors of cell A. With this procedure, the total number of donors of cell B is increased with the total number of donors of cell A minus one[7] and cell A is effectively eliminated as donor of cell B. When cell B is also a donor of a different cell, then it can be eliminated as donor as well, provided that cell A was the only non-field donor of cell B. This procedure is repeated until no fringe cells remain for which an explicit expression in terms of field cells can be found by means of this replacing procedure.

After that, it is still possible that fringe cells exist for which one or more fringe cells provide donor information. These fringe cells — from here on referred to as *incomplete fringe cells* — are collected and a system of linear equations is constructed. This system of linear equations resembles the linear dependency of incomplete fringe cells on other incomplete fringe cells and field cells. The corresponding matrix has a sparse structure with a unit diagonal. All non-zero off-diagonal terms are negative and have a magnitude equal to the weight corresponding to the incomplete fringe cell on which the incomplete fringe cell depends.

To determine the right-hand side of the system of linear equations, first all field cells must be identified that act as donor for at least one of the incomplete fringe cells. Next, the right-hand side is constructed by selecting one of the field cells. The weight corresponding to the selected field cell for an incomplete fringe cell that is explicitly affected by that field cell is assigned to the row entry of the right-hand side that corresponds to the incomplete fringe cell. Therefore, the row entries of the right-hand side corresponding to all incomplete fringe cells that are explicitly affected by this field cell have been assigned a value. The remaining entries are zero. Subsequently, the LU decomposition of the matrix is determined and stored. Note, that determining and storing the LU decomposition of the matrix is feasible because of the presence of a fairly limited number of incomplete fringe cells — at least for the cases considered so far. The solution is then obtained by subsequently performing a forward and a backward substitution for the lower and upper triangular matrix, respectively. This solution provides the value for the weight corresponding to the field cell that was selected, for all incomplete fringe cells. All non-zero values of the result, exceeding a certain threshold value, are stored, together with the incomplete fringe cell to which it corresponds and the field cell that was selected.

This procedure is repeated for all field cells that have been identified to explicitly affect at least one incomplete fringe cell. Since only the right-hand side changes, the previously stored LU decomposition of the matrix can be reused for obtaining the subsequent solutions of the system of linear equations. Once all field cells have been handled, the explicit dependency of the incomplete fringe cells on only field cells is determined.

To limit the size of the interpolation stencil, it is possible to specify a maximum number of donors. If the number of donors of a fringe cell exceeds this maximum, additional donors are ignored, in descending order of the corresponding weight value. To make sure that the combined weights sum to one, an additional normalization is required if the specified maximum number is exceeded, or if donor weights have been encountered that did not exceed the threshold value. Normalization is, however, performed by default, because round-off errors, caused by using finite-precision arithmetic, can also result in the sum of the weights not being equal to one.

To illustrate the whole procedure a flow chart is provided in figure 4.17. Furthermore, consider the following example, in which the dependency of incomplete fringe cells F0

---

[7] Assuming none of the donors of cell A was already a donor of cell B

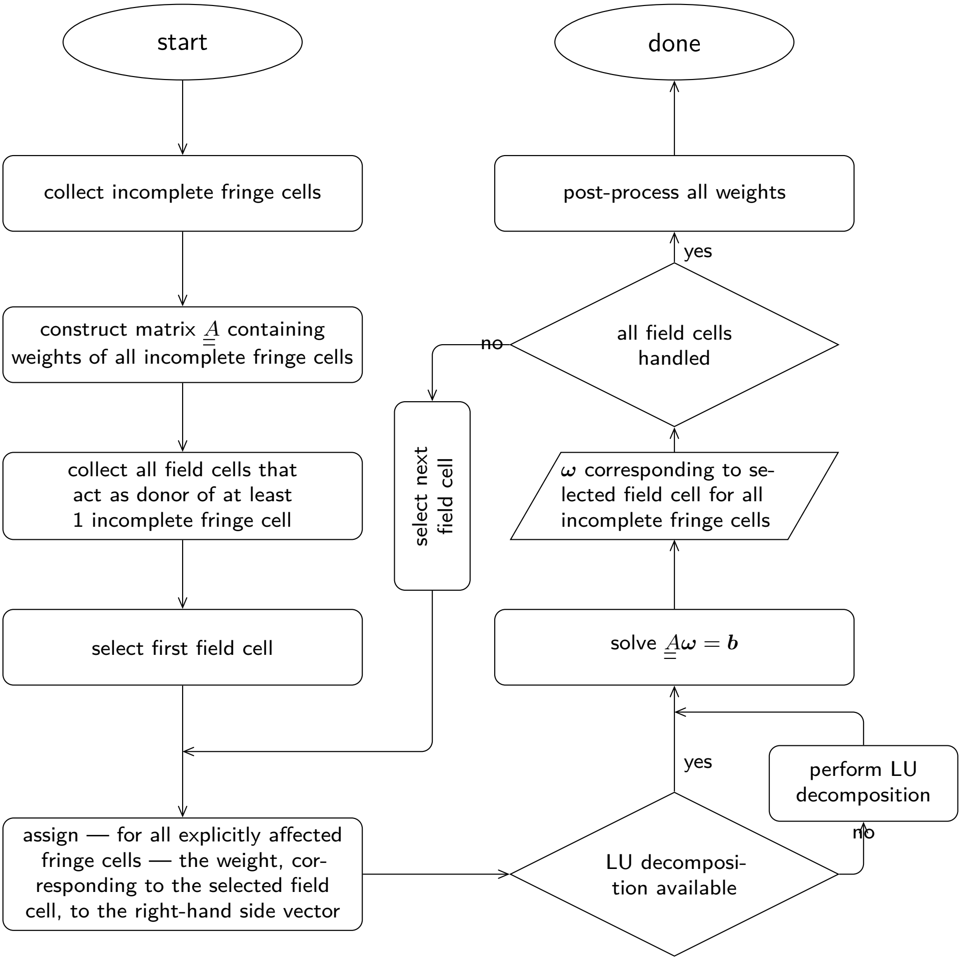

start
collect incomplete fringe cells
construct matrix $\underline{\underline{A}}$ containing weights of all incomplete fringe cells
collect all field cells that act as donor of at least 1 incomplete fringe cell
select first field cell
assign — for all explicitly affected fringe cells — the weight, corresponding to the selected field cell, to the right-hand side vector
LU decomposition available
yes
no
perform LU decomposition
solve $\underline{\underline{A}}\boldsymbol{\omega} = \boldsymbol{b}$
$\boldsymbol{\omega}$ corresponding to selected field cell for all incomplete fringe cells
all field cells handled
no
select next field cell
yes
post-process all weights
done

Figure 4.17: Flow chart of the procedure used to determine an interpolation stencil and the corresponding weights for the incomplete fringe cells, containing only field cells.

to F3 is given by

$$\begin{aligned}
\phi_{\mathsf{F0}} &= \omega_0\phi_{\mathsf{F1}} + \omega_1\phi_{\mathsf{F2}} + \omega_2\phi_{\mathsf{F3}} + \omega_3\phi_{\mathrm{a}},\\
\phi_{\mathsf{F1}} &= \omega_4\phi_{\mathsf{F0}} + \omega_5\phi_{\mathsf{F2}} + \omega_6\phi_{\mathrm{a}} + \omega_7\phi_{\mathrm{b}},\\
\phi_{\mathsf{F2}} &= \omega_8\phi_{\mathsf{F1}} + \omega_9\phi_{\mathsf{F3}} + \omega_{10}\phi_{\mathrm{a}},\\
\phi_{\mathsf{F3}} &= \omega_{11}\phi_{\mathsf{F0}} + \omega_{12}\phi_{\mathrm{b}}.
\end{aligned}$$

In this equation ϕ resembles a conserved flow variable for which the interpolation must be applied and ϕ_{a} and ϕ_{b} correspond to two different field cells. Based on the given dependency, the resulting matrix obtained by means of the procedure described above reads

$$\underline{\underline{A}} \equiv \begin{pmatrix} 1 & -\omega_0 & -\omega_1 & -\omega_2 \\ -\omega_4 & 1 & -\omega_5 & 0 \\ 0 & -\omega_8 & 1 & -\omega_9 \\ -\omega_{11} & 0 & 0 & 1 \end{pmatrix}.$$

Because there are two different field cells involved, two different right-hand side vectors must be constructed, which read

$$\boldsymbol{r}_\mathrm{a} \equiv \begin{pmatrix} \omega_3 \\ \omega_6 \\ \omega_{10} \\ 0 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{r}_\mathrm{b} \equiv \begin{pmatrix} 0 \\ \omega_7 \\ 0 \\ \omega_{12} \end{pmatrix},$$

respectively. Solving the system of linear equations

$$\underline{\underline{A}}\boldsymbol{\omega} = \boldsymbol{b} \qquad \forall\, \boldsymbol{b} \in \{\boldsymbol{r}_\mathrm{a}, \boldsymbol{r}_\mathrm{b}\} \tag{4.9}$$

for $\boldsymbol{\omega}$, gives the dependency of the incomplete fringe cells F0 to F3 on field cells a and b. Note, that all elements of the resulting solution vector $\boldsymbol{\omega}$ still satisfy: $0 \leq \omega_\mathrm{i} \leq 1$.

4.6 Summary

To conclude this chapter, a brief summary is provided, highlighting the most important aspects discussed. The chapter started with an introduction on the history of composite overset grids and its application in computational fluid dynamics. The application of this domain connectivity method in modern research codes has also been treated. Moreover, the terminology used in overset grid technology has been presented. Subsequently, the search procedure, used to efficiently determine the block connectivity has been pointed out. Alternating binary trees are used for this purpose. Then, the method has been explained that is used to cope with overlapping surface grids, regarding the correct and accurate evaluation of surface integrals. The accurate evaluation is achieved employing so-called zipper grids. Zipper grids, consisting of a single layer of triangles, are created by removing quadrilateral surface cells in the region of overlap between different surface grids and closing the resulting void between the surface grids with triangles. These zipper grids are also employed to determine which cells reside outside the physical domain and can therefore be excluded from the computation of the flow solution. By casting a ray originating from a cell centre, in an arbitrary direction, and counting the number of times the ray crosses a surface grid, the inside/outside status of that cell can be determined. Finally, the block connectivity method was discussed. Block connectivity is determined using an implicit hole cutting technique. Three different criteria are used, to decide which cells are used for solving the discretized partial differential equations, governing the flow and which ones are obsolete or must be used to transfer the flow solution between overlapping blocks. For consistency of the interpolation, the interpolation stencil for fringe cells is determined, such that only field cell data is used in the interpolation.

5

Flow model and solution method

> *"Even if there is only one possible unified theory, it is just a set of rules and equations. What is it that breathes fire into the equations and makes a universe for them to describe?"*
>
> — Stephen W. Hawking (1942 – present), *A Brief History of Time*

The physical phenomenon of a fluid in motion can be described by a mathematical model, often composed of partial differential equations. The validity of the model for a certain flow regime depends on the assumptions made in constructing the model. This chapter presents the mathematical model used in the optimization method to model the flow. Subsequently, the spatial discretization of the governing equations is treated, along with the treatment of the boundary conditions that are enforced. Moreover, the method used to solve the discretized equations is discussed. At the end of the chapter, the current implementation is verified by applying the method to subcritical 2D flow around the NACA 0012 aerofoil and comparing the solution with results obtained with different numerical methods, that have previously been verified. Also the spatial order of convergence of the method is investigated. Moreover, for the transonic flow around an ONERA M6 wing, the numerical solution is compared with experimental results from the literature, to validate the method.

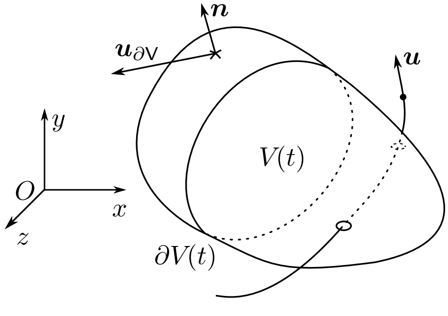


Figure 5.1: Schematic representation of an arbitrary moving control volume $V(t)$ with a moving permeable boundary $\partial V(t)$.

5.1 Governing equations

Starting with basic fundamentals of physics, i.e. the conservation laws, the equations of motion of a fluid can be derived for an arbitrary time-dependent control volume $V(t) \in \mathbb{R}^3$ with permeable boundary $\partial V(t) \in \mathbb{R}^3$, see figure 5.1. Under the assumption that: (i) the fluid is a continuum ; (ii) the fluid has a fixed composition; (iii) no relativistic effects occur [55], then the conservation of momentum is governed by the Navier-Stokes equations. Without an external volumetric force field present, these equations can be expressed in integral conservation form as

$$\frac{\partial}{\partial t} \int\limits_{V(t)} \rho \boldsymbol{u} \, \mathrm{d}V + \int\limits_{\partial V(t)} \rho \boldsymbol{u} \left(\left[\boldsymbol{u}^T - \boldsymbol{u}_{\partial \mathrm{V}}^T \right] \cdot \boldsymbol{n} \right) \mathrm{d}S = - \int\limits_{\partial V(t)} \left(p \boldsymbol{n} - \underline{\underline{\tau}} \cdot \boldsymbol{n} \right) \mathrm{d}S, \tag{5.1}$$

where ρ is the local density of the fluid, $\boldsymbol{u} \equiv (u, v, w)^T$ the local velocity of the fluid, p the static pressure, $\underline{\underline{\tau}}$ the stress tensor which is assumed to behave as a Newtonian fluid , $\boldsymbol{u}_{\partial \mathrm{V}}$ the local velocity of the permeable boundary surface of the control volume and $\boldsymbol{n} \equiv (n_{\mathrm{x}}, n_{\mathrm{y}}, n_{\mathrm{z}})^T$ the corresponding outward pointing unit normal vector. To gain insight in the relative importance of the different components of the Navier-Stokes equations, it is good practice to carry out a dimensional analysis of the equations. For that purpose, a number of scaling parameters are introduced: a characteristic length scale, $\mathcal{L}$, e.g. the local chord length of the wind turbine blade; a characteristic velocity, $\mathcal{U}$, e.g. the magnitude of the relative velocity of the free-stream around the blade; and a characteristic density, ρ_0, and characteristic viscosity, μ_0. For both the density and viscosity the free-stream value can be used. With these scaling parameters, the variables in the Navier-Stokes equations can be expressed as

$$\boldsymbol{u} = \mathcal{U} \tilde{\boldsymbol{u}}, \;\; \rho = \rho_0 \tilde{\rho}, \;\; t = \frac{\mathcal{L}}{\mathcal{U}} \tilde{t}, \;\; p = \rho_0 \mathcal{U}^2 \tilde{p}, \;\; \underline{\underline{\tau}} = \mu_0 \frac{\mathcal{U}}{\mathcal{L}} \underline{\underline{\tilde{\tau}}}, \;\; \mathrm{dS} = \mathcal{L}^2 \mathrm{d}\tilde{S}, \;\; \mathrm{dV} = \mathcal{L}^3 \mathrm{d}\tilde{V},$$

where the variables denoted with a tilde are non-dimensional. Note, that for $\boldsymbol{u}_{\partial \mathrm{V}}$ a similar expression as for $\boldsymbol{u}$ can be found. Substitution of these expressions in the Navier-Stokes equations of equation (5.1), yields

$$\frac{\partial}{\partial \tilde{t}} \int\limits_{V(\tilde{t})} \tilde{\rho} \tilde{\boldsymbol{u}} \, \mathrm{d}\tilde{V} + \int\limits_{\partial V(\tilde{t})} \tilde{\rho} \tilde{\boldsymbol{u}} \left(\left[\tilde{\boldsymbol{u}}^T - \tilde{\boldsymbol{u}}_{\partial V}^T \right] \cdot \boldsymbol{n} \right) \mathrm{d}\tilde{S} = - \int\limits_{\partial V(\tilde{t})} \tilde{p} \boldsymbol{n} \, \mathrm{d}\tilde{S} + \mathrm{Re}^{-1} \int\limits_{\partial V(\tilde{t})} \underline{\underline{\tilde{\tau}}} \cdot \boldsymbol{n} \, \mathrm{d}\tilde{S}, \tag{5.2}$$

where $\mathrm{Re} := \frac{\rho_0 \mathcal{U} \mathcal{L}}{\mu_0}$ represents the Reynolds number. This non-dimensional number was named after Osborne Reynolds [106] — one of the founders of modern fluid dynamics — and represents the ratio between inertia and viscous forces in a flow. Inspection of the non-dimensional Navier-Stokes equations learns that two limiting cases can be distinguished. For $\mathrm{Re} \ll 1$ the viscous term dominates the equations and inertia terms can be neglected. On the other hand, if $\mathrm{Re} \gg 1$, the viscous term has a negligible influence relative to the inertia terms and the hyperbolic nature of the equations becomes predominant. For a typical modern wind turbine, the Reynolds number is of $\mathcal{O}\left(10^6\right)$ for a large part of the blade. It is therefore permitted to ignore the term representing the viscous effects for this kind of application. Note however, that in the region of the blade near the hub of the rotor viscous effects can be quite important. This observation must be taken into account, when optimization results are considered, since physical behaviour that is not modelled, can also not be considered by the optimization method.

By dropping the last term of equation (5.2), the Euler equations are obtained. Together with the equations for conservation of mass and energy the equations that govern the motion of a fluid are expressed in dimensional integral conservation form as

$$\frac{\partial}{\partial t}\int\limits_{V(t)} \boldsymbol{U}\,\mathrm{d}V + \int\limits_{\partial V(t)} \left[\underline{\underline{F}}\left(\boldsymbol{U}\right) - \boldsymbol{U}\boldsymbol{u}_{\partial\mathrm{V}}^{T}\right]\cdot\boldsymbol{n}\,\mathrm{d}S = \boldsymbol{0}. \tag{5.3}$$

Note, that in the equation for energy conservation the effects due to viscosity and heat conduction are neglected. In this set of equations the vector with conserved variables is given by

$$\boldsymbol{U} := \left(\rho, \rho\boldsymbol{u}, \rho e_{\mathrm{t}}\right)^{T} \in \mathbb{R}^{5}, \tag{5.4}$$

where $e_{\mathrm{t}} \equiv e_{\mathrm{int}} + 1/2\,||\boldsymbol{u}||^{2}$ represents the total energy per unit mass, with e_{int} the internal energy per unit mass. The convective flux tensor is defined as

$$\underline{\underline{F}}\left(\boldsymbol{U}\right) \equiv \left[\boldsymbol{F}_{\mathrm{x}}\left(\boldsymbol{U}\right), \boldsymbol{F}_{\mathrm{y}}\left(\boldsymbol{U}\right), \boldsymbol{F}_{\mathrm{z}}\left(\boldsymbol{U}\right)\right] := \left(\rho\boldsymbol{u}^{T}, \rho\boldsymbol{u}\boldsymbol{u}^{T} + p\underline{\underline{I}}, \left[p + \rho e_{\mathrm{t}}\right]\boldsymbol{u}^{T}\right)^{T}, \tag{5.5}$$

with identity matrix $\underline{\underline{I}} \in \mathbb{R}^{3}$. Moreover, the thermodynamic equations of state of a calorically perfect gas are used to relate the static pressure to the density and internal energy. For a calorically perfect gas, the internal energy is equal to the product of the specific heat capacity at constant volume and the absolute temperature. Using this relation together with the perfect gas law, the pressure can be expressed as

$$p = \rho\left(\gamma - 1\right)\left(e_{\mathrm{t}} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2}\right), \tag{5.6}$$

with γ the ratio of specific heats, which is taken to be equal to 1.4 for air.

5.2 Spatial discretization

In order to solve the system of integral conservation equations of equation (5.3) numerically, these equations must be discretized. For this purpose, a finite volume method is employed, which is explained next.

5.2.1 Finite volume method

If the control volume is chosen to be equal to a cell, for instance cell $\mathrm{a} \in \mathbb{N}_0$, of the grid that is used for the discretization of a stationary flow domain, the aforementioned conservation equations can be expressed for control volume a as

$$V_{\mathrm{a}}\frac{\mathrm{d}\left\langle\boldsymbol{U}\right\rangle_{\mathrm{a}}}{\mathrm{d}t} + \int\limits_{\partial V_{\mathrm{a}}} \underline{\underline{F}}\left(\boldsymbol{U}\right)\cdot\boldsymbol{n}\,\mathrm{d}S = \boldsymbol{0}. \tag{5.7}$$

where the volume of the control volume V_{a} and the control-volume-averaged value of the conserved variables $\left\langle\boldsymbol{U}\right\rangle_{\mathrm{a}}$ are defined by

$$V_{\mathrm{a}} := \int\limits_{V_{\mathrm{a}}} \mathrm{d}V \qquad \text{and} \qquad \left\langle\boldsymbol{U}\right\rangle_{\mathrm{a}} := \frac{1}{V_{\mathrm{a}}}\int\limits_{V_{\mathrm{a}}} \boldsymbol{U}\,\mathrm{d}V,$$

respectively. Note, that the term $\left(\boldsymbol{U}\boldsymbol{u}_{\partial \mathrm{V}}^{T}\right)$ present in equation (5.3) is absent, because the domain, and therefore also its discrete representation, is assumed stationary and rigid. For a discrete hexahedral shaped cell, the integral of the flux over the boundary of the control volume can be assembled as a summation of the flux over each of the six faces of the hexahedral control volume. Equation (5.7) can then be expressed as

$$V_{\mathrm{a}}\frac{\mathrm{d}\left\langle \boldsymbol{U}\right\rangle_{\mathrm{a}}}{\mathrm{d}t}+\sum_{\mathrm{m}=0}^{\mathrm{m}=5}\int\limits_{\partial V_{\mathrm{am}}}\underline{\underline{F}}\left(\boldsymbol{U}\right)\cdot\boldsymbol{n}_{\mathrm{m}}\,\mathrm{d}S=\boldsymbol{0}. \tag{5.8}$$

Based on the rotational invariance property of the Euler equations [100,187], the equation can be rewritten to

$$V_{\mathrm{a}}\frac{\mathrm{d}\left\langle \boldsymbol{U}\right\rangle_{\mathrm{a}}}{\mathrm{d}t}+\sum_{\mathrm{m}=0}^{\mathrm{m}=5}\int\limits_{\partial V_{\mathrm{am}}}\underline{\underline{T}}^{-1}\boldsymbol{F}_{\mathrm{x}}\left(\underline{\underline{T}}\boldsymbol{U}\right)\,\mathrm{d}S=\boldsymbol{0}, \tag{5.9}$$

replacing the dot product of the flux tensor with the normal vector by a single flux evaluation for a transformed vector with conserved variables and transforming the result back to the original direction. In this equation, the rotation matrix $\underline{\underline{T}}$ is given by

$$\underline{\underline{T}}:=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & n_{\mathrm{x}} & n_{\mathrm{y}} & n_{\mathrm{z}} & 0\\ 0 & t_{1,\mathrm{x}} & t_{1,\mathrm{y}} & t_{1,\mathrm{z}} & 0\\ 0 & t_{2,\mathrm{x}} & t_{2,\mathrm{y}} & t_{2,\mathrm{z}} & 0\\ 0 & 0 & 0 & 0 & 1\end{bmatrix}, \tag{5.10}$$

with $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ the unit vectors tangent to the surface, which form an orthogonal system together with $\boldsymbol{n}$. Note, that the inverse of $\underline{\underline{T}}$ is easily obtained by taking its transpose. The advantage of applying this transformation is that it suffices to compute a single unidirectional flux and account for the result for the other directions by application of the inverse transformation.

Due to the hyperbolic nature of the set of equations considered, special care must be taken in the treatment of the flux in the numerical method [127]. For this purpose, the numerical flux is introduced as an approximation of the actual flux, i.e.

$$\boldsymbol{\mathcal{F}}\left(\boldsymbol{U}_{\mathrm{L}},\boldsymbol{U}_{\mathrm{R}},\boldsymbol{n}_{\mathrm{m}}\right)\approx\underline{\underline{T}}^{-1}\boldsymbol{F}_{\mathrm{x}}\left(\underline{\underline{T}}\boldsymbol{U}\right), \tag{5.11}$$

where $\boldsymbol{U}_{\mathrm{L}}$ and $\boldsymbol{U}_{\mathrm{R}}$ are the states of the vector with conserved variables on the left and right side of interface m, respectively. Using this approximation and assuming that the numerical flux is constant over the interface considered, equation (5.9) can be written as

$$V_{\mathrm{a}}\frac{\mathrm{d}\left\langle \boldsymbol{U}\right\rangle_{\mathrm{a}}}{\mathrm{d}t}+\sum_{\mathrm{m}=0}^{\mathrm{m}=5}\boldsymbol{\mathcal{F}}\left(\boldsymbol{U}_{\mathrm{L}},\boldsymbol{U}_{\mathrm{R}},\boldsymbol{n}_{\mathrm{m}}\right)S_{\mathrm{m}}=\boldsymbol{0}, \tag{5.12}$$

where S_{m} is the surface area of face m. For a first-order accurate spatial discretization, an upwind method can be used with the control-volume-averaged state of the elements of the vector with conserved variables as an approximation of the state at the interface. To achieve a second-order spatial accuracy with an upwind method, linear reconstruction of the state at the interface needs to be performed. For the simulation of transonic flows, the occurrence of spurious oscillations — due to the linear reconstruction of the state at the interface – can be prevented by application of a so-called MUSCL-type [193] reconstruction. Alternatively, second-order spatial accuracy can also be achieved with a central discretization of the flux, with added artificial dissipation terms to achieve stability. Both approaches are subject of section 5.3.

5.2.2 Computation of metrics

The evaluation of equation (5.12) requires the metrics of the grid to be available. These metrics include the surface area of the faces bounding the control volumes, their corresponding surface unit normal vectors and the volume of the control volumes. The location of the centre of mass of the control volumes is also required for the application of certain boundary conditions, for the reconstruction of the state at the interface and also for the donor search, when a composite overset domain discretization is employed.

The surface normal vectors of a face are computed by taking the cross product of the two diagonal vectors of that face. The magnitude of this vector equals twice the surface area. The unit normal vector can subsequently be obtained by dividing the vector by its magnitude. However, the unnormalized result is stored, because the product of the unit normal vector and the surface area is what is usually required. This approach saves computing the magnitude, performing the normalization and storing them separately.

The volume of a control volume is computed employing the tri-linear transformation also used in the donor search for fringe control volumes, given by the equation

$$\boldsymbol{r}'(\boldsymbol{\xi}) = \frac{1}{8}\sum_{\mathrm{k}=0}^{\mathrm{k}=1}\left(\sum_{\mathrm{j}=0}^{\mathrm{j}=1}\left[\sum_{\mathrm{i}=0}^{\mathrm{i}=1}\left(\left[(-1)^{\mathrm{i}}(1-2\xi)+1\right]\left[(-1)^{\mathrm{j}}(1-2\eta)+1\right]\right.\right.\right.$$
$$\left.\left.\left.\left[(-1)^{\mathrm{k}}(1-2\zeta)+1\right]\boldsymbol{r}'_{\mathrm{n}}\right)\right]\right), \qquad (4.6)$$

where $\boldsymbol{r}'_{\mathrm{n}}$ represent the coordinates of the vertices of the control volume after performing the translation to the origin and $\boldsymbol{\xi} \equiv (\xi, \eta, \zeta)^T$. The vertices are numbered in the same order as shown in figure 4.16 on page 71, with $\mathrm{n} = 4\mathrm{k} + 2\mathrm{j} + \mathrm{i}$. The volume is then computed by performing the integration over the tri-linear coordinate directions:

$$V = \int_0^1\int_0^1\int_0^1 \det\left(\frac{\partial \boldsymbol{r}'}{\partial \boldsymbol{\xi}}\right)\, \mathrm{d}\xi\, \mathrm{d}\eta\, \mathrm{d}\zeta, \qquad (5.13)$$

which requires the determinant of the Jacobian matrix of the transformation specified by equation (4.6).

Using this same coordinate transformation, the location of the centre of mass, $\boldsymbol{x}_{\mathsf{cm}}$, of a control volume can be determined with

$$\boldsymbol{x}_{\mathsf{cm}} = \frac{1}{V}\int_0^1\int_0^1\int_0^1 \boldsymbol{r}'(\boldsymbol{\xi}) \det\left(\frac{\partial \boldsymbol{r}'}{\partial \boldsymbol{\xi}}\right)\, \mathrm{d}\xi\, \mathrm{d}\eta\, \mathrm{d}\zeta. \qquad (5.14)$$

5.3 Numerical fluxes

The hyperbolic nature of the convective flux requires a special treatment in the numerical method to accurately compute the flux at the interface of two control volumes. The flux can be computed in different ways; one way is to exactly solve the Riemann problem — i.e.

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial}{\partial x}\boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{0}, \tag{5.15}$$

for an interface normal to the x-direction — at the interface and use the result to compute the flux at the interface. Also other methods exist, for which either an approximate solution of the Riemann problem is used to compute the flux at the interface or a central discretization of the flux is computed. The latter two methods are discussed in the following subsections, starting with the central discretization of the flux using the so-called Jameson-Schmidt-Turkel (JST) scheme [92]. An example of a very commonly used upwind method is Roe's approximate Riemann solver [147], this method is treated in subsection 5.3.2.

5.3.1 Jameson-Schmidt-Turkel scheme

The JST scheme [92] uses a central discretization of the flux at the interface. However, since a central discretization allows for the occurrence of odd-even decoupling, artificial dissipation must be added to obtain a stable scheme. This artificial dissipation is also required to remove the energy from modes for which the wavelength cannot be resolved by the computational grid, giving rise to aliasing. These high frequency modes have a large amplitude when discontinuities are present in the flow solution, such as in the case of the appearance of shocks in transonic and supersonic flow. Therefore, more artificial dissipation is required to realize a stable numerical scheme, in the event of a shock. With the addition of an artificial dissipation term, the flux at the interface between control volume i and control volume $\mathrm{i}+1$ is computed according to

$$\boldsymbol{\mathcal{F}}\left(\boldsymbol{U}_{\mathrm{L}}, \boldsymbol{U}_{\mathrm{R}}, \boldsymbol{n}_{\mathrm{i}+\frac{1}{2}}\right) = \frac{1}{2}\left[\boldsymbol{F}\left(\langle\boldsymbol{U}\rangle_{\mathrm{i}}\right) + \boldsymbol{F}\left(\langle\boldsymbol{U}\rangle_{\mathrm{i}+1}\right)\right] - \boldsymbol{D}_{\mathrm{i}+\frac{1}{2}}, \tag{5.16}$$

with $\boldsymbol{D}_{\mathrm{i}+\frac{1}{2}}$ the artificial dissipation term and using the control-volume-averaged values of the conserved variables to represent the state at the left and right of the interface. This artificial dissipation term consists of a second-order shock capturing term and a fourth-order background dissipation term, which can be expressed as

$$\boldsymbol{D}_{\mathrm{i}+\frac{1}{2}} = \boldsymbol{D}^{\mathrm{II}}_{\mathrm{i}+\frac{1}{2}} - \boldsymbol{D}^{\mathrm{IV}}_{\mathrm{i}+\frac{1}{2}}. \tag{5.17}$$

The shock capturing term is proportional to a first-order forward-difference operator applied to the control-volume-averaged value of the conserved variables:

$$\boldsymbol{D}^{\mathrm{II}}_{\mathrm{i}+\frac{1}{2}} = \epsilon^{\mathrm{II}}_{\mathrm{i}+\frac{1}{2}}\left(\langle\boldsymbol{U}\rangle_{\mathrm{i}+1} - \langle\boldsymbol{U}\rangle_{\mathrm{i}}\right)\Lambda_{\mathrm{i}+\frac{1}{2}}. \tag{5.18}$$

The background dissipation term is proportional to the first-order accurate third-order forward-difference operator applied to the control-volume-averaged value of the conserved variables:

$$\boldsymbol{D}^{\mathrm{IV}}_{\mathrm{i}+\frac{1}{2}} = \epsilon^{\mathrm{IV}}_{\mathrm{i}+\frac{1}{2}}\left(\langle\boldsymbol{U}\rangle_{\mathrm{i}+2} - 3\langle\boldsymbol{U}\rangle_{\mathrm{i}+1} + 3\langle\boldsymbol{U}\rangle_{\mathrm{i}} - \langle\boldsymbol{U}\rangle_{\mathrm{i}-1}\right)\Lambda_{\mathrm{i}+\frac{1}{2}}. \tag{5.19}$$

In both equations, the coefficient $\Lambda_{\mathrm{i}+\frac{1}{2}}$ is an estimate of the spectral radius of the local convective flux Jacobian, which reads

$$\Lambda_{\mathrm{i}+\frac{1}{2}} = \frac{1}{2}\left(\left|\langle \boldsymbol{u}\rangle_{\mathrm{i}}\right| + \langle c\rangle_{\mathrm{i}} + \left|\langle \boldsymbol{u}\rangle_{\mathrm{i}+1}\right| + \langle c\rangle_{\mathrm{i}+1}\right), \tag{5.20}$$

where $\langle c\rangle_{\mathrm{i}} = \sqrt{\gamma p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}}\right) / \langle \rho\rangle_{\mathrm{i}}}$ is the local speed of sound determined using control-volume-averaged density and pressure determined based on the control-volume-averaged conserved flow variables. The dimensionless artificial dissipation coefficients are defined as

$$\epsilon^{\mathrm{II}}_{\mathrm{i}+\frac{1}{2}} = \min\left\{\frac{1}{4}, k_2 \Theta_{\mathrm{i}+\frac{1}{2}}\right\}, \tag{5.21}$$

$$\epsilon^{\mathrm{IV}}_{\mathrm{i}+\frac{1}{2}} = \max\left\{0, k_4 - \beta\Theta_{\mathrm{i}+\frac{1}{2}}\right\}. \tag{5.22}$$

Parameters k_2, k_4 and β are dimensionless constants and $\Theta_{\mathrm{i}+1/2}$ is a so-called 'shock sensor' or pressure-switch function, which is used to detect regions with a high pressure gradient. For this purpose, the pressure sensor is defined as

$$\Theta_{\mathrm{i}+\frac{1}{2}} = \max\left\{\Theta_{\mathrm{i}}, \Theta_{\mathrm{i}+1}\right\}, \text{ where} \tag{5.23}$$

$$\Theta_{\mathrm{i}} = \frac{\left|p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}+1}\right) - 2p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}}\right) + p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}-1}\right)\right|}{p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}+1}\right) + 2p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}}\right) + p\left(\langle \boldsymbol{U}\rangle_{\mathrm{i}-1}\right)}. \tag{5.24}$$

The dissipation coefficients are defined such that in regions of strong variation in the pressure the fourth-order dissipation term vanishes and the second-order dissipation term is $\mathcal{O}(1)$, making the scheme behave locally as first-order accurate. In regions where the pressure shows smooth behaviour, the second-order dissipation term is small and the fourth-order term is non-zero, resulting in a second-order spatial accuracy. Typical values for the dimensionless constants [20, 199] are $k_2 \in [1/4, 1/2]$, $k_4 \in [1/128, 1/32]$ and $\beta = 2$.

5.3.2 Roe's approximate Riemann solver

In the method of Roe's approximate Riemann solver, the Riemann problem is linearized, resulting in

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial F}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}}{\partial x} \equiv \frac{\partial \boldsymbol{U}}{\partial t} + \underline{\underline{A}}\left(\boldsymbol{U}\right)\frac{\partial \boldsymbol{U}}{\partial x} = \boldsymbol{0}. \tag{5.25}$$

Subsequently, an approximation $\underline{\underline{\tilde{A}}}\left(\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}\right)$, to the flux Jacobian is sought, that retains the properties of the original problem:

(i) the system is hyperbolic, i.e. $\underline{\underline{\tilde{A}}}\left(\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}\right)$ has real eigenvalues $\tilde{\lambda}_{\mathrm{m}}$, $\mathrm{m} \in \mathbb{N}_0$, $\mathrm{m} < 5$, and a corresponding set of linearly independent right eigenvectors $\tilde{\boldsymbol{K}}^{(\mathrm{m})}$;

(ii) the matrix is consistent with the exact Jacobian, i.e. as $\boldsymbol{U}_{\mathsf{L}} \to \boldsymbol{U}_{\mathsf{R}} \to \hat{\boldsymbol{U}}$, then

$$\underline{\underline{\tilde{A}}}\left(\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}\right) \to \left.\frac{\partial F}{\partial \boldsymbol{U}}\right|_{\hat{\boldsymbol{U}}};$$

(iii) conservation is assured, i.e. $\underline{\underline{\tilde{A}}}\left(\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}\right)\left[\boldsymbol{U}_{\mathsf{R}} - \boldsymbol{U}_{\mathsf{L}}\right] = \boldsymbol{F}\left(\boldsymbol{U}_{\mathsf{R}}\right) - \boldsymbol{F}\left(\boldsymbol{U}_{\mathsf{L}}\right)$.

This goal is achieved by using so-called Roe-averaged variables for the evaluation of the entries of the flux Jacobian. These variables read

$$\tilde{\rho} := \sqrt{\rho_{\mathsf{L}}\rho_{\mathsf{R}}}, \qquad \tilde{\boldsymbol{u}} := \frac{\sqrt{\rho_{\mathsf{L}}}\boldsymbol{u}_{\mathsf{L}} + \sqrt{\rho_{\mathsf{R}}}\boldsymbol{u}_{\mathsf{R}}}{\sqrt{\rho_{\mathsf{L}}} + \sqrt{\rho_{\mathsf{R}}}}, \qquad \tilde{H} := \frac{\sqrt{\rho_{\mathsf{L}}}H_{\mathsf{L}} + \sqrt{\rho_{\mathsf{R}}}H_{\mathsf{R}}}{\sqrt{\rho_{\mathsf{L}}} + \sqrt{\rho_{\mathsf{R}}}},$$

where $H \equiv e_{\mathsf{t}} + \rho^{-1}p$ is the total enthalpy per unit mass. The eigenvalues expressed in terms of Roe-averaged quantities are [95, 187]:

$$\tilde{\lambda}_0 = \tilde{u} - \tilde{c}, \qquad \tilde{\lambda}_1 = \tilde{u}, \qquad \tilde{\lambda}_2 = \tilde{u}, \qquad \tilde{\lambda}_3 = \tilde{u}, \qquad \tilde{\lambda}_4 = \tilde{u} + \tilde{c},$$

where $\tilde{c}$ is the speed of sound evaluated using Roe-averaged quantities. The right eigenvectors are

$$\tilde{K}^{(0)} = \begin{pmatrix} 1 \\ \tilde{u} - \tilde{c} \\ \tilde{v} \\ \tilde{w} \\ \tilde{H} - \tilde{u}\tilde{c} \end{pmatrix}, \tilde{K}^{(1)} = \begin{pmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \frac{1}{2}\left|\tilde{\boldsymbol{u}}\right|^2 \end{pmatrix}, \tilde{K}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \tilde{v} \end{pmatrix}, \tilde{K}^{(3)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \tilde{w} \end{pmatrix}, \tilde{K}^{(4)} = \begin{pmatrix} 1 \\ \tilde{u} + \tilde{c} \\ \tilde{v} \\ \tilde{w} \\ \tilde{H} + \tilde{u}\tilde{c} \end{pmatrix}.$$

The hyperbolic system allows the jump in the solution to be expressed as a linear combination of the eigenvectors, i.e.

$$\boldsymbol{U}_{\mathsf{R}} - \boldsymbol{U}_{\mathsf{L}} = \sum_{\mathrm{m}=0}^{\mathrm{m}=4} \alpha_{\mathrm{m}} \tilde{\boldsymbol{K}}^{(\mathrm{m})}, \tag{5.26}$$

where α_{m} represents the wave strength, associated with the direction of eigenvector $\tilde{\boldsymbol{K}}^{(\mathrm{m})}$. Since the jump in conserved variables across the interface is known, equation (5.26) can be used to solve for the wave strengths. The solution yields:

$$\alpha_0 = \frac{\Delta p - \tilde{c}\tilde{\rho}\Delta u}{2\tilde{c}^2}, \quad \alpha_1 = \Delta\rho - \frac{\Delta p}{\tilde{c}^2}, \quad \alpha_2 = \frac{\tilde{\rho}\Delta v}{\tilde{v}}, \quad \alpha_3 = \frac{\tilde{\rho}\Delta w}{\tilde{w}}, \quad \alpha_4 = \frac{\Delta p + \tilde{c}\tilde{\rho}\Delta u}{2\tilde{c}^2}.$$

Using this result, the flux across the interface between control volume i and control volume $\mathrm{i}+1$ can be computed from

$$\boldsymbol{\mathcal{F}}\left(\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}, \boldsymbol{n}_{\mathrm{i}+\frac{1}{2}}\right) = \frac{1}{2}\left[\boldsymbol{\mathcal{F}}\left(\boldsymbol{U}_{\mathsf{L}}\right) + \boldsymbol{\mathcal{F}}\left(\boldsymbol{U}_{\mathsf{R}}\right)\right] - \frac{1}{2}\sum_{\mathrm{m}=0}^{\mathrm{m}=4} \alpha_{\mathrm{m}} \left|\tilde{\lambda}_{\mathrm{m}}\right| \tilde{\boldsymbol{K}}^{(\mathrm{m})}. \tag{5.27}$$

When for the state at the left and right of the interface the control-volume-averaged quantities are used of control volume i and control volume $\mathrm{i}+1$, respectively, this flux scheme is first-order accurate. Second-order accuracy can be achieved, using linear reconstruction of the quantity at the interface. In transonic flow, linear reconstruction gives rise to non-physical oscillatory behaviour of the solution. This non-physical behaviour can be mitigated using a MUSCL-type reconstruction instead, both methods are treated in the next section.

In the Roe scheme as such, expansion shocks violating the entropy condition [187] are admissible. To prevent the occurrence of such non-physical behaviour, various authors have suggested different entropy fixes. In the current implementation, the approach proposed by Harten [78] is used.

5.4 Reconstruction of state at interface

To achieve higher than first-order spatial accuracy for an upwind method, it is not sufficient to use control-volume-averaged values for the evaluation of the flux at the interfaces. In that case, a linear reconstruction of the state at the interface is required. Linear reconstruction requires information on the gradient of the flow solution. Subsequently using this gradient information, the state at the interface can be reconstructed. Different approaches can be taken for the computation of gradients and the reconstruction of the state at the interface. Two methods are discussed. The first method is the more accurate of the two. The second method trades some of this accuracy for an increase in robustness of the method. The general formula for the 1D reconstruction of the state at the interface, for flow variable φ, reads

$$\varphi_{\mathsf{R}} = \langle\varphi\rangle_{\mathrm{i}+1} - \beta_{\mathsf{R}}\left[(1+\hat{\kappa})\,(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}} + (1-\hat{\kappa})\,(\nabla\varphi)_{\mathrm{i}+\frac{3}{2}}\right], \text{ and} \tag{5.28}$$

$$\varphi_{\mathsf{L}} = \langle\varphi\rangle_{\mathrm{i}} \quad + \beta_{\mathsf{L}}\left[(1+\hat{\kappa})\,(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}} + (1-\hat{\kappa})\,(\nabla\varphi)_{\mathrm{i}-\frac{1}{2}}\right], \tag{5.29}$$

with a weighted average of the gradients computed at the interfaces of a control volume, with weighting factor $\hat{\kappa} \in [-1, 1]$. For $\hat{\kappa} = 1/3$, a convective flux discretization of third-order spatial accuracy can be achieved [102]. Coefficients β_{R} and β_{L} depend on the method used for the reconstruction of the state at the interface. The methods used for computing the gradients and more details on the actual reconstruction, including the specification of β_{R} and β_{L}, are presented in the following two subsections. Moreover, subsection 5.4.3 considers the situation in which large gradients result in linearly reconstructed states violating the maximum principle and how this situation can be coped with.

5.4.1 Reconstruction about centre of mass

This reconstruction method explicitly considers the location of the centre of mass of the control volume as well as the centroid of the face for which the reconstruction must be performed. The gradient of variable φ is computed for three consecutive interfaces centred around the interface considered. For the interface between control volume $\mathrm{i} \in \mathbb{N}_0$ and $\mathrm{i}+1$ the gradient is computed according to

$$(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}} = \frac{\langle\varphi\rangle_{\mathrm{i}+1} - \langle\varphi\rangle_{\mathrm{i}}}{\left|(\boldsymbol{x}_{\mathsf{cm}})_{\mathrm{i}+1} - (\boldsymbol{x}_{\mathsf{cm}})_{\mathrm{i}}\right|} \tag{5.30}$$

For the reconstruction of the state at the interface, the distance between the centre of mass and the centre of the face is used, therefore

$$\beta_{\mathsf{R}} = \left|(\boldsymbol{x}_{\mathsf{F}})_{\mathrm{i}+\frac{1}{2}} - (\boldsymbol{x}_{\mathsf{cm}})_{\mathrm{i}+1}\right|,$$

$$\beta_{\mathsf{L}} = \left|(\boldsymbol{x}_{\mathsf{F}})_{\mathrm{i}+\frac{1}{2}} - (\boldsymbol{x}_{\mathsf{cm}})_{\mathrm{i}}\right|,$$

where $(\boldsymbol{x}_{\mathsf{F}})_{\mathrm{i}+\frac{1}{2}}$ is the location of the centroid of the face between control volume i and $\mathrm{i}+1$. By performing the reconstruction about the centre of mass of the control volume, the control-volume-averaged value is conserved [95].

5.4.2 Reconstruction considering grid as uniform Cartesian

The method presented in the preceding section generally provides an accurate approximation of the state at the interface. However, for grids with high aspect ratio cells in regions near curved surfaces, it can occur that the centre of mass of a control volume is not in the control volume itself. In that case using the location of the centre of mass for the reconstruction of the state at the interface, can lead to inconsistencies, hampering convergence of the solution method. Therefore, in this section a different approach for reconstruction of the state at the interface is presented. For this method, the assumption is made that a uniform Cartesian grid is used for the discretization of the computational domain. In this way, the distance between the centre of mass of a control volume and the centroid of the face at which the gradient is approximated does not need to be considered explicitly. This approach increases the robustness of the method for situations in which grids with high aspect ratio cells are used in the region near curved surfaces. The undivided gradient scaled with the distance at the interface between control volume i and $\mathrm{i}+1$ is now approximated as

$$(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}} = \frac{\langle\varphi\rangle_{\mathrm{i}+1} - \langle\varphi\rangle_{\mathrm{i}}}{2}. \tag{5.31}$$

Note, that the actual dimensions of the grid do not matter in the computation of the gradient, because this information drops out in the final result. The corresponding choices for β_{R} and β_{L} are

$$\beta_{\mathsf{R}} = \beta_{\mathsf{L}} = \frac{1}{2}. \tag{5.32}$$

The increase in robustness of the method comes at the price of a reduction of the order of accuracy, because reconstruction is not performed about the centre of mass.

5.4.3 MUSCL-type reconstruction

In regions where the flow solution exhibits large gradients, linear reconstruction of the state at the interface can result in a reconstructed state that does not satisfy the maximum principle. In this situation the reconstructed value exceeds the extrema in its stencil, which can give rise to the occurrence of spurious oscillations in the solution. Therefore, it is necessary to limit the gradients in order to avoid this situation. This solution was originally proposed by Van Leer [193], who named the method the Monotone Upstream-Centred Scheme for Conservation Laws (MUSCL). Since its introduction, many MUSCL-like methods have been constructed, which have in common that some sort of slope limiting is applied in case the gradients are too large. In the present research the Van Albada limiter [5] is used. The corresponding limiting functions read for the interface between control volume i and $\mathrm{i}+1$

$$\Phi^{-}_{\mathrm{i}+\frac{1}{2}} = \frac{2\,(\nabla\varphi)_{\mathrm{i}-\frac{1}{2}}\,(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}} + \epsilon_{\mathrm{i}}}{\left[(\nabla\varphi)_{\mathrm{i}-\frac{1}{2}}\right]^2 + \left[(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}}\right]^2 + \epsilon_{\mathrm{i}}}, \tag{5.33}$$

$$\Phi^{+}_{\mathrm{i}+\frac{1}{2}} = \frac{2\,(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}}\,(\nabla\varphi)_{\mathrm{i}+\frac{3}{2}} + \epsilon_{\mathrm{i}+1}}{\left[(\nabla\varphi)_{\mathrm{i}+\frac{1}{2}}\right]^2 + \left[(\nabla\varphi)_{\mathrm{i}+\frac{3}{2}}\right]^2 + \epsilon_{\mathrm{i}+1}}. \tag{5.34}$$

Variable ϵ_{i} is used to prevent an undefined result in regions of uniform flow, for this purpose, it is defined as

$$\epsilon_{\mathrm{i}} := \left(\epsilon_{\mathsf{a}} \left[\left| \langle \varphi \rangle_{\mathrm{i}} \right| + \epsilon_{\mathsf{b}} \right] \right)^2 , \tag{5.35}$$

where $\epsilon_{\mathsf{a}} \in \mathbb{R}$ and $\epsilon_{\mathsf{b}} \in \mathbb{R}$ are some constants for which the values 0.1 and 0.01 are used, respectively, in the present implementation. Note, that the unit of ϵ_{b} is taken to be equal to that of the flow variable for which limiting is applied, to end up with a dimensionally consistent equation. Including these limiting functions, the approximation of the state at the interface is now computed by taking a weighted average of the slope limited approximation of the derivative at both interfaces — in one direction — of the control volume, i.e.

$$\varphi_{\mathsf{R}} = \langle \varphi \rangle_{\mathrm{i}+1} - \beta_{\mathsf{R}} \left[(1+\hat{\kappa})\, \Phi^{+}_{\mathrm{i}+\frac{1}{2}} \left(\nabla \varphi\right)_{\mathrm{i}+\frac{1}{2}} + (1-\hat{\kappa})\, \Phi^{-}_{\mathrm{i}+\frac{3}{2}} \left(\nabla \varphi\right)_{\mathrm{i}+\frac{3}{2}} \right], \text{ and} \tag{5.36}$$

$$\varphi_{\mathsf{L}} = \langle \varphi \rangle_{\mathrm{i}} \quad + \beta_{\mathsf{L}} \left[(1+\hat{\kappa})\, \Phi^{-}_{\mathrm{i}+\frac{1}{2}} \left(\nabla \varphi\right)_{\mathrm{i}+\frac{1}{2}} + (1-\hat{\kappa})\, \Phi^{+}_{\mathrm{i}-\frac{1}{2}} \left(\nabla \varphi\right)_{\mathrm{i}-\frac{1}{2}} \right]. \tag{5.37}$$

5.5 Boundary conditions

Boundary conditions are used to represent the inevitability of the flow domain being bounded. The boundary can have a physical nature, such as a solid wall, or it can be artificial. The latter is the case for example for periodic boundaries or for a far-field boundary, which accounts for the computational domain being finite. In the present research, boundary conditions are enforced using halo control volumes — also known as ghost control volumes. The number of halo control volumes used depends on the stencil used for the evaluation of the numerical flux. For a second-order accurate spatial discretization this number is equal to two. The flow solution enforced at these halo control volumes depends on the boundary condition that it represents. For the boundary conditions employed in the present work, the approach is discussed in the following subsections.

5.5.1 Solid wall

Neglecting effects of viscosity — in the mathematical model representing the flow — results in a reduction of the order of the partial differential equations, compared to a model that includes effects of viscosity. This reduction, in turn, affects the physical boundary conditions that can be represented in the mathematical model. Enforcing a so-called no-slip condition at a solid wall is therefore not possible in a flow governed by the Euler equations. In inviscid flows, solid walls are represented by a zero normal velocity at the wall. For this purpose, the velocity vector in the first halo control volume, $\boldsymbol{u}_{\mathsf{H0}}$, is computed according to

$$\boldsymbol{u}_{\mathsf{H0}} = \boldsymbol{u}_{\mathsf{D0}} - 2 \left(\boldsymbol{u}_{\mathsf{D0}} \cdot \boldsymbol{n}_{\mathsf{S}} \right) \boldsymbol{n}_{\mathsf{S}}, \tag{5.38}$$

where $\boldsymbol{u}_{\mathsf{D0}}$ is the velocity vector for the first control volume in the interior domain next to the face representing a solid wall, and $\boldsymbol{n}_{\mathsf{S}}$ the unit surface normal vector of the corresponding face, pointing in the direction outward of the flow domain; see figure 5.2 for a schematic representation. The value of the density assigned to the halo control volume is equal to ρ_{D0} and the pressure for the halo control volume is obtained by means of a linear extrapolation of the pressure from inside the domain, based on the value of the

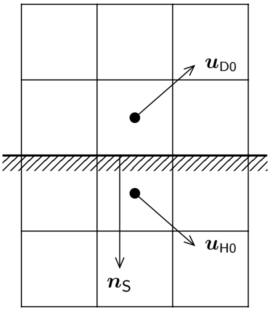


Figure 5.2: Schematic representation of solid wall boundary condition. Hatched area indicates the region outside of the computational domain.

pressure of the control volume in the first and second layer in the interior of the domain and taking into account the location of the respective centres of mass. Subsequently, the conserved flow variables for the halo control volume are determined based on the primitive variables just computed.

When a flux discretization with second-order spatial accuracy is used, the flow solution in the second layer of halo control volumes must also be provided. For these halo control volumes the variables are computed using linear extrapolation of the conserved flow variables, which are then used to determine the primitive variables in the control volumes in the second layer of halo control volumes.

As an alternative to using linear interpolation of the pressure to compute the state for the halo control volume a different approach exists, which uses the local momentum equation to specify the pressure for the halo control volume. This approach reduces the entropy generation and total pressure loss introduced by the boundary condition [95]. However, to employ this method, the local curvature at the solid wall is required, which is not readily available in the present method. Therefore, this curvature corrected boundary condition proposed by Rizzi [146] has not been implemented.

5.5.2 Periodicity

Periodic boundary conditions are employed in situations in which the flow exhibits spatial periodicity. By exploiting spatial periodicity of the flow solution, the flow domain is reduced, allowing for a higher grid resolution in the remaining part of the flow domain. For rotational periodicity, see figure 5.3, the conserved flow variables for the halo control volumes are computed according to

$$\boldsymbol{U}_{\mathrm{H0}} = \underline{\underline{T}}\,(\vartheta)\,\boldsymbol{U}_{\mathrm{D0}}, \qquad \boldsymbol{U}_{\mathrm{H1}} = \underline{\underline{T}}\,(\vartheta)\,\boldsymbol{U}_{\mathrm{D1}}$$

where $\underline{\underline{T}}\,(\vartheta)$ is a transformation matrix, which takes care of the correct rotation of the vector quantity momentum. Angle $\vartheta \in \left\{\frac{2\pi}{\mathrm{n}} : \mathrm{n} \in \mathbb{N}_1\right\}$ is the angle over which periodicity of the solution is observed. Note, that the scalar quantities of the vector with conserved variables are rotationally invariant. Therefore, the top and bottom row of this transformation matrix only have a non-zero diagonal element, of unit magnitude.

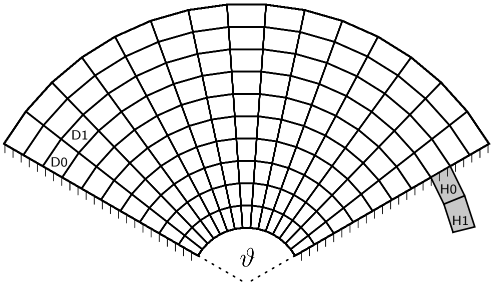


Figure 5.3: Schematic representation of flow domain with point matching rotational periodic boundaries. Halo control volume H0 gets the flow solution from inner domain control volume D0, with the vector quantities transformed, using the corresponding transformation matrix. Similarly, halo control volume H1 is assigned the transformed flow solution from control volume D1.

5.5.3 Symmetry

If the flow solution is supposed to exhibit a symmetrical result, symmetry boundary conditions can be used to enforce this behaviour. The flow solution, assigned to the halo control volumes associated with a symmetry boundary, is obtained by mirroring the flow solution about the boundary surface in a similar way as for the solid wall boundary. The pressure and density for the halo control volume are the same as for the corresponding control volume in the interior domain. For a second-order flux discretization, the flow solution in the second layer of halo control volumes is also obtained by mirroring the flow solution about the boundary surface, now for the control volume in the second layer inside the flow domain.

5.5.4 Far-field

Far-field boundary conditions are imposed in external aerodynamics problems to account for the discretized flow domain being finite. However, in physical reality, no such boundary exists. The existence of the boundary in the discrete representation can give rise to the occurrence of non-physical behaviour. Examples of such non-physical effects are the reflection of pressure waves at the boundary or the sudden dissipation of circulation induced by a lifting body inside the flow domain [20]. Various approaches can be taken to handle the problems associated with a domain of finite dimensions. The most universal and straightforward solution is, however, to place the boundary sufficiently far away, such that effects of the existence of a far-field boundary are negligible. Note, that this approach might not always be appropriate; for example for unsteady flow computations involving travelling pressure waves. For the far-field boundary located sufficiently far away, it is sufficient to just assign the free-stream conditions to the first layer of the corresponding halo control volumes. When appropriate, the flow solution in the second layer of halo control volumes is obtained by linear extrapolation using the values in the first layer inside the flow domain and the first layer outside the domain.

5.5.5 Subsonic outflow

Situations exist for which the flow conditions at the far-field boundary are not correctly represented by the free-stream flow conditions. Another example of such a situation is the flow field downstream of the wind turbine rotor. In the wake of a wind turbine rotor, the flow velocity significantly differs from the free-stream velocity. To determine suitable boundary conditions for situations like these, the properties of the underlying mathematical model are considered. Due to the hyperbolic nature of the Euler equations, the number of boundary conditions imposed must equal the number of characteristics that enter the computational domain [199, 202]. For subsonic outflow, the only characteristic that enters the domain, corresponds to the acoustic wave with eigenvalue $(\boldsymbol{u} \cdot \boldsymbol{n} - c)$ — where $\boldsymbol{n}$ is the unit normal vector pointing into the domain. Therefore, only one boundary condition must be imposed for the subsonic outflow boundary. Considering that the static pressure is constant far downstream from any disturbance, it is a convenient choice to prescribe the free-stream pressure p_∞ at the subsonic outflow boundary. The remaining flow variables can be determined using the property that the Riemann-invariants are constant across a rarefaction wave [187]. For the left rarefaction wave, the wave corresponding to the characteristic travelling into the domain, the Riemann invariants read

$$\boldsymbol{u} \cdot \boldsymbol{n} + \frac{2c}{\gamma - 1} = \text{constant}, \tag{5.39}$$

$$s = \text{constant}, \tag{5.40}$$

where s denotes the entropy. Recall that γ represents the ratio of specific heats. Using these Riemann invariants, the state in the halo control volume can be determined. For the density:

$$\rho_{\text{H0}} = \rho_{\text{D0}} \left(\frac{p_\infty}{p_{\text{D0}}} \right)^{\frac{1}{\gamma}}. \tag{5.41}$$

The density in the halo control volume can subsequently be used to compute the speed of sound associated with the halo control volume

$$c_{\text{H0}} = \sqrt{\gamma \left(\frac{p_{\text{H0}}}{\rho_{\text{H0}}} \right)}. \tag{5.42}$$

Using this value for the speed of sound, the velocity components can be determined according to

$$\boldsymbol{u}_{\text{H0}} = \boldsymbol{u}_{\text{D0}} - \frac{2}{\gamma - 1} \left(c_{\text{H0}} - c_{\text{D0}} \right) \boldsymbol{n}. \tag{5.43}$$

With the primitive variables determined, the conserved variables can be computed. If a second layer of halo control volumes exists, the conserved variables for these control volumes are computed by means of linear extrapolation using the values in the first layer inside the flow domain and the first layer outside the domain.

5.5.6 Overset grids

Overset grid boundary conditions are used when in regions of grid overlap the flow solution must be transferred from one grid to another. In this case, the halo control volumes are represented by the fringe control volumes which have been determined by the overset block connectivity method during the preprocessing of the grid. The conserved flow variables for a halo control volume are determined based on interpolation of the flow solution of the donor control volumes of the particular fringe control volume

$$\langle \boldsymbol{U} \rangle_{\mathsf{fringe}} = \sum_{\mathrm{m}=0}^{\mathrm{N}_{\mathsf{F}}-1} \left(\omega_{\mathrm{m}} \mathbb{U}_{\mathrm{m}}\right), \tag{5.44}$$

where $\mathrm{N}_{\mathsf{F}} \in \mathbb{N}_1$ is the number of cells that are donor of the fringe control volume, ω_{m} the interpolation coefficient corresponding to donor m and $\mathbb{U}_{\mathrm{m}}$ element m of the ordered set of vectors with control-volume-averaged conserved variables of the donor control volumes. When the conserved flow variables have been obtained by means of interpolation, the primitive variables are computed using the appropriate relations for the equations of state.

5.6 Pseudo-time-integration to steady state

In the present research, only flow problems are considered for which a steady-state solution of the governing equations exists. In this situation, the time derivative in the semi-discrete conservation equations (5.12) vanishes. Although time-dependent behaviour is not considered, integration in time is performed as a means of relaxation to a converged solution which corresponds to the steady-state solution. For that purpose, the following approach is taken.

A flow simulation is started with uniform free-stream conditions in every control volume in the flow domain. Then, explicit pseudo-time-integration — by means of a multi-stage Runge-Kutta method — is performed. The aim of this explicit pseudo-time-integration is to obtain a suitable initial condition for Newton's method to commence. Subsequently, Newton's method is used to converge the solution to the final solution.

The residual of the semi-discrete governing equations, for a single control volume $\mathrm{a} \in \mathbb{N}_0$, is defined as

$$\boldsymbol{R}_{\mathrm{a}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right) = -\frac{1}{V_{\mathrm{a}}} \sum_{\mathrm{m}=0}^{\mathrm{m}=5} \boldsymbol{\mathcal{F}}\left[\boldsymbol{U}_{\mathsf{L}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right), \boldsymbol{U}_{\mathsf{R}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right), \boldsymbol{n}_{\mathrm{m}}\right] S_{\mathrm{m}}, \tag{5.45}$$

where $\boldsymbol{\mathcal{U}} := \left(\langle \boldsymbol{U} \rangle_0, \langle \boldsymbol{U} \rangle_1, \ldots, \langle \boldsymbol{U} \rangle_{\mathrm{N}}\right)^T$, i.e. the concatenation of the control-volume-averaged conserved flow variables for all $\mathrm{N} \in \mathbb{N}_1$ control volumes used in the discretization of the domain. Superscript $\mathrm{n} \in \mathbb{N}_0$ denotes the current pseudo-time level.

5.6.1 Runge-Kutta time-integration

The Runge-Kutta method used to integrate the solution from pseudo-time level n to the subsequent level $\mathrm{n}+1$, is a standard low-storage multi-stage Runge-Kutta method, which can be expressed as

$$\begin{aligned} \breve{\boldsymbol{\mathcal{U}}}^{0} &= \boldsymbol{\mathcal{U}}^{\mathrm{n}}, \\ \breve{\boldsymbol{\mathcal{U}}}^{\mathrm{m}+1} &= \breve{\boldsymbol{\mathcal{U}}}^{0} + \beta_{\mathrm{m}} \Delta t_{\mathrm{a}} \boldsymbol{R}_{\mathrm{a}}\left(\breve{\boldsymbol{\mathcal{U}}}^{\mathrm{m}}\right), \quad \text{for } \mathrm{m} = 0, \ldots, 3 \\ \boldsymbol{\mathcal{U}}^{\mathrm{n}+1} &= \breve{\boldsymbol{\mathcal{U}}}^{3}. \end{aligned} \tag{5.46}$$

The value of coefficients β_{m} — which range between 0 and 1 — depends on the characteristics of the method used. In the present approach, the coefficients have been chosen such that the resulting method allows for maximum stability of the scheme. These coefficients read [89, 172]:

$$\beta_0 = \frac{1}{3}, \qquad \beta_1 = \frac{4}{15}, \qquad \beta_2 = \frac{5}{9}, \qquad \beta_3 = 1.$$

The magnitude of the local time-step Δt_{a} is determined based on the CFL-condition [46] as follows

$$\Delta t_{\mathrm{a}} = C \frac{V_{\mathrm{a}}}{\bar{\lambda}_{\mathrm{i}} + \bar{\lambda}_{\mathrm{j}} + \bar{\lambda}_{\mathrm{k}}} \in \mathbb{R}, \tag{5.47}$$

where $C \in \mathbb{R}$ is the Courant number. Although, the coefficients used in the present implementation allow for the use of a Courant number exceeding one, a value of 0.8 is used by default, to assure a stable time-integration. An approximation of the spectral radii — in the three directions of the computational coordinate $(\mathrm{i}, \mathrm{j}, \mathrm{k})^T$ — is obtained by the evaluation of

$$\bar{\lambda}_{\mathrm{i}} = \frac{1}{2} \left| \langle \boldsymbol{u} \rangle_{\mathrm{i}} \cdot \left(\boldsymbol{n}_{\mathrm{i}-\frac{1}{2}} S_{\mathrm{i}-\frac{1}{2}} + \boldsymbol{n}_{\mathrm{i}+\frac{1}{2}} S_{\mathrm{i}+\frac{1}{2}} \right) \right| + \langle c \rangle_{\mathrm{i}} \left(\frac{S_{\mathrm{i}-\frac{1}{2}} + S_{\mathrm{i}+\frac{1}{2}}}{2} \right) \tag{5.48}$$

for the i-direction. The result for the j and k-direction is obtained in a similar fashion. Since time-accuracy of the solution is not relevant, a global time-step size is not required. Therefore, the magnitude of the time-step can be different for each control volume.

Runge-Kutta time-integration is used to determine an intermediate flow solution, sufficiently close to the final solution, such that Newton's method can be used to obtain the final fully converged flow solution.

To determine if an initial guess is sufficiently close, can be a tedious and very time consuming task. Therefore, a more practical approach is adopted here. In this approach the reduction of the ℓ^2-norm of the residual relative to its initial value is considered. When this ratio is below a certain threshold, the solution procedure is switched to Newton's method.

5.6.2 Newton's method

Once a suitable initial guess has been obtained, Newton's method is used to find the solution of the equations for steady flow. Consider equation (5.12), the semi-discrete conservation equation for a single control volume. For each control volume in the flow domain, this equation can be expressed as

$$\frac{\mathrm{d}\boldsymbol{\mathcal{U}}}{\mathrm{d}t} = \boldsymbol{\mathcal{R}}, \tag{5.49}$$

where $\boldsymbol{\mathcal{R}} = (\boldsymbol{R}_0, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_{\mathrm{N}})^T$. Note, that the expression for the residual, given by equation (5.45), is only valid for field control volumes. For a fringe control volume $\mathrm{a} \in \mathbb{N}_0$, the residual is defined as

$$\boldsymbol{R}_{\mathrm{a}} := \langle \boldsymbol{U} \rangle_{\mathrm{a}} - \sum_{\mathrm{m}=0}^{\mathrm{N_F}-1} \omega_{\mathrm{m}} \mathbb{U}_{\mathrm{m}} \equiv \boldsymbol{0}, \tag{5.50}$$

based on equation (5.44), which is actually not really a true residual.

For a steady-state solution $\boldsymbol{\mathcal{U}}^*$, the time derivative vanishes and the residual vector must therefore equal the null vector. Based on this observation, it can be derived from a first-order Taylor series expansion of $\boldsymbol{\mathcal{R}}$ around state $\boldsymbol{\mathcal{U}}^{\mathrm{n}}$ that the following equation must hold

$$\left.\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\mathcal{U}}}\right|_{\boldsymbol{\mathcal{U}}^{\mathrm{n}}} \left(\boldsymbol{\mathcal{U}}^{\mathrm{n}+1} - \boldsymbol{\mathcal{U}}^{\mathrm{n}}\right) = -\boldsymbol{\mathcal{R}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right) + \boldsymbol{\mathcal{O}}\left(\left(\boldsymbol{\mathcal{U}}^* - \boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)^2\right). \tag{5.51}$$

Neglecting the higher-order terms, the resulting system of linear equations is solved iteratively using a Krylov subspace method. An iterative method is used, because the dimensions of a matrix resulting from the accurate spatial discretization of the domain for a 3D flow problem are such that the computational cost — both in terms of memory as well as in terms of the number of floating-point operations required — is prohibitively large if a direct method would be used. Details on the solution strategy follow in the subsequent subsections.

Krylov subspace methods

Most iterative methods for the solution of a system of linear equations:

$$\underline{\underline{A}}\boldsymbol{x} = \boldsymbol{b},$$

with $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^{\mathrm{n}}$ and $\mathrm{n} \in \mathbb{N}_1$, can be considered as a projection method. In a projection method an approximation $\boldsymbol{x}'$ to the exact solution is determined. Consider $\underline{\underline{A}}$ to be an $\mathrm{n} \times \mathrm{n}$ matrix and both $\mathbb{K}$ and $\mathbb{L}$ to be m-dimensional subspaces of $\mathbb{R}^{\mathrm{n}}$. For an initial guess $\boldsymbol{x}_0 \in \mathbb{R}^{\mathrm{n}}$ a projection method finds an approximate solution

$$\boldsymbol{x}' \in \boldsymbol{x}_0 + \mathbb{K} \quad : \quad \boldsymbol{r} \equiv \boldsymbol{b} - \underline{\underline{A}}\boldsymbol{x}' \ \perp \ \mathbb{L}, \tag{5.52}$$

i.e. $\boldsymbol{x}'$ is an element of the affine subspace $\boldsymbol{x}_0 + \mathbb{K}$, such that the resulting residual is orthogonal to subspace $\mathbb{L}$. The characteristics of the projection method depend on the particular choice for subspaces $\mathbb{K}$ and $\mathbb{L}$.

In Krylov subspace methods the definition of $\mathbb{K}$ and $\mathbb{L}$ is based on a so-called Krylov subspace. A Krylov subspace of dimension $\mathrm{m} \in \mathbb{N}_1$ for matrix $\underline{\underline{A}}$ and vector $\boldsymbol{r}$ is defined as

$$\mathcal{K}_{\mathsf{m}}\left(\underline{\underline{A}}, \boldsymbol{r}\right) := \operatorname{span}\left\{\boldsymbol{r}, \underline{\underline{A}}\boldsymbol{r}, \ldots, \underline{\underline{A}}^{\mathrm{m}-1}\boldsymbol{r}\right\} \quad \text{for} \quad \mathrm{m} \geq 3. \tag{5.53}$$

The Krylov subspace method considered in this research is called the generalized minimal residual method [151], GMRES for short. The GMRES method defines the subspaces for the projection method as

$$\mathbb{K} = \mathcal{K}_{\mathsf{m}} \qquad \text{and} \qquad \mathbb{L} = \underline{\underline{A}}\mathcal{K}_{\mathsf{m}}.$$

This choice makes sure that the approximate solution $\boldsymbol{x}'$ satisfying the Galerkin-Petrov condition of equation (5.52) minimizes the ℓ^2-norm of the residual vector $\boldsymbol{r}$ [150]. Instead of using the vectors that define the Krylov subspace — which are in general not orthogonal — an orthonormal basis is constructed, for which

$$\boldsymbol{q}_0 = \frac{\boldsymbol{r}_0}{||\boldsymbol{r}_0||_2}$$

is used as the first basis vector. Every subsequent orthonormal basis vector is computed by means of a Gram-Schmidt process or a similar orthogonalization procedure. Next, denote by $\underline{\underline{Q}}_\mathrm{m}$ the $\mathrm{n} \times \mathrm{m}$ matrix of which the column vectors are $\boldsymbol{q}_0, \ldots, \boldsymbol{q}_{\mathrm{m}-1}$ and denote by $\underline{\underline{H}}_\mathrm{m}$ the $(\mathrm{m}+1) \times \mathrm{m}$ upper Hessenberg matrix for which the non-zero entries h_ij are defined by

$$h_\mathrm{ij} = \underline{\underline{A}}\boldsymbol{q}_\mathrm{j} \cdot \boldsymbol{q}_\mathrm{i}. \tag{5.54}$$

Then, the following equation holds

$$\underline{\underline{A}}\,\underline{\underline{Q}}_\mathrm{m} = \underline{\underline{Q}}_{\mathrm{m}+1}\underline{\underline{H}}_\mathrm{m}. \tag{5.55}$$

Using this equation and that an arbitrary vector $\boldsymbol{x} \in \boldsymbol{x}_0 + \mathcal{K}_\mathrm{m}$ can be expressed as a linear combination of the basis vectors of the subspace, the residual can be written as

$$\boldsymbol{r}\left(\boldsymbol{u}\right) = \underline{\underline{Q}}_{\mathrm{m}+1}\left(||\boldsymbol{r}_0||_2\,\boldsymbol{e}_0 - \underline{\underline{H}}_\mathrm{m}\boldsymbol{u}_\mathrm{m}\right), \tag{5.56}$$

where $\boldsymbol{u}_\mathrm{m} \in \mathbb{R}^\mathrm{m}$ is a vector that minimizes the ℓ^2-norm of $\boldsymbol{r}$ and $\boldsymbol{e}_0 \in \mathbb{R}^{\mathrm{m}+1}$ represents the first column vector of the identity matrix. Since $\underline{\underline{Q}}_{\mathrm{m}+1}$ is constructed of orthonormal vectors, the problem of minimizing the ℓ^2-norm of equation (5.56) can be reduced to solving an $(\mathrm{m}+1) \times \mathrm{m}$ least-square problem for $\boldsymbol{u}_\mathrm{m}$. Once the result $\boldsymbol{u}_\mathrm{m}$ is obtained, the approximate solution $\boldsymbol{x}'$ is found by evaluating

$$\boldsymbol{x}' = \boldsymbol{x}_0 + \underline{\underline{Q}}_\mathrm{m}\boldsymbol{u}_\mathrm{m}. \tag{5.57}$$

Since its introduction, the GMRES method has been widely used for solving systems of linear equations and various implementations are readily available. An example of such an implementation is available in the Portable Extensible Toolkit for Scientific Computing (PETSc) [12], developed at Argonne National Laboratory. The GMRES implementation of this particular numerical method is employed in the present research.

Jacobian matrix

The elements of the matrix resulting from the derivative of the residual of the flow equations with respect to the conserved flow variables — commonly referred to as the Jacobian matrix — are computed using the dual number method [60, 61], explained in section 1.6.4. Therefore, the functions used for computing the residual for each of the control volumes is called with a dual number data type. Since the dual number method provides numerically exact results, the resulting Jacobian matrix is also exact to machine precision. A more elaborate description on the construction of the Jacobian matrix is given in the next chapter, in section 6.4, starting on page 129.

The Jacobian matrix can be computed based on the second-order discretization of the convective flux, but also based on the first-order discretization — from here on referred to as first-order Jacobian matrix. The latter approach is useful for the construction of the preconditioning matrix, which is employed to improve the convergence characteristics of the Krylov subspace method.

Preconditioning

The rate of convergence of a Krylov subspace method depends on the condition number of the matrix [40]. For a Jacobian matrix, computed based on the second-order discretization of the convective flux, this condition number can be quite high — which is also apparent from the significant amount of scattering of the eigenvalues of this matrix in the left Argand diagram of figure 5.4 — resulting in a poor convergence of the iterative method. This problem can be alleviated by application of a preconditioning matrix, which transforms the original problem of equation (5.51) to

$$\underline{\underline{A}}\,\underline{\underline{P}}^{-1}\underline{\underline{P}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}+1}-\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)=-\boldsymbol{\mathcal{R}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right), \tag{5.58}$$

for right preconditioning. In this equation, matrix $\underline{\underline{A}}$ is used to designate the Jacobian matrix and matrix $\underline{\underline{P}}$ to denote the preconditioning matrix. Note, moreover, that the higher-order term present in the original equation is dropped here and that matrices and residual vector are evaluated for $\boldsymbol{\mathcal{U}}^{\mathrm{n}}$. By the application of right preconditioning, the original problem must be solved in two steps, first for an intermediate vector $\boldsymbol{y}$

$$\underline{\underline{A}}\,\underline{\underline{P}}^{-1}\boldsymbol{y}=-\boldsymbol{\mathcal{R}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right). \tag{5.59}$$

The final result is then obtained by solving

$$\underline{\underline{P}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}+1}-\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)=\boldsymbol{y}. \tag{5.60}$$

Note, that for a right preconditioned GMRES method, the intermediate vector $\boldsymbol{y}$ does not need to be obtained explicitly. In that case it is sufficient to construct an orthonormal basis for the Krylov subspace $\mathcal{K}_{\mathsf{m}}\left(\underline{\underline{A}}\,\underline{\underline{P}}^{-1},-\boldsymbol{\mathcal{R}}\right)$ — using an initial guess for the increment in $\boldsymbol{\mathcal{U}}$ to the next pseudo-time level equal to the null vector. Once the corresponding orthonormal basis is obtained and vector $\boldsymbol{u}_{\mathsf{m}}$ is found, which minimizes the ℓ^2-norm of the GMRES residual, the update for the conserved variables of the flow solution is computed according to

$$\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}+1}-\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)=\underline{\underline{P}}^{-1}\underline{\underline{Q}}_{\mathsf{m}}\boldsymbol{u}_{\mathsf{m}}. \tag{5.61}$$

A carefully constructed preconditioning matrix clusters the eigenvalues of the matrix, resulting in a reduction of the condition number of the matrix and an improvement of the convergence of the iterative method. The challenge is to construct a preconditioning matrix that realizes this clustering of the eigenvalues, while making sure that computing the inverse of the preconditioning matrix is computationally less expensive than computing the inverse of the original Jacobian matrix. Different strategies have been devised for the construction of a suitable preconditioning matrix. In the present research an incomplete lower-upper (ILU) decomposition is used.

The lower-upper decomposition of a matrix refers to the method used to construct a lower and an upper triangular matrix for which the product is equal to the original matrix. For an ILU(k) decomposition, a lower and upper triangular matrix are also constructed. However, in this case, the extent of fill-in — relative to the sparsity pattern of the original matrix — is limited to a certain level $\mathrm{k}\in\mathbb{N}_0$. This level is related to the number of steps to be performed in a Gaussian elimination process [150]; for ILU(0), the sparsity pattern of the triangular matrices resulting from the ILU decomposition correspond to the sparsity pattern of the upper and lower triangular part of the original matrix.

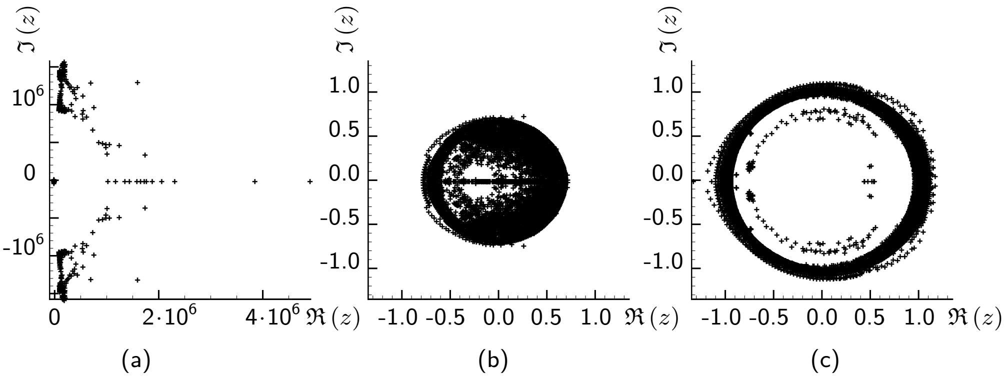


Figure 5.4: The eigenvalues computed for different matrices presented in an Argand diagram. The matrices have been computed for the flow about a swept wing subject to transonic flow conditions. The flow domain has been discretized with a single block grid consisting of 64 × 44 × 40 control volumes, the dimension of the resulting matrix equals 563 200 × 563 200. Value between the square brackets represents the number of distinct Eigenvalues depicted. (a) Jacobian matrix based on the second-order discretization of the convective flux [494]; (b) Same Jacobian matrix with ILU(0) preconditioning based on the first-order discretization of the convective flux [9162]; (c) Same Jacobian matrix with ILU(0) preconditioning based on the second-order discretization of the convective flux [8462].

An ILU decomposition can be performed for the Jacobian matrix, computed based on the second-order discretization of the convective flux. Alternatively, the first-order Jacobian matrix can be used for this purpose. Numerical experiments have indicated that the latter approach results in better convergence characteristics of the iterative method. Similar behaviour has also been observed by McHugh et al. [119]. Further investigation of the eigenvalues shows that this can be attributed to the preconditioning matrix, based on the first-order Jacobian matrix, being more successful in clustering the eigenvalues, as can be seen in figure 5.4. Therefore, the first-order Jacobian matrix is used for the construction of the preconditioning matrix. Since the first-order Jacobian matrix is more sparse, this approach also requires less memory and the construction takes less time; which both are desirable properties for the solution method.

Solution strategy

First, the flow solution for the current pseudo-time level is used to compute the Jacobian matrix based on both the second-order discretization of the convective flux, as well as on the first-order discretization. The first-order Jacobian matrix is used for constructing the preconditioning matrix and the second-order Jacobian matrix is required for the system of linear equations itself. Subsequently, the system of linear equations (5.51) is solved by means of a restarted GMRES method. Restarted means that the maximum dimension of the Krylov subspace is chosen to be smaller than the dimension of the system of linear equations for which it is used to solve; this approach reduces the memory requirements of the method. The dimension used in the present work is 200. Initial guess x_0 is taken equal to the null vector. Moreover, the required convergence of the GMRES iterations is made dependent on the number of Newton iterations, $\bar{\mathrm{n}} \in \mathbb{N}_0$, performed:

$$\left|\left|\underline{\underline{A}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}+1}-\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)+\boldsymbol{\mathcal{R}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)\right|\right|_2 \leq \eta_{\bar{\mathrm{n}}}\left|\left|\boldsymbol{\mathcal{R}}\left(\boldsymbol{\mathcal{U}}^{\mathrm{n}}\right)\right|\right|_2, \tag{5.62}$$

where $\eta_{\bar{\mathrm{n}}}$ controls the required accuracy for the update of the solution, starting with $\eta_0 = 0.5$. Note, that limiting the required convergence is permitted because of the monotonic convergence behaviour of the GMRES method. Subsequent values of $\eta_{\bar{\mathrm{n}}}$ are provided by the following relation [56]

$$\eta_{\bar{\mathrm{n}}} = \max\left\{\eta_{\mathsf{min}}, \eta_{\bar{\mathrm{n}}-1}^{\frac{1}{2}\left(1+\sqrt{5}\right)}\right\} \in \mathbb{R} \quad \text{for } \bar{\mathrm{n}} \geq 1, \tag{5.63}$$

where parameter η_{min} can be specified by the user; a value of 0.01 is a suitable choice [82]. This approach is taken to prevent the solution of the system of linear equations, that determines the increment in the flow solution from one pseudo-time-step to the next, to be unnecessarily accurate when the intermediate solution is still relatively far from the final solution. In this way faster convergence, in terms of CPU time [82], is achieved. Moreover, it has been observed that this method is more robust than using a fully converged solution of the system of linear equations, especially for the early iterations. Because limiting the required convergence of the GMRES iterations effectively acts as a kind of under-relaxation.

Once the convergence criterion is satisfied, equation (5.61) is used to compute the required update for the flow solution. Subsequently, the flow solution is updated and the residual vector is recomputed, using the flow solution for the new pseudo-time level. Note, that the flow solution for the fringe control volumes is updated by means of interpolation of the updated flow solution of the donor control volumes. The solution of the system of linear equations is not used for this purpose, because this approach can lead to an inconsistent flow solution, since the system is not solved to full precision. When the ℓ^2-norm of the residual vector is below a certain threshold, the flow solution is considered converged. Otherwise, the Jacobian matrix for both the first and second-order discretization are also recomputed and the solution procedure is repeated until the convergence criteria are met.

From the aforementioned description of Krylov subspace methods, it is apparent that these methods only require matrix vector products during the iterative solution procedure — equation (5.54). It is therefore also possible to use a Fréchet derivative [4] to directly compute the matrix vector product. This approach avoids the need to explicitly compute and store the second-order Jacobian matrix, reducing the memory requirements of the method. Moreover, by exploiting dual numbers for the evaluation of the residual function, the Fréchet derivative still yields the exact result. However, this approach only outperforms the method in which the second-order Jacobian matrix is computed explicitly, in terms of CPU-time, if the number of GMRES iterations required to reach convergence is very limited. It was found that this situation does generally not apply. Another reason for using a matrix-free approach is the limited memory requirements of the method, compared to storing the complete matrix. However, storing the matrix is already required for solving the adjoint equations[8], used for efficiently computing the

---

[8] Note, that it is possible to use a matrix-free approach for the adjoint equations as well, if reverse mode algorithmic differentiation is used. It is however not possible to use the dual number method for this purpose, since the dual number method only provides the forward mode, see section 6.1.

gradients. Therefore, the memory requirements of storing the matrix need to be met anyway. For these two reasons, the so-called matrix-free approach is not used in the present research.

5.7 Non-inertial frame of reference

Consider an isolated wind turbine rotor, rotating at a constant angular velocity $\boldsymbol{\Omega}$ in an infinite domain, with the rotor plane perpendicular to the uniform steady wind. In an inertial frame of reference, this flow configuration is unsteady. However, if the same configuration is considered in a frame of reference co-rotating with the wind turbine rotor, a steady situation is encountered[9]. From an optimization and computational point of view, it is advantageous to be able to obtain a steady flow solution. For that purpose, the governing equations considered in a co-rotating frame of reference are presented here.

Variables, considered in the co-rotating frame of reference are denoted by the subscript 'rot'. Furthermore, it is assumed that the origin of both reference frames coincide. The velocity in the inertial frame of reference is then related to the velocity in the co-rotating frame of reference by

$$\boldsymbol{u} = \boldsymbol{u}_{\text{rot}} + \boldsymbol{\Omega} \times \boldsymbol{x}. \tag{5.64}$$

The time derivative of an arbitrary time dependent vector quantity $\boldsymbol{f}(t)$ considered in the rotating frame of reference can be expressed as

$$\frac{\mathrm{d}\boldsymbol{f}(t)}{\mathrm{d}t} = \left(\frac{\mathrm{d}\boldsymbol{f}(t)}{\mathrm{d}t}\right)_{\text{rot}} + \boldsymbol{\Omega} \times \boldsymbol{f}(t)\,. \tag{5.65}$$

Using these two relations, the conservation equations, considered in an inertial frame of reference, expressed in terms of absolute quantities, read [197, 199]

$$\frac{\partial}{\partial t} \int\limits_{V(t)} \boldsymbol{U}\,\mathrm{d}V + \int\limits_{\partial V(t)} \underline{\underline{F}}_{\text{rot}}(\boldsymbol{U}) \cdot \boldsymbol{n}\,\mathrm{d}S - \int\limits_{V(t)} \boldsymbol{Q}(\boldsymbol{U})\,\mathrm{d}V = \boldsymbol{0}, \tag{5.66}$$

where the co-rotating motion is taken into account in the flux tensor $\underline{\underline{F}}_{\text{rot}}(\boldsymbol{U})$:

$$\underline{\underline{F}}_{\text{rot}}(\boldsymbol{U}) := \left(\rho\boldsymbol{u}_{\text{rot}}^T, \rho\boldsymbol{u}\boldsymbol{u}_{\text{rot}}^T + p\underline{\underline{I}}, p\boldsymbol{u}^T + \rho e_{\text{t}}\boldsymbol{u}_{\text{rot}}^T\right)^T \text{ with } \boldsymbol{u}_{\text{rot}} = \boldsymbol{u} - (\boldsymbol{\Omega} \times \boldsymbol{x})\,. \tag{5.67}$$

The volumetric source term in equation (5.66) reads

$$\boldsymbol{Q}(\boldsymbol{U}) = \begin{pmatrix} 0 \\ -\boldsymbol{\Omega} \times [\rho\boldsymbol{u}] \\ 0 \end{pmatrix}. \tag{5.68}$$

Transforming the governing equations to a rotating frame of reference results in the introduction of a body force in the momentum equations. This force term represents the combined effect of the centrifugal force and Coriolis force, which are known to arise when a physical system is considered in a rotating frame of reference. Note, that these forces do not contribute to the energy equation, because the forces are perpendicular to the direction of motion and do therefore not perform any work. Moreover, note that the control volumes are stationary, considered in the co-rotating frame of reference and since they are assumed rigid, the term in equation (5.3) involving $\boldsymbol{u}_{\partial\text{V}}$ is not present in equation (5.66).

---

[9] Assuming the flow field does not exhibit any unsteady effects like flow separation or vortex shedding.

5.7.1 Geometric conservation law

Solving the discretized system of equations presented in equation (5.66), one must make sure that the so-called geometric conservation law is satisfied [184, 207]. Otherwise, it may occur that for a flow configuration that should result in a steady uniform flow, does not yield the expected result. Moreover, for a more arbitrary flow configuration that involves moving grids, not satisfying the geometric conservation law (GCL) can result in hampering the convergence of the solution method and loss of accuracy of the solution.

For an arbitrary control volume, rotating with a constant angular velocity around an axis through the origin — where the rotation is specified by $\boldsymbol{\Omega}$ — for the continous case, the following equation can be derived from equation (5.66):

$$\frac{\partial}{\partial t} \int\limits_{V(t)} \mathrm{d}V \equiv \int\limits_{\partial V(t)} (\boldsymbol{\Omega} \times \boldsymbol{x}) \cdot \boldsymbol{n} \, \mathrm{d}S. \tag{5.69}$$

Recall, that Euler equations in semi-discrete form for control volume a reads

$$V_{\mathrm{a}} \frac{\mathrm{d} \langle \boldsymbol{U} \rangle_{\mathrm{a}}}{\mathrm{d}t} + \sum_{\mathrm{m}=0}^{\mathrm{m}=5} \boldsymbol{\mathcal{F}} (\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}, \boldsymbol{n}_{\mathrm{m}}) \, S_{\mathrm{m}} = \boldsymbol{0}. \tag{5.12}$$

Using the definition of the control-volume-averaged value, the term involving a time derivative in equation (5.66) can be expressed as

$$\frac{\partial}{\partial t} (V_{\mathrm{a}} \langle \boldsymbol{U} \rangle_{\mathrm{a}}) = V_{\mathrm{a}} \frac{\mathrm{d} \langle \boldsymbol{U} \rangle_{\mathrm{a}}}{\mathrm{d}t} + \langle \boldsymbol{U} \rangle_{\mathrm{a}} \frac{\partial V_{\mathrm{a}}}{\partial t}. \tag{5.70}$$

Note, that the second term in equation (5.70) is not accounted for in the semi-discrete representation of the Euler equations. To account for this discrepancy, an additional source term, originating from the equivalence relation of equation (5.69), is added to the semi-discrete equations (5.12). Including the source term due to rotation, the equation reads

$$\begin{aligned} & V_{\mathrm{a}} \frac{\mathrm{d} \langle \boldsymbol{U} \rangle_{\mathrm{a}}}{\mathrm{d}t} + \sum_{\mathrm{m}=0}^{\mathrm{m}=5} \boldsymbol{\mathcal{F}} (\boldsymbol{U}_{\mathsf{L}}, \boldsymbol{U}_{\mathsf{R}}, \boldsymbol{n}_{\mathrm{m}}) \, S_{\mathrm{m}} \\ & \qquad - \underbrace{V_{\mathrm{a}} \boldsymbol{Q} (\langle \boldsymbol{U} \rangle_{\mathrm{a}})}_{\text{rotation source term}} \overbrace{- \langle \boldsymbol{U} \rangle_{\mathrm{a}} \sum_{\mathrm{m}=0}^{\mathrm{m}=5} (\boldsymbol{\Omega} \times \boldsymbol{x}) \cdot \boldsymbol{n} S_{\mathrm{m}}}^{\text{GCL source term}} = \boldsymbol{0}. \end{aligned} \tag{5.71}$$

By adding the GCL source term, an artificial source of error, originating from discretization of the governing equations, is eliminated. The result is that a uniform flow solution will remain uniform, when a simulation is performed on a moving grid considered in a moving frame of reference. Note, that analytically this additional term is equal to zero, for the case of the motion of a rigid control volume in a uniform flow field.

5.8 Verification

In the verification of the flow solution method it is considered whether the implementation of the method corresponds with the mathematical model the method is based on. Verification can either be achieved by: (i) obtaining a numerical solution for a flow configuration for which an analytical solution exists; (ii) comparing the numerical result of the present implementation with an accurate numerical result obtained with one or more implementations that have been verified to be correct. The latter approach is taken here. First, it is verified if the solution method provides correct results for a single block discretization of a 2D flow domain around a NACA 0012 aerofoil. To this end, an assessment is made of the order of convergence of the spatial discretization. Subsequently, the influence of using composite overset grids for the discretization of the same flow configuration is investigated.

5.8.1 Spatial order of convergence

One way of assessing a flow solution is by the evaluation of a discrete functional ϕ which quantifies the flow solution. Examples of such functionals include the lift and drag coefficient, respectively denoted by c_l and c_d for 2D flow. These discrete functionals can also be employed to assess the spatial order of convergence of the flow solution method — and to obtain an estimate for the limiting case of the grid resolution going to infinity. For a grid, characterized by typical control volume size h, an estimate of the functional value of infinite resolution ϕ^* can be obtained using Richardson extrapolation [145]:

$$\phi^* = \phi\left(h\right) + Ch^{\bar{p}} + \mathcal{O}\left(h^{\bar{p}+1}\right), \tag{5.72}$$

where $\bar{p}$ is the spatial order of convergence and C a proportionality constant independent of the characteristic size. The three unknowns of equation (5.72) — i.e. $\bar{p}$, C and ϕ^* — can be determined, if the functional has been evaluated for three different values of h. Note, that this relation assumes that: (i) the discretization error is the leading order error, (ii) the grid aspect ratio is the same for all grids [154] and (iii) the discretization error can be expressed as $Ch^{\bar{p}}$ — which is only true for sufficiently small values of h, i.e. when the method operates within its asymptotic range [29]. For the family of grids considered in the present research, the control volume sizes are related by

$$h_\mathsf{c} = 2h_\mathsf{m} = 4h_\mathsf{f},$$

for the coarse, medium and fine grid, respectively. This family of grids is constructed by first generating the grid for the finest resolution. Subsequent levels are obtained by removing every other vertex of the grid in both directions.

5.8.2 Flow configuration

The flow configuration considered to investigate the spatial order of convergence, is the inviscid subcritical flow around the NACA 0012 aerofoil. In contrast to the original geometry, which has a blunt trailing edge, the aerofoil geometry is extended to have a sharp trailing edge. This modification was made to accommodate the generation of a single block high quality grid and to be able to compare the results with results presented in the literature [194, 195], for which the same approach was taken.

Following the procedure, described by Vassberg and Jameson [195], an algebraic grid, with an O-grid topology, is generated around the extended NACA 0012 aerofoil. The

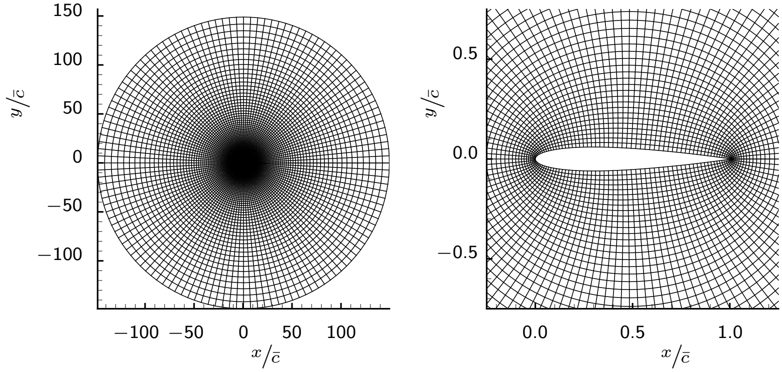


Figure 5.5: Flow domain used for studying the spatial convergence characteristics of the flow solution method. The O-grid depicted consists of 128 × 128 control volumes. Left picture shows detail of the grid around the aerofoil. Dimensions are scaled by the chord length $\bar{c}$.

algebraic grid generation method uses the Kármán-Trefftz conformal transformation [122]

$$\frac{\zeta - \zeta_1}{\zeta - \zeta_2} = \left(\frac{z - z_1}{z - z_2}\right)^P, \quad P = \frac{\pi}{2\pi - \tau_{\text{te}}}, \tag{5.73}$$

to map the extended NACA 0012 aerofoil to a near circle. In this equation τ_{te} is the trailing edge angle of the aerofoil. For the other mapping parameters z_1, z_2, ζ_1, ζ_2 and for the full description of the grid generation procedure, see appendix B. The nice feature of a conformal mapping is that angles between grid lines are preserved. Therefore, by generating an orthogonal[10] grid in the mapped space, the resulting grid obtained by performing the inverse transformation is also orthogonal. Figure 5.5 shows the grid and a close-up of the aerofoil for a resolution of 128 × 128 control volumes. The far-field is situated at approximately 150 chord lengths away from the aerofoil surface.

The investigation is performed for a free-stream Mach number of $\mathrm{M}_\infty = 0.5$ and angle of attack $\alpha = 1.25°$. The finest grid consists of 4096 × 4096 control volumes. This grid has been coarsened up to 7 times, resulting in a coarse grid containing 32 × 32 control volumes. The force on the aerofoil is computed by integrating the pressure over the surface, employing the direction of the local outward pointing unit surface normal vector $\boldsymbol{n}$. For a 2D geometry the surface integral reduces to a line integral, which reads

$$\boldsymbol{F}_{\text{a}} = -\oint p(s)\,\boldsymbol{n}(s)\left|\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s}\right|\mathrm{d}s, \tag{5.74}$$

where $\boldsymbol{F}_{\text{a}} \in \mathbb{R}^3$ is the force per unit span exerted by the flow on the aerofoil, $p(s)$ is the local pressure as a function of parametric variable s, which follows the closed contour of the aerofoil, see figure 5.6 for a schematic representation. For the discretized aerofoil surface, this integral is computed using numerical quadrature by means of the midpoint rule. The resulting force can subsequently be used to evaluate lift and drag coefficients,

---

[10] Actually nearly orthogonal, since the aerofoil is not a perfect circle in the mapped space.

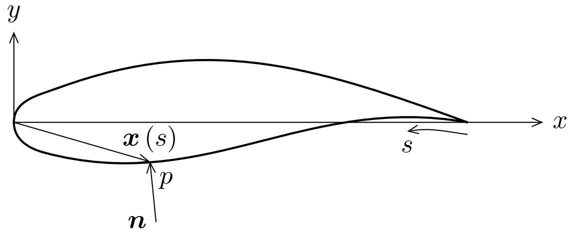


Figure 5.6: Schematic representation of an aerofoil with the parameters required for computing the force exerted on the aerofoil indicated.

which are respectively defined as

$$c_{\mathsf{l}} := \frac{\boldsymbol{F}_{\mathsf{a}} \cdot \boldsymbol{k}_{\perp}}{\frac{1}{2}\rho_{\infty} U_{\infty}^{2} \bar{c}}, \tag{5.75}$$

$$c_{\mathsf{d}} := \frac{\boldsymbol{F}_{\mathsf{a}} \cdot \boldsymbol{k}_{\parallel}}{\frac{1}{2}\rho_{\infty} U_{\infty}^{2} \bar{c}}. \tag{5.76}$$

In these equations ρ_{∞} denotes the free-stream density, U_{∞} is the magnitude of the free-stream velocity and $\bar{c}$ is the chord length of the aerofoil. Unit vectors $\boldsymbol{k}_{\perp} \in \mathbb{R}^3$ and $\boldsymbol{k}_{\parallel} \in \mathbb{R}^3$ point in the direction perpendicular to the free-stream direction and in the direction of the free-stream direction, respectively. Moreover, note that in this specific case the original chord length of 1.0 $[\mathrm{m}]$ is used instead of the actual chord length of the extended NACA 0012 aerofoil, which is slightly larger. The pitching moment $M_{\mathsf{z},\frac{\bar{c}}{4}}$ per unit span around the z-axis with respect to the quarter chord length point is computed by evaluating

$$M_{\mathsf{z},\frac{\bar{c}}{4}} = \left(\oint \left[\boldsymbol{x}(s) - \left(\frac{\bar{c}}{4} \right) \boldsymbol{e}_{\mathsf{x}} \right] \times p(s)\, \boldsymbol{n}(s) \left| \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} \right| \mathrm{d}s \right) \cdot \boldsymbol{e}_{\mathsf{z}}; \tag{5.77}$$

again by means of the midpoint quadrature rule. In this equation $\boldsymbol{e}_{\mathsf{x}} \in \mathbb{R}^3$ and $\boldsymbol{e}_{\mathsf{z}} \in \mathbb{R}^3$ represent the unit vector in x and z-direction, respectively. The dimensionless representation of the pitching moment reads

$$c_{\mathsf{m}} := \frac{M_{\mathsf{z},\frac{\bar{c}}{4}}}{\frac{1}{2}\rho_{\infty} U_{\infty}^{2} \bar{c}^{2}}. \tag{5.78}$$

5.8.3 Results for single block grid

In section 5.3 two different methods have been discussed for handling the convective flux. Moreover, for the upwind approach, different methods can be used for performing the reconstruction of the state at the interface. Therefore, to verify these different methods, three different situations are considered: (i) Roe's approximate Riemann solver with linear reconstruction about the centre of mass of the control volumes; (ii) Roe's approximate Riemann solver with linear reconstruction assuming a uniform Cartesian grid; (iii) central

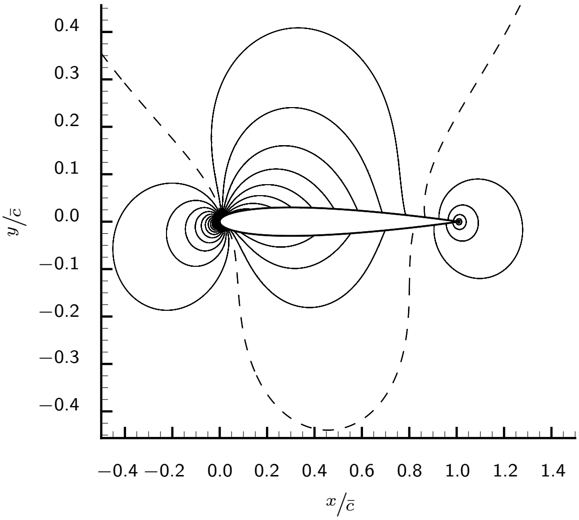


Figure 5.7: Subcritical inviscid flow about an extended NACA 0012 aerofoil for $\alpha = 1.25^\circ$ at $\mathrm{M}_\infty = 0.5$. Flow domain discretized by means of a composite overset grid; grid dimensions are listed in table 5.7 on page 110. Iso-Mach contours: lower level $\mathrm{M} = 0.0037$, upper level $\mathrm{M} = 0.6637$, increment $\Delta\,\mathrm{M} = 0.02$, dashed contour line $\mathrm{M} = 0.5037$.

discretization of the flux using the JST scheme. Moreover, an additional series of flow simulations has been performed with a considerably larger distance between the surface of the aerofoil and the far-field boundary, in order to investigate the possible influence of the distance of the far-field boundary on the spatial convergence behaviour. The Mach number distribution of the flow solution for the flow configuration considered is shown in figure 5.7[11].

Upwind discretization with linear reconstruction about the centre of mass

The results of computing the force coefficients and moment coefficient for the different grid resolutions, obtained for flow simulations performed with an upwind convective flux discretization, using Roe's approximate Riemann solver, with linear reconstruction about the centre of mass of the control volumes, are presented in table 5.1. For the linear reconstruction the κ-scheme has been used, with $\hat{\kappa} = {}^1\!/_3$. For all the results, the flow residual was converged up to machine accuracy. The table also lists the order of convergence for the different coefficients — computed based on the three finest grids[12] — and the expected value if an infinite grid resolution would have been used. The differences between the extrapolated value and the values listed in table 5.1 are plotted in figure 5.8, for each of the three dimensionless coefficients. Each of the graphs also includes a trend line which illustrates the order of convergence observed.

---

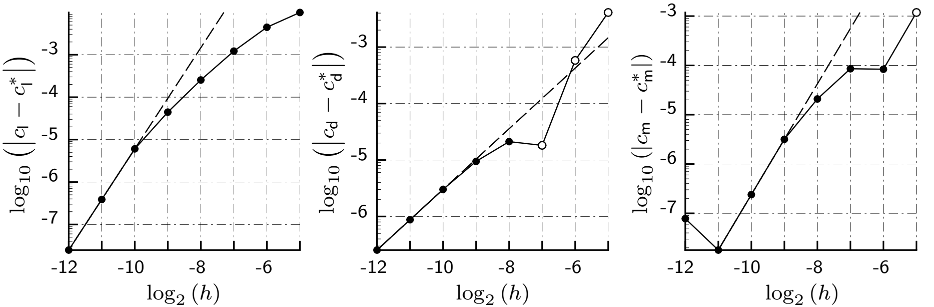
[11] Note, that the flow solution depicted in the figure has actually been obtained using a composite overset grid. However, for the grid resolution used, no visual differences from the finest single block grid result can be observed for the scale at which the results have been presented in the figure.

[12] In some cases non-monotonic convergence behaviour was observed. In those cases the alternative approach for computing the convergence order is mentioned in the corresponding caption.

Table 5.1: Results obtained on single block O-grid with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 1/3$; far-field at 150 $\bar{c}$.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.79774289 \cdot 10^{-1}$	$9.34057594 \cdot 10^{-6}$	$2.26213760 \cdot 10^{-3}$
2048	$1.79779950 \cdot 10^{-1}$	$1.14818243 \cdot 10^{-5}$	$2.26235858 \cdot 10^{-3}$
4096	$1.79780313 \cdot 10^{-1}$	$1.21035183 \cdot 10^{-5}$	$2.26229847 \cdot 10^{-3}$
ϕ^*	$1.79780338 \cdot 10^{-1}$	$1.23578715 \cdot 10^{-5}$	$2.26237657 \cdot 10^{-3}$
$\bar{p}$	3.961	1.784	3.731 †

† Order of convergence estimated using result of the coarser grid with $\mathrm{N_c} = 512$, for the corresponding value of c_m see table C.1 on page 208.



$\left(\bar{p} = 3.961,\ c_\mathsf{l}^* \approx 1.7978 \cdot 10^{-1}\right)$ (a)

$\left(\bar{p} = 1.784,\ c_\mathsf{d}^* \approx 1.2358 \cdot 10^{-5}\right)$ (b)

$\left(\bar{p} = 3.731,\ c_\mathsf{m}^* \approx 2.2624 \cdot 10^{-3}\right)$ (c)

Figure 5.8: Spatial convergence of the present method for the aerodynamic coefficients: (a) lift; (b) drag; and (c) moment. For increasing number of control volumes $\mathrm{N_c}$; typical grid size $h \equiv 1/\mathrm{N_c}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.1. Open circles indicate that the result is larger than the estimate for $h = 0$; the settings used and more accurate values for the estimate for $h = 0$ are also found in table 5.1.

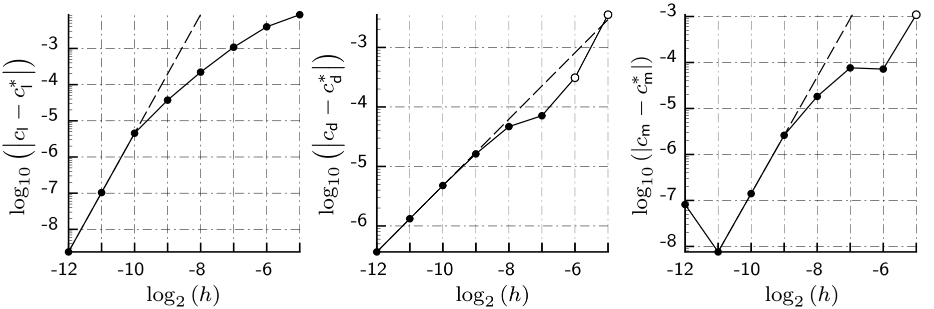
With the type of reconstruction employed, it should be possible to achieve a third-order spatial convergence. The observed order of convergence however, differs from the expected order. For the drag coefficient the order of convergence is lower than expected, which is probably caused by the contribution of the entropy production at the trailing edge to the drag. This entropy production being a first-order effect, is likely to cause a reduction in the spatial order of convergence. Moreover, the estimate for $h = 0$ for the drag is non-zero. As reported by Vassberg and Jameson, the non-zero drag is caused by not applying vortex-correction for the far-field boundary condition [195].

The order of convergence for the lift coefficient on the other hand, exceeds the expected value of three. The precise reason for this discrepancy is not clear. It might be due to the force coefficients being an integrated quantity, which can give rise to cancellation of errors [133]. It might, however, also be caused by the way the pressure

Table 5.2: Results obtained on single block O-grid with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volumes; linear reconstruction weighting factor $\hat{\kappa} = 1/3$; far-field at 150 $\bar{c}$; modified pressure boundary condition at solid wall.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.79775848 \cdot 10^{-1}$	$7.57986239 \cdot 10^{-6}$	$2.26224080 \cdot 10^{-3}$
2048	$1.79780283 \cdot 10^{-1}$	$1.10345247 \cdot 10^{-5}$	$2.26237412 \cdot 10^{-3}$
4096	$1.79780385 \cdot 10^{-1}$	$1.19908439 \cdot 10^{-5}$	$2.26230003 \cdot 10^{-3}$
ϕ^*	$1.79780387 \cdot 10^{-1}$	$1.23569050 \cdot 10^{-5}$	$2.26238171 \cdot 10^{-3}$
$\bar{p}$	5.449	1.853	4.213[†]

[†] Order of convergence estimated using result of the coarser grid with $\mathrm{N_c} = 512$, for the corresponding value of c_m see table C.2 on page 208.



$\left(\bar{p} = 5.449,\ c_\mathrm{l}^* \approx 1.7978 \cdot 10^{-1}\right)$ (a)

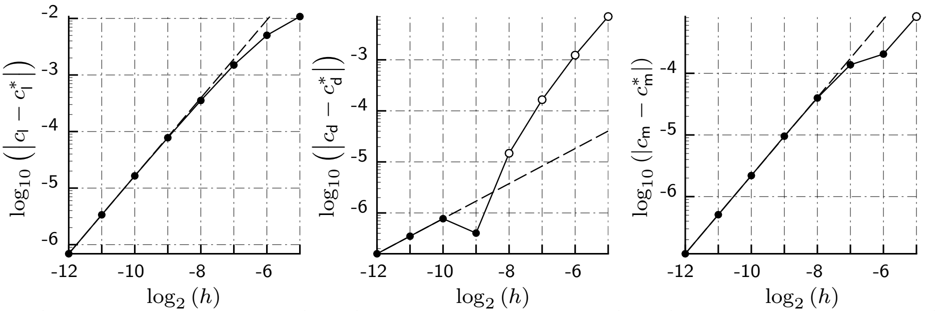
$\left(\bar{p} = 1.853,\ c_\mathrm{d}^* \approx 1.2357 \cdot 10^{-5}\right)$ (b)

$\left(\bar{p} = 4.213,\ c_\mathrm{m}^* \approx 2.2624 \cdot 10^{-3}\right)$ (c)

Figure 5.9: Spatial convergence of the present method for the aerodynamic coefficients: (a) lift; (b) drag; and (c) moment. For increasing number of control volumes $\mathrm{N_c}$; typical grid size $h \equiv 1/\mathrm{N_c}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.2. Open circles indicate that the result is larger than the estimate for $h = 0$; the settings used and more accurate values of the estimate for $h = 0$ are also found in table 5.2.

Table 5.3: Results obtained on single block O-grid with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 0$; far-field at 150 $\bar{c}$; modified boundary condition for the pressure at the solid wall.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.79763492 \cdot 10^{-1}$	$1.16340609 \cdot 10^{-5}$	$2.26003639 \cdot 10^{-3}$
2048	$1.79776578 \cdot 10^{-1}$	$1.20558493 \cdot 10^{-5}$	$2.26171927 \cdot 10^{-3}$
4096	$1.79779257 \cdot 10^{-1}$	$1.22470654 \cdot 10^{-5}$	$2.26211023 \cdot 10^{-3}$
ϕ^*	$1.79779947 \cdot 10^{-1}$	$1.24056429 \cdot 10^{-5}$	$2.26222855 \cdot 10^{-3}$
$\bar{p}$	2.288	1.141	2.106



$\left(\bar{p} = 2.288,\ c_\mathrm{l}^* \approx 1.7978 \cdot 10^{-1}\right)$ (a)

$\left(\bar{p} = 1.141,\ c_\mathrm{d}^* \approx 1.2406 \cdot 10^{-5}\right)$ (b)

$\left(\bar{p} = 2.106,\ c_\mathrm{m}^* \approx 2.2622 \cdot 10^{-3}\right)$ (c)

Figure 5.10: Spatial convergence of the present method for the aerodynamic coefficients: (a) lift; (b) drag; and (c) moment. For increasing number of control volumes $\mathrm{N_c}$; typical grid size $h \equiv {}^1\!/\mathrm{N_c}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.3. Open circles indicate that the result is larger than the estimate for $h = 0$; the settings used and more accurate values for the estimate for $h = 0$ are also found in table 5.3.

is handled in the solid wall boundary condition. This suggestion is based on the work presented by Destarac [51], who investigated the effect of the method used for handling the pressure in the solid wall boundary conditions on the spatial convergence behaviour of the drag coefficient. That research showed that performing linear extrapolation of the pressure causes an effect that has been qualified as anti-dissipative, resulting in a negative drag-coefficient. Considering that the order of magnitude of the reported effect is similar to the order of the digits significant for the spatial order of convergence of the lift coefficient, for the finest grid considered, the boundary condition effect could well be the cause of the apparent increase in the spatial order of convergence for the lift coefficient. Moreover, this boundary condition effect is probably also the cause for the non-monotonic convergence behaviour of the moment coefficient.

To check this assumption, also simulations have been performed with an implementation of the modified pressure boundary condition, i.e. extrapolation of the pressure to the halo control volume is done without taking into account the centre of mass of the two control volumes near the boundary and the halo control volume. The results of these simulations are presented in table 5.2 and figure 5.9 on page 103. Considering these results, the supposed effect is not observed. Changing the boundary condition even results in an increase of the observed order of convergence, with respect to the previous calculations.

Since the weighting factor $\hat{\kappa}$ — for the linear reconstruction of the state at the interface — affects the spatial order of convergence, its effect is also investigated. Table 5.3 presents the results of the series of flow simulations performed for $\hat{\kappa} = 0$ and the modified pressure boundary condition, the corresponding graphs are found in figure 5.10. For these settings, a well-behaved convergence behaviour is observed for all three dimensionless coefficients investigated for the three finest grids considered. Moreover, for the lift coefficient and the moment coefficient, the spatial order of convergence is in reasonable agreement with the second-order convergence expected. The order of convergence for the drag coefficient is slightly lower, probably for reasons previously discussed. When the graph that shows the convergence behaviour of the drag coefficient is considered more closely, some striking behaviour is observed. For the coarser grids, it appears that the most prominent contribution to the drag coefficient shows a third-order trend. Then, for the finer grids, the effect showing a third-order trend has significantly diminished and an effect that shows a first-order trend has the most prominent contribution.

Upwind discretization with linear reconstruction based on uniform grid assumption

The next series of flow simulations has been performed, again using an upwind convective flux discretization by means of Roe's approximate Riemann solver. However, now the linear reconstruction is performed assuming a uniform Cartesian grid. Therefore, the coordinates of both the centre of mass of the control volume as well as the centre of the face for which the flux is computed, do not need to be considered explicitly. Based on the experience from the preceding flow simulations, a weighting factor $\hat{\kappa} = 0$ and the modified pressure boundary condition are employed. The results of this investigation are presented in table 5.4. A graphical representation of the order of convergence and the differences between the results for finite grid resolutions and its limiting value at the continuum where $h = 0$ are depicted in figure 5.11 for the dimensionless coefficients computed.

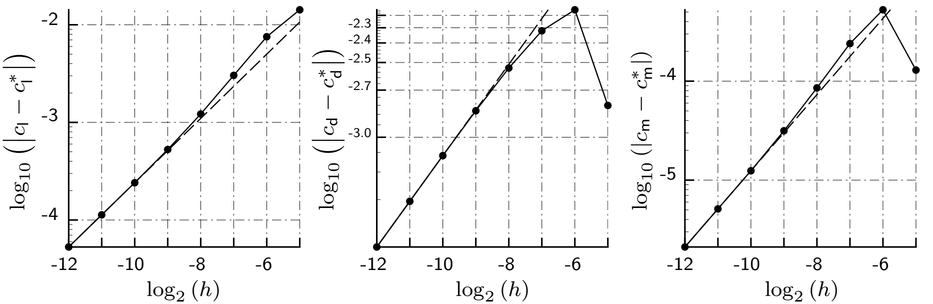
Considering these results, it is observed that using the uniform Cartesian grid assumption has a detrimental effect on the spatial order of convergence. This effect is likely caused by the fact that only for reconstruction about the centre of mass, the control-volume-averaged quantities are conserved [95]. Another, notable observation concerns the sign of the drag coefficient. Although the modified pressure boundary condition is used, to prevent the so-called anti-dissipative effect [51] from occurring, still a negative value is found for all grids considered. For the estimate of the exact value, however, a positive value is obtained.

Central discretization

For the series of flow simulations performed using the convective flux, discretized by means of the JST scheme, the results are presented in table 5.5 on page 107. The differences

Table 5.4: Results obtained on single block O-grid with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, assuming uniform Cartesian grid; linear reconstruction weighting factor $\hat{\kappa} = 0$; far-field at 150 $\bar{c}$; modified pressure boundary condition at solid wall.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.79535005 \cdot 10^{-1}$	$-7.45009630 \cdot 10^{-4}$	$2.24953016 \cdot 10^{-3}$
2048	$1.79663081 \cdot 10^{-1}$	$-3.71797592 \cdot 10^{-4}$	$2.25683221 \cdot 10^{-3}$
4096	$1.79723088 \cdot 10^{-1}$	$-1.81079366 \cdot 10^{-4}$	$2.25984132 \cdot 10^{-3}$
ϕ^*	$1.79775988 \cdot 10^{-1}$	$1.82339234 \cdot 10^{-5}$	$2.26195053 \cdot 10^{-3}$
$\bar{p}$	1.094	0.969	1.279



$\left(\bar{p} = 1.094,\ c_\mathsf{l}^* \approx 1.7978 \cdot 10^{-1}\right)$ (a)

$\left(\bar{p} = 0.969,\ c_\mathsf{d}^* \approx 1.8234 \cdot 10^{-5}\right)$ (b)

$\left(\bar{p} = 1.279,\ c_\mathsf{m}^* \approx 2.2620 \cdot 10^{-3}\right)$ (c)

Figure 5.11: Spatial convergence of the present method for the aerodynamic coefficients: (a) lift; (b) drag; and (c) moment. For increasing number of control volumes $\mathrm{N_c}$; typical grid size $h \equiv {}^{1}/_{\mathrm{N_c}}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.4. The settings used and more accurate values for the estimate for $h = 0$ are also found in table 5.4.

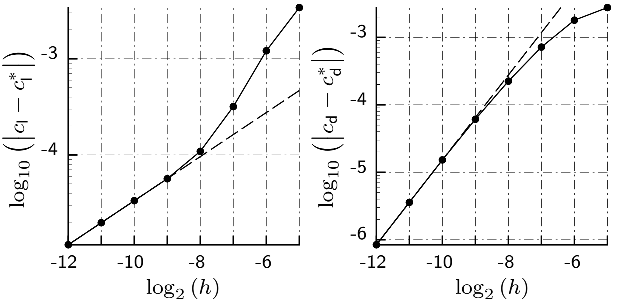
between the estimate for $h = 0$ — based on the observed order of convergence — and the results for the different grid resolutions are presented graphically in figure 5.12.

Based on the convective flux discretization employed, a second-order spatial order of convergence is expected. However, this order of convergence is not achieved for the lift coefficient. The order of convergence found does not even exceed one. The lower order of convergence found might be related to the uniform Cartesian grid assumption that was made in the derivation of the JST scheme. As observed in the preceding subsection, this assumption negatively affects the observed spatial order of convergence for the lift coefficient.

For the drag coefficient on the other hand, an observed spatial order of convergence very close to 2.0 is found. Considering the result for c_l, this is a surprising result, certainly because the entropy production at the trailing edge should also negatively affect the order of convergence for c_d, in a similar manner as the behaviour observed for the upwind discretization of the convective flux. A possible cause might be the occurrence of the

Table 5.5: Results obtained on single block O-grid with following settings: JST scheme to compute convective flux; dissipation coefficients used: $k_2 = 0.25$, $k_4 = 0.015625$; far-field at 150 $\bar{c}$; modified pressure boundary condition at solid wall. No estimate could be obtained for the spatial convergence order for c_m.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.79685217 \cdot 10^{-1}$	$-1.39285941 \cdot 10^{-6}$	$2.25727058 \cdot 10^{-3}$
2048	$1.79698923 \cdot 10^{-1}$	$1.02730972 \cdot 10^{-5}$	$2.25834029 \cdot 10^{-3}$
4096	$1.79707007 \cdot 10^{-1}$	$1.30144502 \cdot 10^{-5}$	$2.25959865 \cdot 10^{-3}$
ϕ^*	$1.79718633 \cdot 10^{-1}$	$1.38565062 \cdot 10^{-5}$	—
$\bar{p}$	0.762	2.089	—



$\left(\bar{p} = 0.762,\ c_\mathrm{l}^* \approx 1.7972 \cdot 10^{-1}\right)$ $\left(\bar{p} = 2.089,\ c_\mathrm{d}^* \approx 1.3857 \cdot 10^{-5}\right)$

(a) (b)

Figure 5.12: Spatial convergence of the present method for the force coefficients: (a) lift and (b) drag. For increasing number of control volumes $\mathrm{N_c}$; typical grid size $h \equiv 1/\mathrm{N_c}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.5. The settings used and more accurate values for the estimate for $h = 0$ are also found in table 5.5.

cancellation of errors [133], due to integrating the pressure.

For the moment coefficient, the convergence behaviour observed for the finest grids is such that no accurate estimate of the value for $h = 0$ could be made using Richardson extrapolation.

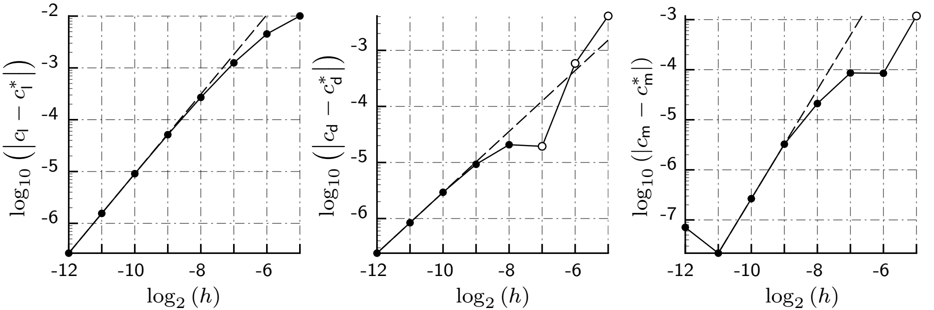
Upwind discretization with a larger far-field distance

To reduce the effect of the far-field boundary — and its corresponding boundary condition enforced there — on the results, the computational domain is expanded. To obtain a fair comparison, the grid in the 'near-field' region is kept unmodified. To that end, the number of cells in the radial direction is increased in order to increase the far-field distance. Following exactly the same procedure as for the original O-grid generation, using double the number of cells in the radial direction results in a far-field situated at approximately $80 \cdot 10^3$ times the chord length away from the aerofoil. For the discretization of the convective flux, Roe's approximate Riemann solver is used, with linear reconstruction

Table 5.6: Results obtained on single block O-grid with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 1/3$; far-field at 80 $\mathrm{k}\bar{c}$.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.80336340 \cdot 10^{-1}$	$-2.89396662 \cdot 10^{-6}$	$2.26883841 \cdot 10^{-3}$
2048	$1.80343874 \cdot 10^{-1}$	$-7.974599 \cdot 10^{-7}$	$2.26908260 \cdot 10^{-3}$
4096	$1.80345162 \cdot 10^{-1}$	$-1.968521 \cdot 10^{-7}$	$2.26903326 \cdot 10^{-3}$
ϕ^*	$1.80345428 \cdot 10^{-1}$	$4.42929 \cdot 10^{-8}$	$2.26910430 \cdot 10^{-3}$
$\bar{p}$	2.548	1.803	$3.615^{\dagger}$

†Order of convergence estimated using result of the coarser grid with $\mathrm{N_c} = 512$, for the corresponding value of c_m see table C.6 on page 210.



$\left(\bar{p} = 2.548,\ c_\mathrm{l}^* \approx 1.8035 \cdot 10^{-1}\right)$ $\left(\bar{p} = 1.803,\ c_\mathrm{d}^* \approx 4.4293 \cdot 10^{-8}\right)$ $\left(\bar{p} = 3.615,\ c_\mathrm{m}^* \approx 2.2691 \cdot 10^{-3}\right)$

(a) (b) (c)

Figure 5.13: Spatial convergence of the present method for the aerodynamic coefficients: (a) lift; (b) drag; and (c) moment. For increasing number of control volumes $\mathrm{N_c}$; typical grid size $h \equiv 1/\mathrm{N_c}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.6. Open circles indicate that the result is larger than the estimate for $h = 0$; the settings used and more accurate values for the estimate for $h = 0$ are also found in table 5.6.

about the centre of mass and weighting factor $\hat{\kappa} = 1/3$, i.e. the same settings as for the first investigation that has been presented.

Considering the results — presented in table 5.6 and figure 5.13 — the effect of the far-field boundary on both force coefficients is obvious. The lift coefficient becomes slightly higher, while the drag coefficient approximates the theoretical value of zero considerably closer than when a smaller far-field distance is used. The influence on the moment coefficient is less significant. When the order of convergence is considered, some differences are observed. Just as for the smaller far-field distance, the moment coefficient shows non-monotonic convergence behaviour. The convergence rate for the drag coefficient also shows the same discrepancy from the expected value as observed for the situation with a smaller far-field distance. The convergence rate for the lift however, is significantly lower, changing from almost 4 to less than 3.

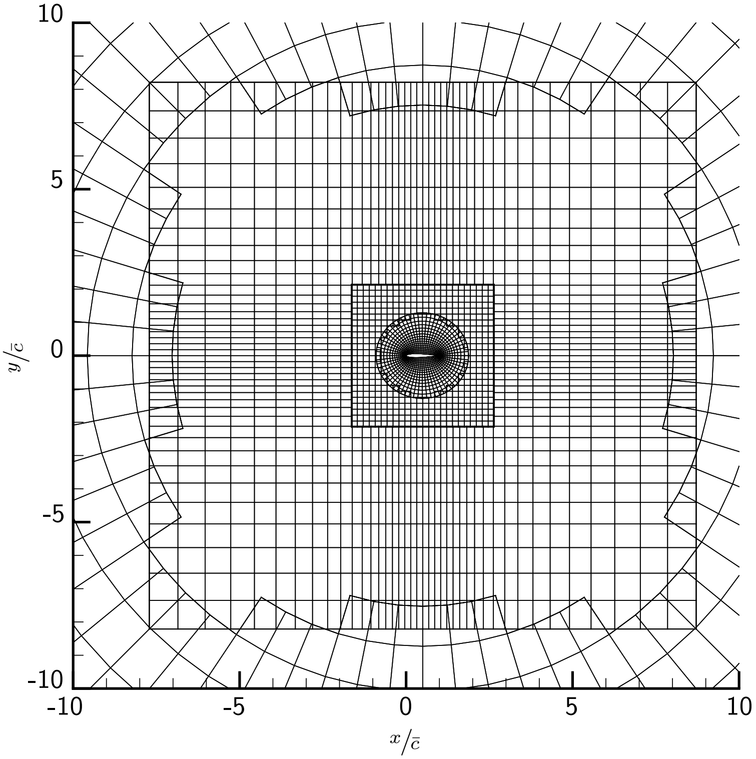


Figure 5.14: Detail of the overset grid in the region near the aerofoil, used for studying the spatial convergence characteristics of the flow solution method. This grid is obtained after coarsening the finest grid 6 times, giving a grid resolution for the body-fitted grid of $\mathrm{n}_1 = 64$ and $\mathrm{n}_2 = 16$ cells.

5.8.4 Results for composite overset discretization

Verification of the solution method, using a single block discretization of the flow domain, indicates a correct implementation. However, it is also necessary to check whether similar convergence characteristics are achieved, when an overset discretization is used instead. For this purpose, the part of the grid used for the discretization of the flow domain close to the aerofoil is retained. For the discretization of the far-field, a sequence of background grids is used. The configuration used is shown in figure 5.14 for the region around the aerofoil. The body-fitted grid reaches to about $1.0\,\bar{c}$, then there is a square uniform Cartesian grid of dimension $5.0\,\bar{c}$, for which the cell size approximately matches the cell size of the cells in the outer layer of the body-fitted grid. The second background grid is also Cartesian and it is refined towards the centre of the grid, with the cell size of the smallest cell matching that of the uniform grid. To enable a proper comparison of the results of the composite overset discretization with the previous results, the far-field boundary is chosen to be a circle with a radius of $150\,\bar{c}$ centred at the aerofoil at $x = 0.5\,\bar{c}$. Therefore, the third background grid is cylindrical with an inner radius of $5.5\,\bar{c}$ and an outer radius of $150\,\bar{c}$.

To generate a family of grids, the grid for the finest level is constructed, based on the procedure just pointed out. Each coarser grid is obtained by removing every other vertex

Table 5.7: Dimensions of the blocks for the finest grid used in the composite overset discretization of the flow domain around an extended NACA 0012 aerofoil. For body-fitted and cylindrical grid, n_1 denotes number of cells in circumferential direction and n_2 the number of cells in radial direction. Coarser grids used in the investigation are created by removing every other vertex in both directions. The bottom row lists the total number of control volumes used for the discretization.

block	n_1	n_2
body-fitted	4096	1024
uniform Cartesian	1792	1792
refined Cartesian	2816	2816
cylindrical	4096	1792
total	22 675 456	

in both directions. Moreover, special care is taken to make sure that the outer boundary of the body-fitted grid, i.e. the region where the transition from field control volumes to fringe control volumes takes place, is always exactly at the same location. For this purpose, additional cells are added, before the coarsening, to the body-fitted grid for the most fine member of the family, such that a layer of two fringe control volumes remains for the coarsest grid. The number of control volumes used for each of the blocks of the finest grid is listed in table 5.7.

To investigate the influence of using an overset discretization, it is assumed not to be necessary to test the different flux discretizations and settings previously considered. Therefore, this investigation has only been performed using Roe's approximate Riemann solver, with linear reconstruction about the centre of mass of the control volume. Weighted averaging of the gradient is performed with $\hat{\kappa}$ equal to $1/3$. Results of the calculations are shown in figure 5.15 and in table 5.8 for the three finest grids used.

Considering the close agreement of the computed coefficients as well as their corresponding spatial order of convergence with the results of the single block discretization, the correct implementation of the methods used for handling a composite overset grid is verified. Moreover, the results also indicate that the choice regarding how the convective flux is handled, appears to have a larger effect on the results than the method used for the discretization of the domain. Therefore, it can be concluded that a similar accuracy can be achieved using a composite overset discretization than when using a single block grid.

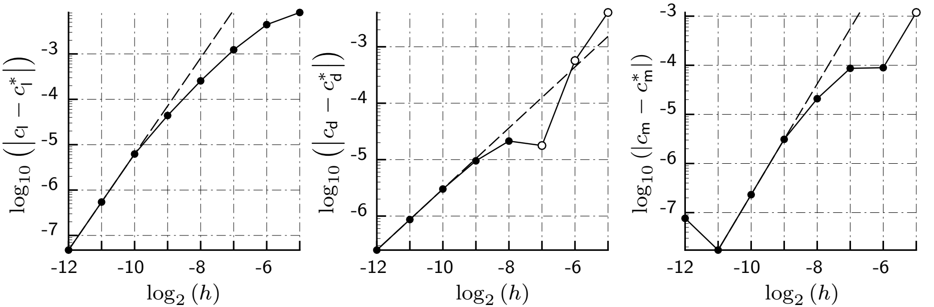
5.8.5 Summary of the results and concluding remarks

In the process of verification of the numerical implementation of the flow solution method, it has been found that the spatial order of convergence that is achieved is strongly affected by the different approaches that can be taken to solve the partial differential equations numerically. For instance, it has been found that the method used for extrapolating the pressure to impose a solid wall boundary condition has quite a significant effect on the observed spatial order of convergence. Moreover, the method used for numerically handling the convective flux and the way that the state at the interface is reconstructed

Table 5.8: Results obtained for composite overset discretization of the domain with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 1/3$; far-field at $150\,\bar{c}$.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
1024	$1.79778184 \cdot 10^{-1}$	$9.25195908 \cdot 10^{-6}$	$2.26219100 \cdot 10^{-3}$
2048	$1.79783864 \cdot 10^{-1}$	$1.13978406 \cdot 10^{-5}$	$2.26240533 \cdot 10^{-3}$
4096	$1.79784361 \cdot 10^{-1}$	$1.20153169 \cdot 10^{-5}$	$2.26234616 \cdot 10^{-3}$
ϕ^*	$1.79784408 \cdot 10^{-1}$	$1.22647777 \cdot 10^{-5}$	$2.26242260 \cdot 10^{-3}$
$\bar{p}$	3.516	1.797	3.745 †

†Order of convergence estimated using result of the coarser grid with $\mathrm{N_c} = 512$, for the corresponding value of c_m see table C.7 on page 211.



$\left(\bar{p} = 3.516,\ c_\mathrm{l}^* \approx 1.7978 \cdot 10^{-1}\right)$ (a)

$\left(\bar{p} = 1.797,\ c_\mathrm{d}^* \approx 1.2265 \cdot 10^{-5}\right)$ (b)

$\left(\bar{p} = 3.745,\ c_\mathrm{m}^* \approx 2.2624 \cdot 10^{-3}\right)$ (c)

Figure 5.15: Spatial convergence of the present method for the aerodynamic coefficients: (a) lift; (b) drag; and (c) moment, computed using a composite overset discretization. For increasing number of control volumes $\mathrm{N_c} = \mathrm{n}_1$ of the body-fitted grid; typical grid size $h \equiv 1/\mathrm{N_c}$. Dashed line represents the trend line for the observed order of convergence $\bar{p}$, listed in table 5.8. Open circles indicate that the result is larger than the estimate for $h = 0$; the settings used and more accurate values for the estimate for $h = 0$ are also found in table 5.8.

Table 5.9: Comparison of observed spatial order of convergence for force coefficients and moment coefficient of the present method, for the different settings used. For flow about an extended NACA 0012 aerofoil at $\alpha = 1.25°$ and $\mathrm{M}_\infty = 0.5$. Used settings are indicated between the parentheses, with: $\boldsymbol{x}_{\mathsf{cm}}$ = reconstruction about centre of mass of the control volumes; $1/2$ = reconstruction assuming uniform Cartesian grid; $150\,\bar{c}$ or $80\,\mathrm{k}\bar{c}$ = distance of far-field boundary; P = modified pressure boundary condition used; $\hat{\kappa}_x$ = value of x used for weighting factor $\hat{\kappa}$.

solver (settings)	c_{l}	c_{d}	c_{m}
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_{1/3}$)	3.961	1.784	3.731
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_{1/3}$, P)	5.449	1.853	4.213
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_0$, P)	2.288	1.141	2.106
Roe ($1/2$, $150\,\bar{c}$, $\hat{\kappa}_0$, P)	1.094	0.969	1.279
JST ($150\,\bar{c}$, P)	0.762	2.089	—
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $80\,\mathrm{k}\bar{c}$)	2.548	1.803	3.615
Roe, overset ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_{1/3}$)	3.516	1.797	3.745

also has a strong influence on the asymptotic behaviour of the flow solution method.

Regarding the location of the far-field boundary, it is well known that for the simulation of the flow around a lifting body, either the distance to the boundary must be very large or a correction must be applied at the far-field boundary for the circulation associated with a lift generating body. The results for the flow simulations with a far-field distance of about 80 $\mathrm{k}\bar{c}$ show that even a far-field distance of 150 $\bar{c}$ is insufficient for having a negligible effect on the result, when there is no vortex-correction applied to the boundary condition. A considerable difference is observed in the value for lift and drag coefficient, from the simulations for the large and the one for the 'small' far-field distance. Note, that this effect is less significant when a 3D flow configuration is considered. Furthermore, application of a vortex-correction to the far-field boundary conditions for a 3D flow configuration is far less trivial than in the 2D situation. Therefore, no vortex-correction has been implemented in the present method. Moreover, note that the distance of the far-field also appears to have an effect on the spatial order of convergence of the lift coefficient.

Although, a strong influence has been observed, for different aspects in the numerical solution procedure, no such influence was found for the methods that have been considered for the discretization of the flow domain. For that purpose, both a single block discretization, as well as a discretization of the flow domain employing composite overset grids were used. When the results for these two discretization methods are compared, a good correspondence is found for both the observed spatial order of convergence, as well as for the estimate of the exact value.

Finally, the results of all the series of flow simulations that have been performed are summarized. Table 5.9 lists observed orders of spatial convergence for the three dimensionless coefficients and the different settings used. Table 5.10 on page 113, lists the corresponding estimates for the value for $h = 0$. This table also includes the results presented by Vassberg and Jameson [195], who performed a similar study for the same flow configuration.

Table 5.10: Comparison of the extrapolated value for $h = 0$ for force coefficients and moment coefficient of present method, for different settings — separated by the horizontal line — with the results presented in the literature [195]. For flow about an extended NACA 0012 aerofoil at $\alpha = 1.25^\circ$ and $\mathrm{M}_\infty = 0.5$. Used settings are indicated between the parentheses, with: $\boldsymbol{x}_{\mathsf{cm}}$ = reconstruction about centre of mass; ½ = reconstruction assuming uniform Cartesian grid; $150\,\bar{c}$ or $80\,\mathrm{k}\bar{c}$ = distance of far-field boundary; P = modified pressure boundary condition used; $\hat{\kappa}_x$ = value of x used for weighting factor $\hat{\kappa}$; VORTEX = vortex-correction boundary condition used.

solver (settings)	c_{l}	c_{d}	c_{m}
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_{1/3}$)	$1.79780338 \cdot 10^{-1}$	$1.2358 \cdot 10^{-5}$	$2.262377 \cdot 10^{-3}$
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_{1/3}$, P)	$1.79780387 \cdot 10^{-1}$	$1.2357 \cdot 10^{-5}$	$2.262382 \cdot 10^{-3}$
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_0$, P)	$1.79779947 \cdot 10^{-1}$	$1.2406 \cdot 10^{-5}$	$2.262229 \cdot 10^{-3}$
Roe (1/2, $150\,\bar{c}$, $\hat{\kappa}_0$, P)	$1.79775988 \cdot 10^{-1}$	$1.8234 \cdot 10^{-5}$	$2.261951 \cdot 10^{-3}$
JST ($150\,\bar{c}$, P)	$1.79718633 \cdot 10^{-1}$	$1.3857 \cdot 10^{-5}$	—
Roe, overset ($\boldsymbol{x}_{\mathsf{cm}}$, $150\,\bar{c}$, $\hat{\kappa}_{1/3}$)	$1.79784408 \cdot 10^{-1}$	$1.2265 \cdot 10^{-5}$	$2.262423 \cdot 10^{-3}$
Roe ($\boldsymbol{x}_{\mathsf{cm}}$, $80\,\mathrm{k}\bar{c}$)	$1.80345428 \cdot 10^{-1}$	$4.4293 \cdot 10^{-8}$	$2.269104 \cdot 10^{-3}$
FLO82	$1.80345850 \cdot 10^{-1}$	$5.0 \cdot 10^{-8}$	$2.268812 \cdot 10^{-3}$
CFL3Dv6 (VORTEX)	$1.80351940 \cdot 10^{-1}$	$1.34 \cdot 10^{-7}$	$2.277383 \cdot 10^{-3}$
CFL3Dv6	$1.79783519 \cdot 10^{-1}$	$1.2221 \cdot 10^{-5}$	$2.270588 \cdot 10^{-3}$
Overflow v2.1t	$1.79777193 \cdot 10^{-1}$	$1.0030 \cdot 10^{-5}$	$2.262569 \cdot 10^{-3}$

Table 5.9 shows, that the observed order of convergence is strongly influenced by the settings used. Only when Roe's approximate Riemann solver is used, with linear reconstruction of the state at the interface about the centre of mass of the control volume, the observed order of convergence is in reasonable agreement with the expected formal order of convergence. Note, that this statement is only true if a weighted averaging factor of $\hat{\kappa} = 0$ is used. Moreover, for the extrapolation of the pressure to the solid wall halo control volumes, geometrical information should not be taken into account.

Comparing the estimates for the exact values of the dimensionless coefficients considered, a good agreement is found between the results presented by Vassberg et al. and the results obtained in the present research. The largest differences that exist are due to the larger distance from the aerofoil of the far-field boundary or the use of vortex-correction for the far-field boundary condition — the FLO82 method uses vortex-correction, as well as the first entry for CFL3Dv6 method in table 5.10. In these situations, the lift coefficient is about five counts higher. Moreover, the estimate for $h = 0$ for the drag coefficient is considerably closer to zero. Based on the close agreement of the results of the present method with the results presented in the literature, obtained with previously verified computational methods, the present implementation of the mathematical model is considered correct.

5.9 Validation

In the process of validation, an assessment is made to what extent the numerical implementation of the mathematical model, used to model the physics, possesses a satisfactory range of accuracy for representing reality, within the domain of applicability of the mathematical model [133]. Performing this task is important, because existence of a large discrepancy between numerical solution and reality would render the solution of an aerodynamic optimization problem useless.

For the validation of the present method, the transonic flow around the ONERA M6 wing is considered. The particular validation case that was selected, has been chosen because of the very well documented experimental set-up and results [158]. Moreover, this particular validation case has frequently been used to validate similar numerical simulation methods for inviscid flow as the one developed in the present research [152, 202]. Among the numerical methods that have been validated using the experimental results presented in the AGARD report are also other methods that were used for solving aerodynamic shape optimization problems [31, 82, 179].

Of the different flow conditions that were investigated experimentally, simulations corresponding to the experiment performed at a Mach number of $\mathrm{M}_\infty = 0.8395$ are presented here. First, the numerical set-up for the flow simulations is treated. Subsequently, the results of the numerical simulations are compared with the results from the experiment.

5.9.1 Numerical set-up

The geometry used in this validation test case is the ONERA M6 wing. An accurate description of the original geometry is provided in the AGARD report in which the experimental results have been presented [158]. The cross-section of the original wing geometry is defined by a single ONERA D aerofoil, which is a symmetrical aerofoil with a finite thickness at the trailing edge. In the present numerical investigation, however, a wing with a sharp trailing edge is employed. For this purpose, the tangent of the geometry of the ONERA D aerofoil at the blunt trailing edge location is computed. The aerofoil is then extended in that tangent direction, to the point that a zero thickness is achieved. Note, that the original geometry description specifies that the ONERA D profile is normal to the generator at 40.18 % chord. It was found, however, that the geometry used in similar validations of the flow solution method does not always comply with this aspect of the geometry definition. A likely cause for this quite common anomaly is an incorrect analytical description for the ONERA D profile, which was also extended to have a sharp trailing edge, that was provided in a report [152] which compared different numerical solutions for this particular test case.

The geometry of the tip-cap also slightly differs from the one used in the experiment. In the experiment, a half-body of revolution was used as a basis for the tip-cap. For the wing geometry used in the present numerical simulations, the wing tip is constructed from circular arcs that are exactly tangent to the wing at the location where the tip-cap starts. The geometrical properties of the ONERA M6 wing, used in the present numerical simulations, are given in table 5.11.

To facilitate the grid generation method, a NURBS surface representation of the wing has been computed that accurately resembles the wing geometry just described.

Table 5.11: Geometrical properties of the ONERA M6 wing used in the numerical simulations. Values include the wing tip geometry, except for the semi-span.

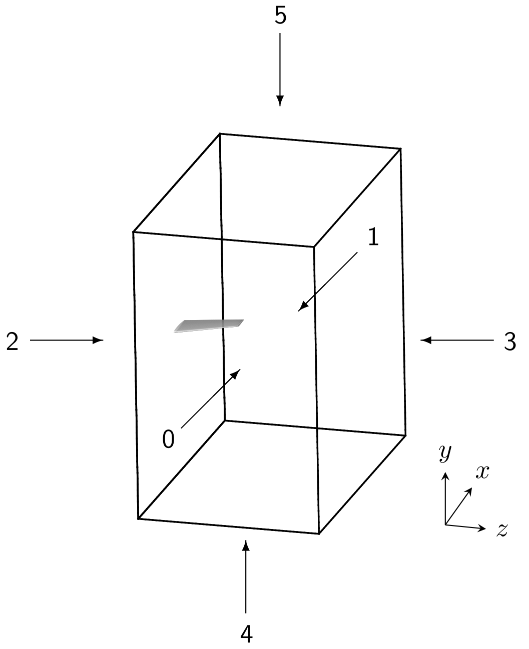
property	symbol	value
semi-span	$^{b}/_{2}$	1.1963 $[\,\mathrm{m}\,]$
mean aerodynamic chord	$\hat{c}$	0.6410 $[\,\mathrm{m}\,]$
semi plan form area	$^{S}/_{2}$	0.7668 $[\,\mathrm{m}^2]$
taper ratio	λ	0.5625
leading-edge sweep angle	Λ_{le}	30.00°
trailing-edge sweep angle	Λ_{te}	15.70°

The control point mesh corresponding to this NURBS surface is depicted in figure 6.10 on page 142 of chapter 6. The free-stream Mach number equals 0.8395. The free-stream inflow has an angle of incidence of 3.06° with respect to the wing chord line. A body-fitted grid with a CO-topology is constructed, that stretches out to approximately 5 times the mean aerodynamic chord of the wing. The surface grid on the wing is shown in figure 5.17. There is a square uniform Cartesian background grid, centred at the centre of the root cross-section of the wing. The block has a dimension of 10 times the aerodynamic mean chord length and about 3 times the semi-span, in the spanwise direction of the wing. The side boundary of this block is flush with the root of the wing. For this grid, the cell size approximately matches the cell size of the cells in the outer layer of the body-fitted grid. The second background grid is also a square Cartesian grid and the grid spacing is refined towards the centre of the grid, with the cell size of the smallest cell matching that of the uniform background grid. The boundaries of this block are at about $100\,\hat{c}$ away from the wing surface. In the spanwise direction, the block extends a distance of about 8 times the wing semi-span.

Figure 5.16 shows the ONERA M6 wing with a schematic representation of the flow domain. Note, that the dimensions of the depicted flow domain are not to scale. At the surface of the wing a slip-wall boundary condition is employed. At the outer boundary of the body-fitted grid, boundary conditions are enforced by means of fringe control volumes. At the symmetry plane, symmetry boundary conditions are enforced. The boundary conditions that have been imposed on the outer boundary of the background grid are indicated in figure 5.16.

5.9.2 Results

Apart from the validation of the inviscid flow solution method, the present investigation also serves the purpose of verifying the grid resolution required for properly solving a 3D optimization problem, in terms of the accuracy with which the force coefficients can be obtained. Therefore, the flow simulation has been performed for different grid resolutions. The number of cells used for the discretization in the finest grid are listed in table 5.12. Table 5.13 lists the number of cells of the body-fitted grid for the coarser grids used in the present investigation. These force coefficients are computed by the evaluation of the surface integral of the pressure times the unit normal vector over S_{wing}, the surface of a single wing half and then taking the dot product with the appropriate unit vector.



boundary	boundary condition
0	free-stream
1	subsonic outflow
2	symmetry
3	far-field
4	free-stream
5	subsonic outflow

Figure 5.16: The ONERA M6 wing with a schematic representation of the flow domain and the imposed boundary conditions indicated. Note, that the dimensions of the flow domain are not to scale.

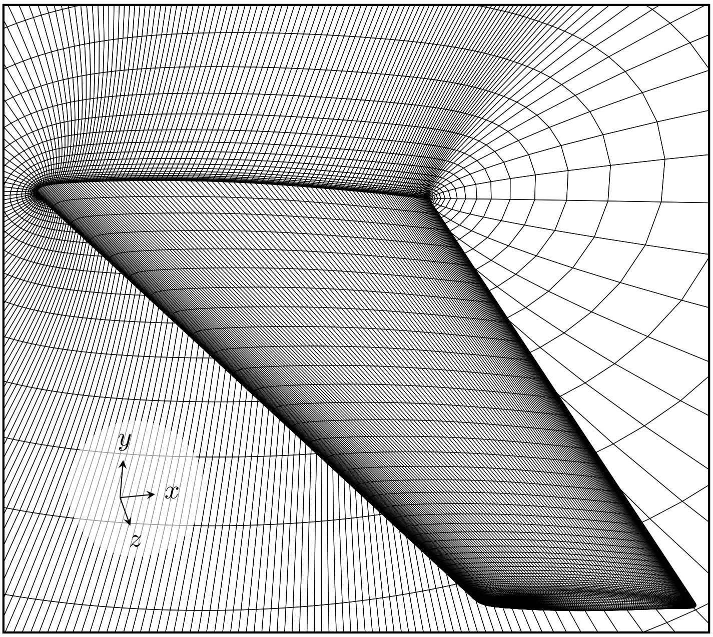


Figure 5.17: Surface grid on the ONERA M6 wing and on the symmetry plane for grid IV. Only every other grid line is shown.

Table 5.12: The number of cells used for the discretization of the flow domain around the ONERA M6 wing, for the finest grid. For the body-fitted grid n_1 is in the circumferential direction, n_2 is in the spanwise direction and n_3 is in the direction normal to the surface of the wing. For the background grids n_1 is in the chordwise direction, n_3 is in the spanwise direction and n_2 is in the third direction, perpendicular to both the chordwise and spanwise direction. The bottom row lists the total number of control volumes used for the discretization.

block	n_1	n_2	n_3
body-fitted	512	100	54
uniform Cartesian	50	50	50
refined Cartesian	60	60	30
total	2 997 800		

Table 5.13: Grid resolution for the body-fitted grid of the ONERA M6 wing of the coarser grids employed. See the caption of table 5.12 for the directions corresponding to n_1, n_2 and n_3.

grid	n_1	n_2	n_3
I	128	60	44
II	192	80	44
III	256	80	44

Subsequently, the result is non-dimensionalized with the free-stream dynamic pressure and the planform area of the wing. Note, that an additional factor of two is included to account for the fact that the surface integral is performed over a single wing half. The resulting expressions read

$$C_{\mathsf{L}} = \frac{2}{\frac{1}{2}\rho_\infty \left|\left|\boldsymbol{u}_\infty\right|\right|^2 S} \oint\limits_{S_{\mathsf{wing}}} p\boldsymbol{n}\,\mathrm{d}A \cdot \boldsymbol{k}_\perp, \tag{5.79}$$

$$C_{\mathsf{D}} = \frac{2}{\frac{1}{2}\rho_\infty \left|\left|\boldsymbol{u}_\infty\right|\right|^2 S} \oint\limits_{S_{\mathsf{wing}}} p\boldsymbol{n}\,\mathrm{d}A \cdot \boldsymbol{k}_{||}, \tag{5.80}$$

where $\boldsymbol{n}$ is the unit normal vector pointing outward of the flow domain, i.e. into the wing interior. Furthermore, $\boldsymbol{k}_{||}$ is the unit vector parallel to the free-stream flow, in the same direction. Vector $\boldsymbol{k}_\perp$ is found by taking the cross product of vector $\boldsymbol{k}_{||}$ with the unit normal vector of the symmetry plane pointing out of the flow domain. The results for the force coefficients for the different grid resolutions are listed in table 5.14, where the finest grid resolution is denoted by grid IV. Based on these results, the accuracy of the flow solution for grid II is considered sufficient to be used in the optimization.

A contour plot of the pressure distribution on the upper and lower side of the wing is shown in figure 5.18. The pressure distribution that was obtained for the experiment is plotted in figure 5.19, together with the ones computed, for grid II and IV. The pressure

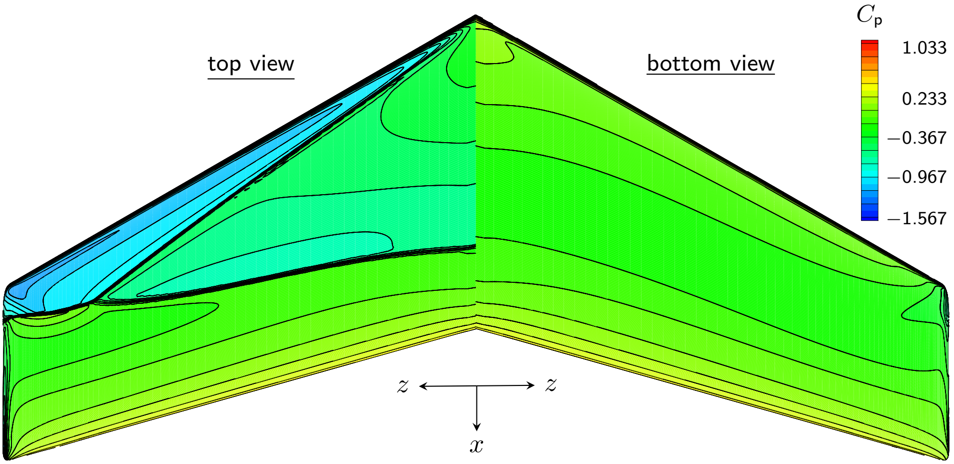


Figure 5.18: Iso-C_p contours on the surface of the ONERA M6 wing at a free-stream Mach number of $\mathrm{M}_\infty = 0.8395$ and an angle of attack of 3.06°. Results presented obtained for grid IV.

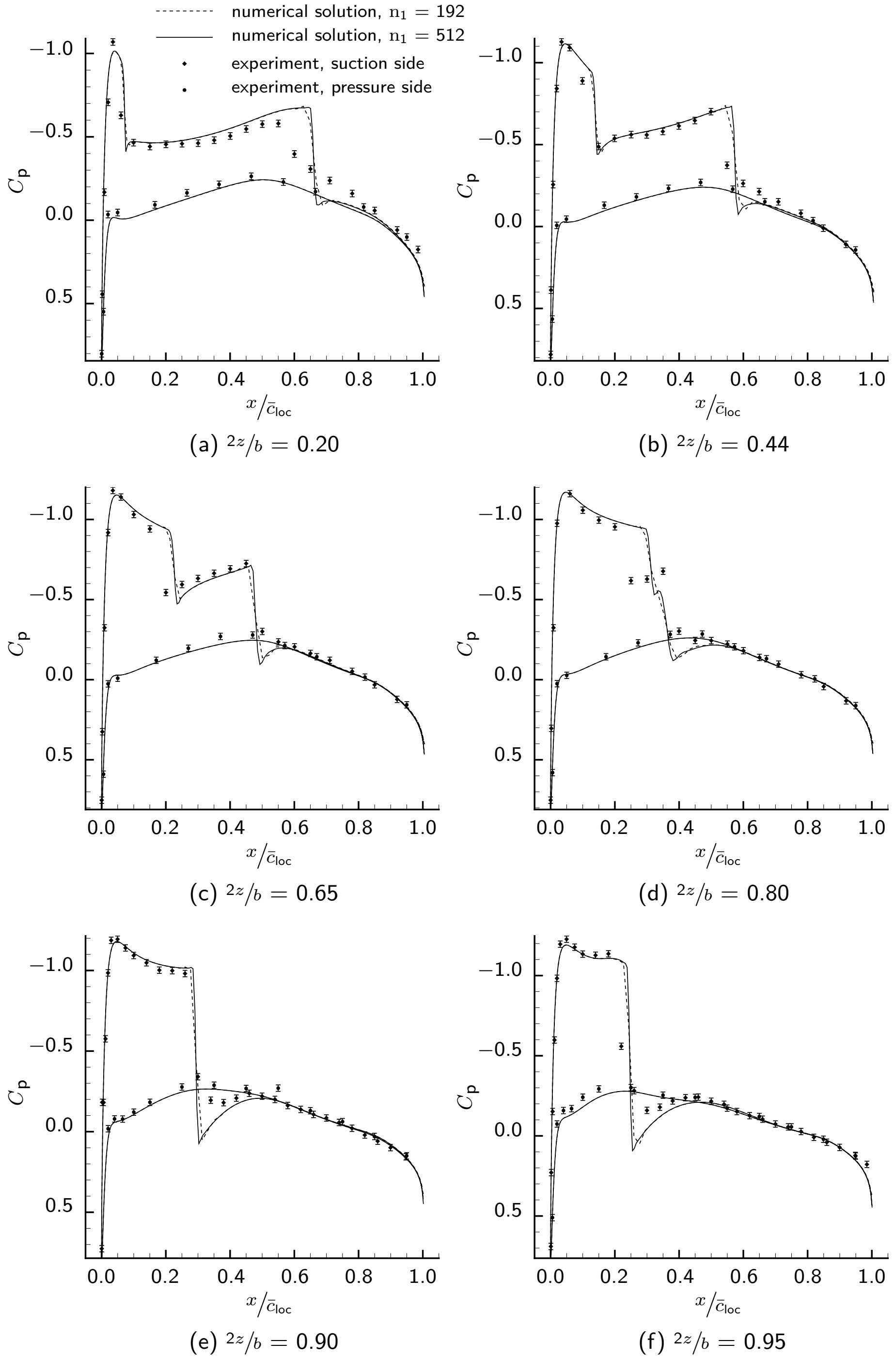
Table 5.14: Force coefficients from the numerical simulation of the flow around ONERA M6 wing at $\mathrm{M}_\infty = 0.8395$ and angle of attack $\alpha = 3.06°$.

grid	C_L	C_D
I	$2.87761 \cdot 10^{-1}$	$1.03179 \cdot 10^{-2}$
II	$2.88559 \cdot 10^{-1}$	$1.01124 \cdot 10^{-2}$
III	$2.88554 \cdot 10^{-1}$	$1.00779 \cdot 10^{-2}$
IV	$2.89034 \cdot 10^{-1}$	$1.01107 \cdot 10^{-2}$

distribution is plotted in terms of the dimensionless pressure coefficient, which is defined as

$$C_\mathsf{p} := \frac{(p - p_\infty)}{\frac{1}{2}\rho_\infty \left|\left|\boldsymbol{u}_\infty\right|\right|^2} \equiv \frac{2}{\gamma\,\mathrm{M}_\infty^2}\left(\frac{p}{p_\infty} - 1\right). \tag{5.81}$$

The numerical results show a good agreement with the results from the experiment. Moreover, only minor differences are observed between solutions from the fine and the coarser grid. Some of the differences observed between numerical solution and experiment — for instance, the pressure increase across the shock in sub-figure 5.19 (e) and (f) is larger for the simulation than was measured in the experiment — are caused by not incorporating viscous flow effects in the mathematical model. However, differences in the results can also be caused by other discrepancies between the true physics and the mathematical model used to represent it. For example, for the numerical simulation, the wing is rigid, while the model in the wind tunnel might be susceptible to aero-elastic deformation. Moreover, in the numerical simulation, the flow is assumed to be symmetric about the plane of the root section of the wing, while in the experiment an end-plate was used. Together with wind tunnel wall effects, this discrepancy can be another source of the differences observed for the results at the first cross-section, depicted in



(a) $2z/b = 0.20$ (b) $2z/b = 0.44$

(c) $2z/b = 0.65$ (d) $2z/b = 0.80$

(e) $2z/b = 0.90$ (f) $2z/b = 0.95$

Figure 5.19: Comparison between the experimental [158] and numerical results, for grid II and IV, for the flow around the ONERA M6 wing at a free-stream Mach number of $M_\infty = 0.8395$ at an angle of attack of 3.06° for different sections of the wing. The error bars provide a measure for the accuracy of the experimental results.

sub-figure 5.19 (a). These suggestions were also forwarded by Lyu, who performed similar simulations for the same flow configuration at the same flow conditions, however, including effects of viscosity, by incorporating a RANS turbulence model [113]. Results presented for these viscous flow simulations show similar differences between experimental and computational results as the ones observed here — except for the differences that are due to the absence of viscous flow effects, in the present results.

6

SENSITIVITY ANALYSIS

> *"To those who ask what the infinitely small quantity in mathematics is, we answer that it is actually zero. Hence there are not so many mysteries hidden in this concept as they are usually believed to be."* [14]
>
> — LEONHARD EULER (1707 – 1783)

SENSITIVITY analysis is an important aspect that must be considered when solving gradient-based optimization problems. In chapter 1, different methods for sensitivity analysis have been introduced. The approach taken in this research was also pointed out in that chapter, viz. a combination of the dual number method and the adjoint equation method. In this chapter, the approach is discussed in more detail. First, a more elaborate presentation of the concept of dual numbers is given, including details of its implementation in the present method. Subsequently, the discrete adjoint equation method is treated. Thereafter, details are given on the approach taken for computing the different partial derivatives involved in the adjoint equations and the expression for the total derivative. With the appropriate derivatives determined, the adjoint equations must be solved, for which the solution method is also discussed in this chapter.

6.1 Dual numbers

Dual number arithmetic is used to evaluate derivatives of functions that require a relatively limited number of mathematical operations. Recall the definition of the set of dual numbers $\mathbb{D}$, as defined in subsection 1.6.4:

$$\mathbb{D} := \left\{ x + \varepsilon y \ : \quad x, y \ \in \ \mathbb{R}, \quad \varepsilon^2 \equiv 0 \right\}. \tag{1.9}$$

As discussed in that same section, one of the implications of this definition is that dual numbers can be used to compute derivatives, because the derivative is proportional to the non-real part of the dual number. If function $f : \mathbb{D} \to \mathbb{D}$, for which $f(x) \in \mathbb{R} \; \forall \, x \in \mathbb{E} \underline{\Subset} \mathbb{R}$, is evaluated at $z = a + \varepsilon$ — i.e. a dual number with a non-real part of unit magnitude — the derivative equals the non-real part of the result, i.e.

$$\left. \frac{\mathrm{d}f}{\mathrm{d}x} \right|_a \equiv \mathfrak{D}\left[f\left(a + \varepsilon\right) \right], \tag{6.1}$$

which can be derived from a Taylor series expansion, as has been shown in subsection 1.6.4 on page 10.

To elucidate the concept of dual numbers, a couple of examples of the use of dual numbers in mathematical operations are presented below for $a, b, c, d \in \mathbb{R}$. Addition and multiplication of dual numbers is straightforward. Division can be done in the same way as for complex numbers, i.e. by multiplying numerator and denominator with the conjugate of the dual number in the denominator and then rewriting both numerator and denominator by applying the rules for addition and multiplication of dual numbers. The exponential function for dual numbers can be derived by considering its Maclaurin series[13]. The dual number expression for trigonometric functions like sine and cosine can subsequently be found using Euler's identity.

Examples of computational rules for dual numbers.

`Addition`:

$$(a+\varepsilon b)+(c+\varepsilon d)=(a+c)+\varepsilon\,(b+d)$$

`Multiplication`:

$$(a+\varepsilon b)\cdot(c+\varepsilon d)=ac+\varepsilon\,(ad+bc)+\varepsilon^2 bd=ac+\varepsilon\,(ad+bc)$$

`Division`:

$$\frac{a+\varepsilon b}{c+\varepsilon d}=\frac{(a+\varepsilon b)\,(c-\varepsilon d)}{(c+\varepsilon d)\,(c-\varepsilon d)}=\frac{ac+\varepsilon\,(-ad+bc)-\varepsilon^2 bd}{c^2-\varepsilon^2 d^2}=\frac{ac+\varepsilon\,(bc-ad)}{c^2}$$

`Exponential function`:

$$\mathrm{e}^{(a+\varepsilon b)}=\mathrm{e}^{a}\mathrm{e}^{\varepsilon b}=\mathrm{e}^{a}\sum_{\mathrm{n}=0}^{\infty}\frac{(\varepsilon b)^{\mathrm{n}}}{\mathrm{n}!}=\mathrm{e}^{a}\left[(1+\varepsilon b)+\varepsilon^2\sum_{\mathrm{n}=2}^{\infty}\frac{b^{\mathrm{n}}\varepsilon^{\mathrm{n}-2}}{\mathrm{n}!}\right]=\mathrm{e}^{a}+\varepsilon b\mathrm{e}^{a}$$

`Sine`:

$$\sin\left(a+\varepsilon b\right)=\frac{\mathrm{e}^{i(a+\varepsilon b)}-\mathrm{e}^{-i(a+\varepsilon b)}}{2i}=\frac{\mathrm{e}^{ia}-\mathrm{e}^{-ia}+\varepsilon ib\left(\mathrm{e}^{ia}+\mathrm{e}^{-ia}\right)}{2i}=\sin\left(a\right)+\varepsilon b\cos\left(a\right)$$

Other mathematical operations can be derived in a similar manner. Or, they can also be found by determining the derivative of the mathematical operation and considering the fact that this result is equal to the non-real part of the dual number, according to equation (6.1). Note, that the quotient is undefined for a dual number divisor with the real part equal to zero. This property does not present a problem, since it is also the case for the real-valued quotient and should therefore not be encountered in the optimization method. Furthermore note that, in contrast to complex numbers, the real part is not affected at all by the non-real part.

6.1.1 Implementation

The programming language used for the implementation of the optimization method is C++ [180]. One of the features of this language is the possibility to define so-called template functions. Template functions are functions that have not yet been specialized for a particular data type, which means that the same function can be used with different data types. Therefore, all functions in the optimization method have been defined as

---

[13] i.e. essentially a Taylor series expansion about argument 0.

template functions. Moreover, a new data type for dual numbers has been introduced in the method. For this dual number data type, all mathematical operations used in the flow solution method and hole cutting method have been defined. These two procedures are all that is required to use the dual number method for computing the required derivatives. Since, in this way, all functions can be evaluated for both a real-valued floating-point data type as well as for a dual number data type.

When the computation of the derivatives is considered, it can be observed that it is common that derivatives of a function need to be computed with respect to multiple elements of a vector quantity, i.e. partial derivatives. For instance the derivative of a function f with respect to all elements of a coordinate, i.e. $\frac{\partial f}{\partial \boldsymbol{x}}$. Another example is the derivative of a function f with respect to all elements of the vector of conserved variables, i.e. $\frac{\partial f}{\partial \boldsymbol{U}}$. To compute these derivatives more efficiently, an alternative means for the definition of the dual number is introduced:

$$\overline{\mathbb{D}}^{\mathrm{n}} := \left\{ x + \varepsilon \boldsymbol{y} \; : \; x \in \mathbb{R}, \quad \boldsymbol{y} \in \mathbb{R}^{\mathrm{n}}, \quad \varepsilon^2 \equiv 0 \right\}, \quad \mathrm{n} \in \mathbb{N}_1. \tag{6.2}$$

For a data type that complies with this definition, it is possible to compute multiple derivatives with a single function evaluation. The advantage of this so-called vector mode [75] is that the real part of the result is computed only once. For the non-real part of, for instance, dimension $\mathrm{n} = 5$, this saves four addition floating-point operations for each summation of two dual numbers and four floating-point multiplications for each dual number product encountered in the function evaluation. For a quotient of dual numbers, this modification saves eight multiplications and four division operations. The ratio of the number of operations saved, with respect to using a dual number with a single non-real component, increases with the length of the vector used. For very large vectors, this ratio will eventually approach an asymptote, for which the value depends on the combination of mathematical operations used. If the function, for instance, consists of a single summation only, than the ratio is limited to a factor of two. A more involved example concerns the computation of the residual vector of the Euler equations for unsteady flow. The asymptotic behaviour for this function is shown in the left graph of figure 6.1. This figure shows that, in the asymptotic range, the ratio of the number of operations saved amounts to approximately a factor of 63 for the situation considered. However, the right graph in figure 6.1 shows the corresponding ratio of the time saved by using a dual number with a non-real vector of variable size. The behaviour observed in this graph does not correspond with the result expected, based on the number of operations saved. For small vectors the efficiency of the method for computing the derivatives increases. For larger vectors, however, the efficiency deteriorates to a point that there is no significant advantage over using a dual number with a single non-real component. Further investigation revealed that this discrepancy between the expected efficiency and the observed one is due to an increase in the number of read misses from the L1 instruction-cache. These cache misses can have a serious effect on the performance [58], which explains the gain being less than expected for long vectors. Moreover, using large vectors, has the additional disadvantage of increased memory requirements of the method.

Since the performance gain does not strongly depend on the length of the vector for the non-real part, dual numbers with a non-real part of dimension three and five are used. This choice is made based on practical grounds, because these dimensions correspond to respectively the number of components of a coordinate and the number of conserved variables for each control volume. In this way, the derivatives with respect to all three

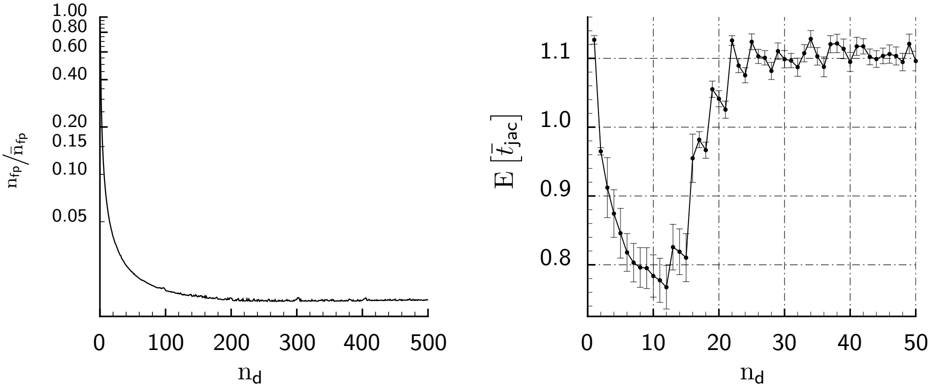


Figure 6.1: The left graph shows the ratio of the estimate for the number of floating-point arithmetic operations saved, as a function of n_d, the dimension of the non-real part of the dual number, for the computation of the sensitivity of the flow residual and the number of operations required to compute the result by means of a central-difference finite-difference method, denoted by $\bar{n}_{fp}$. The right graph shows the expected value for the corresponding timing, relative to using a real-valued central-difference approach, denoted by $E\left[\bar{t}_{jac}\right]$. Error bars indicate the standard deviation observed for the timing. The results have been obtained for $n_i = 257$, $n_{res} = 200$ and $n_{run} = 1000$. Note, that the range for the abscissa of the two graphs does not correspond. For more information on the method used to obtain these results, see appendix D.

components of a coordinate can be computed simultaneously. The same applies for the derivatives with respect to the conserved variables in a control volume, for a dual number with a non-real part of dimension $n = 5$.

Comparison of dual number method with algorithmic differentiation

Comparing the dual number method for computing derivatives with algorithmic differentiation by means of operator overloading [45], a number of similarities are observed: (i) both methods use a new data type for which the mathematical operators need to be defined; (ii) both methods yield the numerically exact derivative as a result. Therefore, the dual number method can be considered equivalent to an algorithmic differentiation method — just like the complex-step method [116, 117].

6.1.2 Approach

The general approach for computing derivatives using the dual number method is as follows. First, assign the value 1.0 to the non-real part of the dual number of the independent variable with respect to which the derivative needs to be determined. Make sure, that the non-real part of all other variables equals zero. Then, the function that computes the output variable for which the derivative needs to be determined is evaluated. Subsequently, the derivative is found by considering the non-real part of the output variable.

When the dual number method is used in vector mode, multiple derivatives can be computed with a single function evaluation. For that purpose, assign the value 1.0 to a different component of the dual number for each of the independent variables with respect to which the derivative needs to be determined. After evaluation of the function that computes the output variable, the derivative is found by considering the non-real component of the dual number corresponding to the non-real component of the independent variable that was assigned the value 1.0. The resulting value is equal to the derivative of the output parameter with respect to that independent variable. The other non-real components of the output variable correspond to the derivative with respect to the other input variables for which the non-real part of the dual number was assigned a value of unit magnitude.

6.2 Discrete adjoint equation method

The adjoint equation method is used to efficiently compute the derivatives of the objective function with respect to a large number of input parameters. In section 1.6.6 a derivation was given of the general adjoint equation method. Here, first a derivation of the adjoint equations, focused on the application in aerodynamic shape optimization, is presented. Subsequently, the distinction between the continuous and discrete adjoint equation method is pointed out.

6.2.1 Derivation

Consider an objective function $\mathcal{I} \in \mathbb{R}$ for an aerodynamic shape optimization problem. This objective function is in general a function of the design — represented by the design parameters $\boldsymbol{\chi} \in \mathbb{R}^{\mathrm{n_p}}$, with $\mathrm{n_p} \in \mathbb{N}_1$ the number of design parameters — and the flow solution, represented by the conserved flow variables for all control volumes $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{\mathrm{n_{cons}} \cdot \mathrm{n_c}}$, with $\mathrm{n_c} \in \mathbb{N}_1$ the number of control volumes and $\mathrm{n_{cons}} \in \mathbb{N}_1$ the number of conserved variables. Moreover, since in general only a discrete representation of the flow solution $\boldsymbol{\mathcal{U}}$ can be obtained, the objective function also depends on the computational grid $\boldsymbol{X} \in \mathbb{R}^{3 \cdot \mathrm{n_v}}$, with $\mathrm{n_v} \in \mathbb{N}_1$ the number of vertices, used for the discretization of the flow domain. Therefore,

$$\mathcal{I} = \mathcal{I}\left(\boldsymbol{\chi}, \boldsymbol{\mathcal{U}}\left[\boldsymbol{\chi}, \boldsymbol{X}\left(\boldsymbol{\chi}\right)\right], \boldsymbol{X}\left[\boldsymbol{\chi}\right]\right).$$

The total derivative of the objective function with respect to a single arbitrary design variable $\alpha \in \boldsymbol{\chi}$ then reads

$$\frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\alpha} = \frac{\partial \mathcal{I}}{\partial \alpha} + \frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}} \frac{\mathrm{d}\boldsymbol{\mathcal{U}}}{\mathrm{d}\alpha} + \frac{\partial \mathcal{I}}{\partial \boldsymbol{X}} \frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}. \tag{6.3}$$

The term requiring a number of flow solutions proportional to the number of design variables is the term with $\frac{\mathrm{d}\boldsymbol{\mathcal{U}}}{\mathrm{d}\alpha}$. This term can be expressed differently, employing the derivative of the vector of residuals $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{\mathrm{n_{cons}} \cdot \mathrm{n_c}}$ of the discretized flow equations[14] with respect to the same design variable α

$$\frac{\mathrm{d}\boldsymbol{\mathcal{R}}}{\mathrm{d}\alpha} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \alpha} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\mathcal{U}}} \frac{\mathrm{d}\boldsymbol{\mathcal{U}}}{\mathrm{d}\alpha} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}} \frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha} = \mathbf{0}. \tag{6.4}$$

---

[14] Note, that the vector of residuals actually just acts as the vector of constraint functionals of equation (1.12) on page 12.

This derivative is equal to the null vector, because the flow has to satisfy the governing equations, i.e. $\boldsymbol{\mathcal{R}} = \mathbf{0}$, independent of the design considered. This observation is used to obtain an expression for $\frac{\mathrm{d}\boldsymbol{\mathcal{U}}}{\mathrm{d}\alpha}$

$$\frac{\mathrm{d}\boldsymbol{\mathcal{U}}}{\mathrm{d}\alpha} = -\left[\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\mathcal{U}}}\right]^{-1}\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \alpha} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}\right). \tag{6.5}$$

Subsequently, this result is substituted in equation (6.3), resulting in

$$\frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\alpha} = \frac{\partial \mathcal{I}}{\partial \alpha} - \frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}}\left[\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\mathcal{U}}}\right]^{-1}\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \alpha} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}\right) + \frac{\partial \mathcal{I}}{\partial \boldsymbol{X}}\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}. \tag{6.6}$$

Then, define the adjoint vector $\boldsymbol{\psi}$ as

$$\boldsymbol{\psi} := -\left(\frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mathcal{U}}}\right]^{-1}\right)^{T}. \tag{6.7}$$

Substitution of the adjoint vector in equation (6.6) gives the expression that is used to compute the total derivative of the objective function with respect to the design variable α:

$$\frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\alpha} = \frac{\partial \mathcal{I}}{\partial \alpha} + \boldsymbol{\psi}^{T}\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \alpha} + \left(\frac{\partial \mathcal{I}}{\partial \boldsymbol{X}} + \boldsymbol{\psi}^{T}\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}\right)\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}. \tag{6.8}$$

The adjoint vector $\boldsymbol{\psi}$ is computed solving the system of linear equations, denoted by the term adjoint equations, which read

$$\left[\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\mathcal{U}}}\right]^{T}\boldsymbol{\psi} = -\left(\frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}}\right)^{T}. \tag{6.9}$$

6.2.2 Discrete versus continuous

As with most partial differential equations, obtaining an analytical solution for the adjoint equations is difficult and can usually only be done for very specific cases; for an example of an analytical solution see the work of Cnossen [44]. For the purpose of obtaining a solution of these equations for general cases, the adjoint equations must be discretized; just as is the case for the governing equations describing the flow. In order to discretize the partial differential equations, different approaches can be taken. One way is to start with the analytical non-linear partial differential equations, linearize these equations and subsequently derive an adjoint formulation based on the linearized equations. This adjoint formulation is then discretized to be able to solve the problem numerically. This approach is known as the continuous adjoint equation method.

The first step for the discrete adjoint equation method on the other hand, is to discretize the non-linear partial differential equations of the primal problem, i.e. the flow equations. Then linearize these discretized equations. Subsequently, this result is used to derive the adjoint equations. This approach is called the discrete adjoint equation method, which is the method used in the present research. An investigation of the differences between both approaches has been performed by Giles [69]. The major difference between both methods is that the continuous approach requires the explicit handling of the boundary conditions, while these are automatically enforced for the discrete approach, through boundary condition treatment of the original problem. Advantages of the discrete approach are:

- the derivatives obtained using the discrete adjoint equation method are consistent with the solution of the discretized governing equations. For the continuous adjoint equation method this consistency is not automatically achieved;
- the implementation of the adjoint equation method is relatively simple, by using algorithmic differentiation tools — or similar methods for automatically computing derivatives — for computing partial derivatives. Furthermore, methods implemented for obtaining the solution of the governing equations can be reused for solving the adjoint equations.

6.3 Efficient application of dual number method

The partial derivatives in equation (6.8) and (6.9) are computed using the dual number method. This method requires the evaluation of the function for which the partial derivative needs to be determined with the non-real part of the dual number for the variable considered disturbed. To limit the time required for the development of the method it is desirable to be able to reuse functions already available in the original method that is used to solve the governing equations. For instance, the Jacobian matrix, i.e. matrix $\frac{\partial \mathcal{R}}{\partial \mathcal{U}}$, can be computed reusing the functions for the evaluation of the residuals of the governing equations for each control volume in the flow domain. However, for efficiency of the method to compute these residuals, a loop is performed over all control volume faces in each block. For each face, the flow solution on the face is reconstructed on both sides of the face and this reconstructed flow solution is subsequently used to compute the flux over the face. This result is then utilized for both control volumes adjacent to this face, i.e. it is added to the residual of one control volume and subtracted from the other — depending on the sign of the flux and the direction of the normal vector of the face. In this way, conservation is ensured and the flux only needs to be computed once for each face. Straightforward reuse of this function for the dual number method is, however, very inefficient, because this approach requires a number of evaluations of the residual proportional to the number of control volumes in the flow domain to compute all entries of the Jacobian matrix. However, since the residual of the flow equations only depends on a limited number of neighbouring control volumes, a more efficient method can be employed. For this purpose, the concept of graph colouring, or more specifically graph vertex colouring is introduced.

6.3.1 Graph vertex colouring

The purpose of a so-called *proper* graph vertex colouring is to assign a label, commonly — and from here on — referred to as colour, to each vertex of the graph, such that no two vertices connected by an edge have the same colour. Figure 6.2 shows such a proper vertex colouring for a simple graph, where the different colours are designated by an integer number. Vertices of the graph that have been assigned the same colour

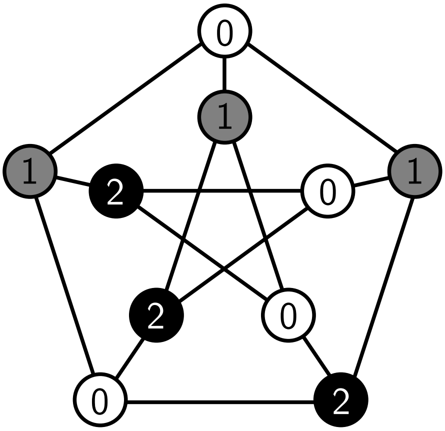


Figure 6.2: Example of a proper vertex colouring for the Petersen graph [21]. Each integer number designates a different colour.

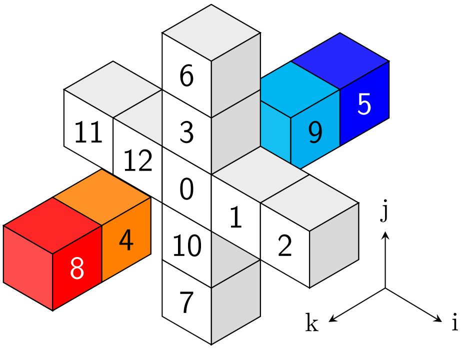


Figure 6.3: Stencil for residual computation of the Euler equations. Figure adapted from Lyu [112].

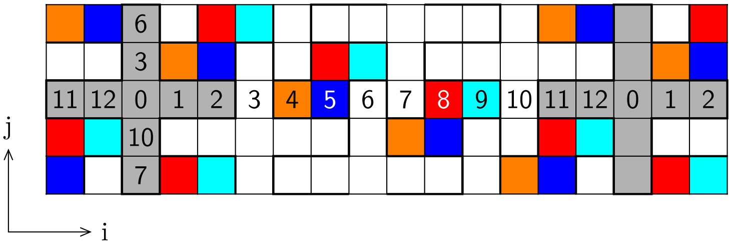


Figure 6.4: Cross-section of the colouring of the control volumes used for computing the Jacobian matrix. Each integer number represents a different colour. Figure adapted from Lyu [112].

form an independent set. This property is useful for the construction of, for instance, the Jacobian matrix, using the dual number method.

For this purpose, a graph can be constructed which resembles the method employed for discretizing the governing equations on the finite volume grid used. Subsequently, a proper vertex colouring must be determined for the resulting graph. If this proper colouring uses k colours, then the complete Jacobian matrix can be computed — with the dual number method — with a number of residual evaluations proportional to k. The number of colours required for colouring the graph depends on the method used for this purpose. The minimum number of colours is bound by the chromatic number of the graph considered [21]. Independent of the method used[15], the number of colours found is in general considerably smaller than the total number of control volumes used for the discretization of the flow domain. Therefore, the application of graph vertex colouring, to compile independent sets of control volumes, greatly reduces the number of evaluations of the function that computes the residual of the flow equations.

---

[15] Excluding the naive approach of assigning each vertex a unique colour.

For the computation of the different matrices, treated in the subsequent sections, the method used for determining the colouring will be discussed.

6.4 Jacobian matrix $\left(\text{i.e. } \frac{\partial \mathcal{R}}{\partial \boldsymbol{\mathcal{U}}}\right)$

As pointed out in the preceding section, the aim is to reuse the existing methods for obtaining the flow solution, without sacrificing too much on efficiency. This approach not only limits the time required for the implementation and verification of the method, but also renders the resulting method for computing derivatives fully consistently with the original method.

For the purpose of computing the Jacobian matrix efficiently, first a suitable colouring of the control volumes must be obtained. Using a method similar to the procedure explained by Goldfarb [74] — i.e. a method that seeks to find a three-dimensional packing sequence that minimizes the unused space between stencils — an expression can be obtained for determining the colour number $\mathrm{c} \in \mathbb{N}_0$ of a control volume, for a second-order accurate spatial discretization of the Euler equations, for which the stencil is displayed in figure 6.3. This expression reads:

$$\mathrm{c} = (\mathrm{i} + 3\mathrm{j} + 4\mathrm{k}) \mod 13, \tag{6.10}$$

where i, j and $\mathrm{k} \in \mathbb{N}_0$ are the control volume indices of the control volume considered and mod refers to the modulus operation, which can be expressed as

$$\mathrm{a} \bmod \mathrm{b} \equiv \mathrm{a} - \left\lfloor \frac{\mathrm{a}}{\mathrm{b}} \right\rfloor \mathrm{b}, \quad \mathrm{a}, \mathrm{b} \in \mathbb{Z},$$

with $\lfloor \cdot \rfloor$ the floor operation, defined for $x \in \mathbb{R}$ as: $\lfloor x \rfloor := \max\{\mathrm{m} \in \mathbb{Z} : \mathrm{m} \leq x\}$. A cross-section of the three-dimensional packing is shown in figure 6.4. Considering equation (6.10), it is observed that the number of colours required for this colouring strategy amounts to 13.

For determining the colours of the control volumes in the multi-block discretization of the flow domain, the different blocks are treated independently. Halo control volumes that represent a physical boundary condition, like a solid wall, are not assigned a colour. Because these boundary conditions are enforced in the method used for the computation of the residual and are therefore automatically taken into account. Halo control volumes that represent an internal-matching type of boundary condition, on the other hand, are assigned a colour. In this way, requiring communication is prevented in the case the matching block resides in the memory belonging to a different processor core. Moreover, this approach also prevents the occurrence of two control volumes in the stencil with the same colour. This situation would otherwise arise if two different boundaries of the same block match. An alternative solution to this problem is the use of a different colouring algorithm. However, given the elegance and simplicity of the current algorithm, an alternative approach would most likely result in a more elaborate colouring procedure. Furthermore, the number of colours required is also likely to increase, because the current algorithm requires a number of colours equal to the chromatic number of the graph. Since the computational effort is proportional to the number of colours, the resulting method for computing the Jacobian matrix would therefore be less efficient, when the number of colours increases.

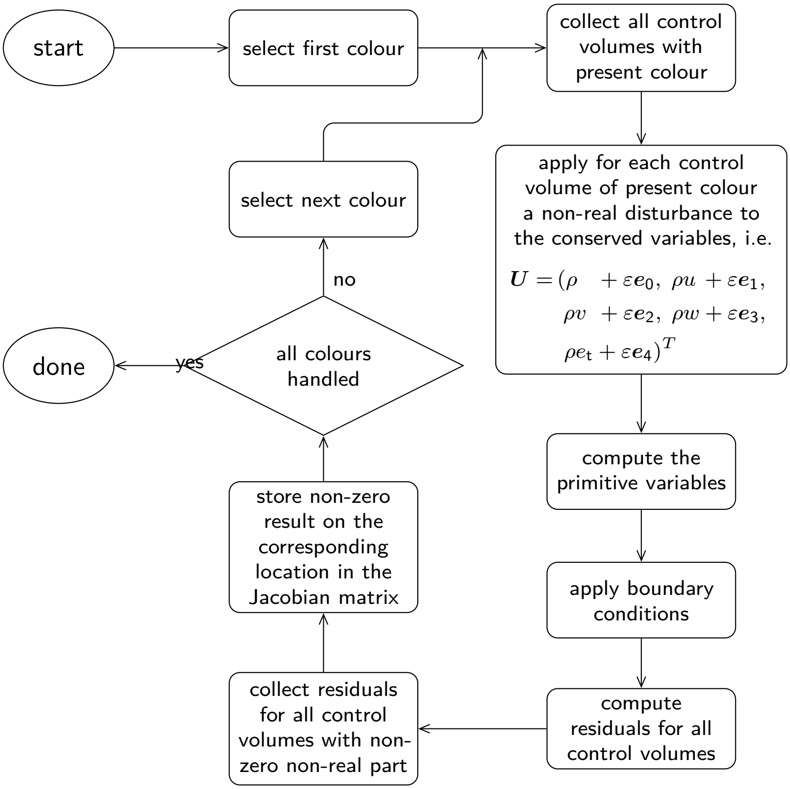

start
select first colour
collect all control volumes with present colour
apply for each control volume of present colour a non-real disturbance to the conserved variables, i.e.
$U = (\rho + \varepsilon \boldsymbol{e}_0,\ \rho u + \varepsilon \boldsymbol{e}_1,\ \rho v + \varepsilon \boldsymbol{e}_2,\ \rho w + \varepsilon \boldsymbol{e}_3,\ \rho e_{\mathrm{t}} + \varepsilon \boldsymbol{e}_4)^T$
select next colour
no
all colours handled
yes
done
compute the primitive variables
store non-zero result on the corresponding location in the Jacobian matrix
apply boundary conditions
collect residuals for all control volumes with non-zero non-real part
compute residuals for all control volumes

Figure 6.5: Flow chart of the approach for computing the entries of the Jacobian matrix.

Once the vertex colouring for the control volumes has been determined, the control volumes which have been assigned the same colour are collected. Then, the entries of the Jacobian matrix are determined using the dual number method. Using dual numbers with a non-real part of dimension $\mathrm{n}_{\mathsf{ncons}}$, the derivative of the residual with respect to all conserved variables of a control volume can be determined simultaneously. The procedure, which is summarized in figure 6.5, is therefore as follows. For all control volumes with the same colour number the conserved variables are disturbed. For each of the conserved variables a different component of the non-real part of the dual number is used. Subsequently, the function used for computing the residuals is called. This procedure includes computing the primitive variables and applying physical boundary conditions. For halo control volumes that correspond to a rotational periodic boundary, the boundary conditions are also applied. Note, that this approach requires the conserved flow variables to have the value of the conserved variables of the donor control volume, before rotation is applied, before assigning a non-real disturbance. The other internal matching boundary conditions are not applied. After the residuals have been computed, the non-real part of the residual of the control volumes is considered. Only control volumes for which the stencil contains one of the control volumes for which the non-real part of the conserved variables has been disturbed, have a non-real part unequal to

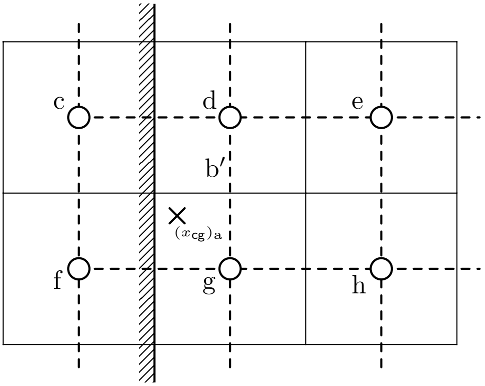


Figure 6.6: Schematic representation of halo control volumes c and f that act as donor of a fringe control volume a for which only the centre of mass is depicted. Cell b' represents a cell of the dual grid.

zero. These non-zero contributions are then collected and placed in their corresponding location in the Jacobian matrix. Subsequently, the non-real part of all flow variables is reset back to zero and control volumes corresponding to the next colour are considered. This procedure is repeated until all colours have been handled and therewith the Jacobian matrix has been populated completely.

Extension to composite overset discretizations

To construct the Jacobian matrix for composite overset grids, additional work needs to be performed, because of the occurrence of fringe control volumes. In contrast to halo control volumes, which have not been given an explicit representation in the matrix — in order to limit the dimension of the matrix — fringe control volumes are represented as an entry in the matrix. This choice was made to simplify the implementation of the construction of the Jacobian matrix. Halo control volumes are, depending on the boundary conditions, influenced by a maximum of two control volumes, always from a single block. While the flow solution for a fringe control volume is in general affected by at least 8 different control volumes, possibly from different blocks. Therefore, to account for the effect of a halo control volume for the control volumes affected is a relatively easy task, while it is considerably more elaborate for the fringe control volumes.

Recall that the residual of a fringe control volume with global index $\mathrm{a} \in \mathbb{N}_0$ was defined as

$$\boldsymbol{R}_{\mathrm{a}} := \langle \boldsymbol{U} \rangle_{\mathrm{a}} - \sum_{\mathrm{m}=0}^{\mathrm{m}<\mathrm{N_F}} \left(\omega_{\mathrm{m}} \mathbb{U}_{\mathrm{m}} \right) \equiv \boldsymbol{0}. \tag{5.50}$$

Linearization of this expression with respect to the conserved variables of the control volumes involved, gives the relation

$$\frac{\partial \boldsymbol{R}_{\mathrm{a}}}{\partial \boldsymbol{\mathcal{U}}} \Delta \boldsymbol{\mathcal{U}} = \Delta \langle \boldsymbol{U} \rangle_{\mathrm{a}} - \sum_{\mathrm{m}=0}^{\mathrm{m}<\mathrm{N_F}} \left(\omega_{\mathrm{m}} \Delta \mathbb{U}_{\mathrm{m}} \right). \tag{6.11}$$

Considering this equation, it can be observed that constructing the matrix entries for a fringe control volume does not require the flow solution of the donor control volumes. Therefore, when the donor control volumes belong to a block that resides in the memory of a different processor, no communication is required for the construction of the matrix. Note however, that this statement is only true if all donor control volumes are field control volumes. Because a cell-centred approach is employed in the present research, it may happen that the set of donor control volumes contains one or more halo control volumes as well. But since halo control volumes do not always have an explicit representation in the matrix, the dependency of the halo control volume must be taken into account, when the contribution of the fringe control volume is determined. A schematic representation of such a situation in 2D is depicted in figure 6.6. Consider cell b' of the dual grid to be the donor of fringe control volume a. In this case two of the donors of the fringe control volume are halo control volumes. Let the boundary condition be given by the relation

$$\langle \boldsymbol{U} \rangle_{\mathsf{H}} = f\left(\langle \boldsymbol{U} \rangle_{\mathsf{F0}}, \langle \boldsymbol{U} \rangle_{\mathsf{F1}}\right), \tag{6.12}$$

where $\langle \boldsymbol{U} \rangle_{\mathsf{H}}$ is the control-volume-averaged value of the conserved variables in the halo control volume and $\langle \boldsymbol{U} \rangle_{\mathsf{F0}}$ and $\langle \boldsymbol{U} \rangle_{\mathsf{F1}}$ are the first and second control volume, respectively, neighbouring the halo control volume. The linearization of the expression for the residual of this fringe control volume can then be expressed as

$$\begin{aligned}\frac{\partial \boldsymbol{R}_{\mathrm{a}}}{\partial \boldsymbol{\mathcal{U}}} \Delta \boldsymbol{\mathcal{U}} =& \Delta \langle \boldsymbol{U} \rangle_{\mathrm{a}} - \left(\left[\omega_1 \underline{\underline{I}} + \omega_0 \frac{\partial f}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{g}}} \right] \Delta \langle \boldsymbol{U} \rangle_{\mathrm{g}} \right. \\ & \left[\omega_3 \underline{\underline{I}} + \omega_2 \frac{\partial f}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{d}}} \right] \Delta \langle \boldsymbol{U} \rangle_{\mathrm{d}} \\ & \left. \omega_2 \frac{\partial f}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{e}}} \Delta \langle \boldsymbol{U} \rangle_{\mathrm{e}} + \omega_0 \frac{\partial f}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{h}}} \Delta \langle \boldsymbol{U} \rangle_{\mathrm{h}} \right),\end{aligned} \tag{6.13}$$

which only involves control volumes that have a representation in the matrix. The consequence of this approach is that

$$\frac{\partial f\left(\langle \boldsymbol{U} \rangle_{\mathrm{d}}, \langle \boldsymbol{U} \rangle_{\mathrm{e}}\right)}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{d}}}, \quad \frac{\partial f\left(\langle \boldsymbol{U} \rangle_{\mathrm{d}}, \langle \boldsymbol{U} \rangle_{\mathrm{e}}\right)}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{e}}}, \quad \frac{\partial f\left(\langle \boldsymbol{U} \rangle_{\mathrm{g}}, \langle \boldsymbol{U} \rangle_{\mathrm{h}}\right)}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{g}}} \quad \text{and} \quad \frac{\partial f\left(\langle \boldsymbol{U} \rangle_{\mathrm{g}}, \langle \boldsymbol{U} \rangle_{\mathrm{h}}\right)}{\partial \langle \boldsymbol{U} \rangle_{\mathrm{h}}}$$

must now be determined explicitly. Moreover, if the halo control volume belongs to a block that resides on a different processor, communication of this sensitivity information is required in order to construct the Jacobian matrix.

Regarding the implicit solution method, there is an important note on updating the flow solution with the solution of the system of linear equations. For field control volumes, the flow solution is updated by adding the result of solving the system of linear equations given by equation (5.51) on page 91. However, although fringe control volumes have been given a representation in the matrix, the flow solution in fringe control volumes is always determined by means of interpolation of the solution from donor control volumes. Updating the flow solution for fringe control volumes in the same way as for field control volumes leads to inconsistencies when the system of linear equations is not solved to machine accuracy. Since the customary approach in the present research is to let the residual of solving the system of linear equations only converge to a modest value of

Table 6.1: The number of cells used for the discretization of the flow domain around the ONERA M6 wing. For the body-fitted grid n_1 is in the circumferential direction, n_2 is in the spanwise direction and n_3 is in the direction normal to the surface of the wing. For the background grids n_3 is in the spanwise direction.

block	n_1	n_2	n_3
body-fitted	128	60	44
uniform Cartesian	50	48	47
refined Cartesian	50	48	47
total		563 520	

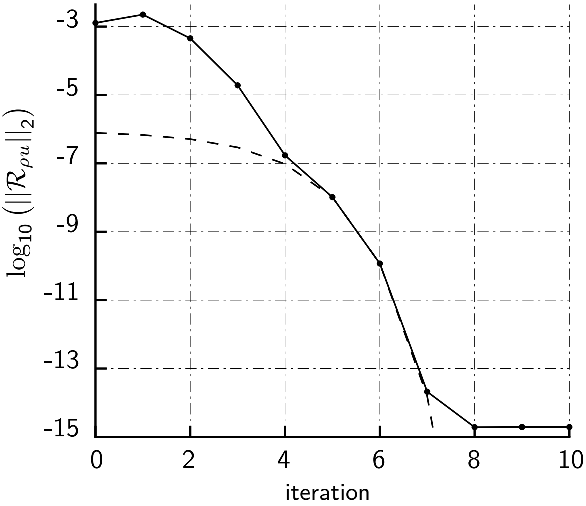
the residual, inconsistencies would therefore be introduced with this updating strategy. These inconsistencies accumulate and will eventually lead to an unstable or not converging iterative process.

6.4.1 Verification

For the verification of the correctness of the determination of the Jacobian matrix, it is employed in the procedure to obtain the flow solution. As pointed out in section 5.6, Newton's method is used for this purpose. When the initial state is sufficiently close to the steady-state solution, Newton's method features quadratic convergence. There are techniques to determine if an initial guess is within the domain of attraction of the method, for which quadratic convergence is observed. However, a more practical approach is to just try an initial guess and check if the convergence observed is quadratic. If no quadratic convergence is observed, there are three possible reasons, viz. (i) the initial guess is not sufficiently close to the solution, (ii) the construction and computation of the Jacobian is incorrect or (iii) the function considered has particular properties that prevent Newton's method from converging quadratically. However, when quadratic convergence is observed, it shows that neither of these conditions apply, which therefore implies that the construction of the Jacobian matrix is done correctly. Moreover, it is considered unlikely that the third reason applies for the current application.

To present a test case that is sufficiently representative for the proposed application of the flow solution method, the same case as already considered for the validation of the flow solution method is considered here as well: the transonic flow around an ONERA M6 wing, subject to transonic flow conditions, i.e. a free-stream Mach number of $\mathrm{M}_\infty = 0.8395$ and an angle of attack of $\alpha = 3.06°$. For the discretization of the flow domain, a composite overset grid is used, similar to the grid described in section 5.9.1. The number of control volumes used for the discretization is listed in table 6.1.

The initial conditions for obtaining the flow solution are free-stream conditions everywhere in the flow domain. For the purpose of obtaining a suitable initial guess for Newton's method, first a fixed number of 5000 multistage Runge-Kutta iterations is performed, with a local time-step to speed-up convergence. The resulting nonconverged intermediate flow solution is then used as the initial guess for Newton's method. The system of linear equations involved in Newton's method is solved by means of a GMRES method, which is made to converge 15 orders of magnitude, to make sure that the under-



iteration	$\|\|\mathcal{R}_{\rho u}\|\|_2$
0	$1.278 \cdot 10^{-3}$
1	$2.243 \cdot 10^{-3}$
2	$4.524 \cdot 10^{-4}$
3	$1.912 \cdot 10^{-5}$
4	$1.711 \cdot 10^{-7}$
5	$1.026 \cdot 10^{-8}$
6	$1.175 \cdot 10^{-10}$
7	$2.099 \cdot 10^{-14}$
8	$1.925 \cdot 10^{-15}$
9	$1.947 \cdot 10^{-15}$
10	$1.945 \cdot 10^{-15}$

Figure 6.7: Graph showing the convergence history of the ℓ^2-norm of the residual of the x-component of the equation for conservation of momentum — scaled by the value of the ℓ^2-norm of the residual computed for free-stream conditions everywhere in the domain. Geometry: ONERA M6 wing; conditions: $\mathrm{M}_\infty = 0.8395$ and $\alpha = 3.06°$. The solid line represents the actual convergence history and the dashed line shows the trend for a quadratic convergence.

lying iterative method does not affect the results. The corresponding convergence history for Newton's method is depicted in figure 6.7. This graph shows the convergence history of the ℓ^2-norm of the residual of the x-component of the equation for conservation of momentum. This convergence behaviour is also typical for the other components of the residual vector, which are, however, not shown here. Together with the convergence history a line has also been plotted that represents a quadratic convergence. Considering the close agreement between this trend line and a part of the convergence history of Newton's method, it can be concluded that quadratic convergence is indeed observed. The deviations from quadratic convergence are either due to (i) not yet being within the domain of attraction of the method for which convergence is quadratic, this property applies for the first few iterations; or (ii) reaching the machine accuracy for double-precision arithmetic, which is true for the last couple of iterations.

Since quadratic convergence of Newton's method has been observed, non of the reasons mentioned, that would prevent such behaviour, apply. Therefore, the implementation of the method for determination of the Jacobian matrix is considered to be correct.

6.5 Grid sensitivity of flow residual $\left(\text{i.e. } \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}\right)$

To compute the sensitivity of the residual of the flow equations with respect to the vertex coordinates of the grid, i.e. $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}$, the functions for computing the residual of the flow equations can also be reused. Because the dependency of the residual on the vertex

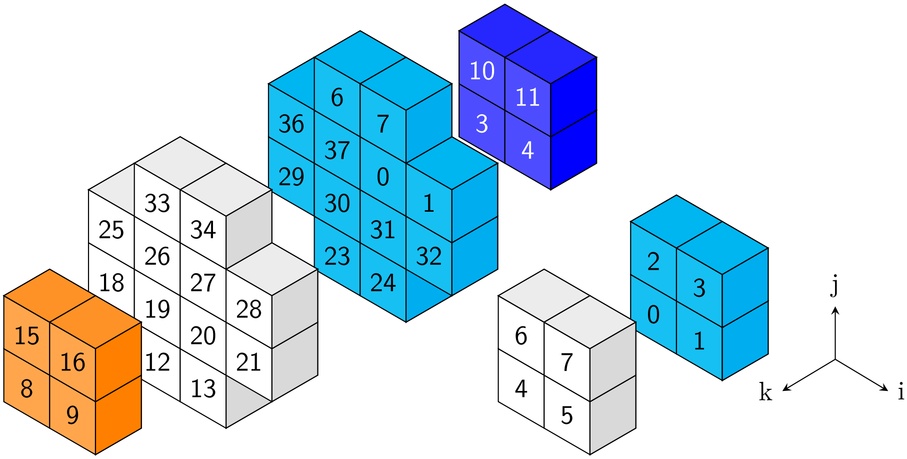


Figure 6.8: Stencil for the dependency of the residual computation of the Euler equations with respect to the vertices of the grid. Stencil depends on method used for computing convective flux. Left schematic applies to Roe's approximate Riemann solver with linear reconstruction around the centre of mass of the control volume. Right schematic applies to a central discretization by means of the JST scheme. Note, that here the cubes represent vertices instead of control volumes. Figure adapted from Lyu [112].

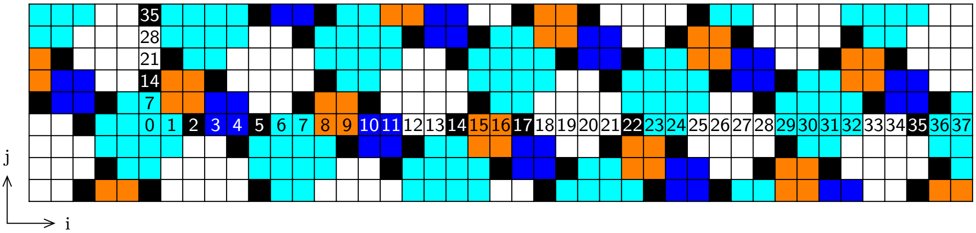


Figure 6.9: Cross-section of the colouring — defined by equation (6.14) — of the vertices used for computing $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}$. Each integer number represents a different colour. Figure adapted from Lyu [112].

coordinates differs from the dependency of the residual on the control-volume-averaged flow variables, a different graph colouring has to be used. Moreover, this graph colouring also depends on which approach is taken for computing the convective flux. For Roe's scheme with linear reconstruction about the centre of mass of the control volume, the following function for computing the colour number is used [112]

$$\mathrm{c} = (\mathrm{i} + 7\mathrm{j} + 27\mathrm{k}) \mod 38. \tag{6.14}$$

This approach requires a total of 38 colours to create independent sets. If on the other hand the JST scheme is used, or if the linear reconstruction is performed assuming a uniform Cartesian grid, then

$$\mathrm{c} = (\mathrm{i} + 2\mathrm{j} + 4\mathrm{k}) \mod 8, \tag{6.15}$$

can be used to construct independent sets. For this colouring scheme, as few as 8 different colours is sufficient. A schematic representation of both colouring strategies is shown in figure 6.8. Moreover, a cross-section of the three-dimensional packing for the colouring given by equation (6.14) is depicted in figure 6.9.

To compute the sensitivity, a dual number with a non-real part of dimension 3 is used. In this way, a non-real disturbance is applied to all three components of the coordinate of a vertex simultaneously. After applying the disturbance, the metrics are recomputed for the control volumes and faces affected; such as surface normal vectors and cell volumes. Then, the appropriate fluxes are evaluated and the residual of the flow equations is computed. The entries of the matrix are subsequently determined, by considering the residual of each control volume that is affected by a certain vertex being disturbed.

Different blocks of a multiblock discretization of the domain can be handled independently, provided that point matching boundary vertices — i.e. different vertices that represent the same physical location — are treated as a single entity. For vertices belonging to a boundary for which different boundary conditions apply than the internal-matching boundary conditions, the boundary conditions must be applied, after that the metrics have been computed.

Extension to composite overset discretizations

When a composite overset discretization is used, the sensitivity of the residual of the flow equations with respect to the vertex coordinates of the grid requires handling the grid sensitivity of the fringe control volumes. Considering the approach taken for determining the interpolation stencil of the fringe control volumes, presented in section 4.5, explicitly determining the dependency of the residual on the coordinate of the vertices involved is a very complicated and tedious task. Therefore, this step is circumvented by not explicitly computing the matrix $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}$, but only computing the product

$$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}} \frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}, \tag{6.16}$$

i.e. the product of the residual grid sensitivity matrix with the vector that represents the sensitivity of the grid with respect to a geometric design variable. For this purpose, the grid must be computed using dual numbers, with the design variable considered being assigned a non-real part of unit magnitude. Thereafter, the interpolation coefficients for fringe control volumes must be recomputed. Once the grid has been generated, the interpolation coefficients have been determined and the metrics of the grid have been computed, the result of equation (6.16) can be obtained by simply performing a residual evaluation on that grid. Modification to the grid generation method, required to handle dual numbers is discussed in section 6.9. Moreover, note that computing the interpolation coefficients of the fringe control volumes for the dual number grid requires that the holecutting procedure is carried out with dual numbers as well.

6.6 Flow sensitivity of objective function $\left(\text{i.e. } \frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}}\right)$

In the present research, objective functions considered involve integrals of the pressure over the surface of the object present in the flow field. For the solid wall boundary condition, discussed in section 5.5.1 on page 85, the pressure in the halo control volume is determined by linear extrapolation of the pressure inside the domain. Therefore, only two control volumes — i.e. the two control volumes closest to the wall — affect the pressure on a single face situated on the surface of the body. Straightforward reuse of the function that computes the objective function, for computing the flow sensitivity of the objective function by means of the dual number method, requires the surface integral to be evaluated a number of times proportional to the number of faces that are used for the discretization of the surface geometry. Note, that this number cannot be reduced through the application of colouring, because all control volumes involved affect the final result. In an effort to limit the amount of computational work, a separate function is created, which computes the pressure at the face and subsequently the contribution of that face to the objective function, based on the control-volume-averaged conserved flow variables of the two control volumes that determine the result. This function can then be used to compute the contribution of a control volume near the surface to $\frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}}$, using the dual number method.

Extension to composite overset discretizations

For a composite overset discretization of the domain, overlap of the surface grid can occur, which results in requiring the use of zipper grids to achieve a non-overlapping non-open discretization of the surface. Section 4.3.3 describes the approach taken for the evaluation of surface grids in these cases, i.e. the flow solution for the triangular faces of the zipper grid is obtained by means of interpolation of the flow solution of the non-blanked quadrilateral faces of the surface grid. Therefore, when a quadrilateral face, that is used for this interpolation, has a non-zero non-real part, the resulting flow solution for the triangle will have a non-zero non-real component as well. The contribution of the triangle to the surface integral — and therefore also its contribution to the sensitivity — is simply accounted for by considering the flow solution of the quadrilateral face and its corresponding interpolation weight.

6.7 Grid sensitivity of objective function $\left(\text{i.e. } \frac{\partial \mathcal{I}}{\partial \boldsymbol{X}}\right)$

To determine the sensitivity of the objective function with respect to the vertices of the grid used for the discretization of the flow domain, using the dual number method, the coordinate of the vertex must be disturbed and the objective function must be evaluated. Based on the observations made in the preceding section, it suffices to evaluate only a limited part of the surface integral in order to compute the required sensitivity.

A single vertex affects the value of the pressure in the halo control volume of at most four halo control volumes for a single level. Therefore, for the evaluation of the surface integral, only four faces of the surface grid need to be considered. For this purpose, a function is created which computes the effect of disturbing the three components of the coordinate of a vertex on the pressure at the four faces affected[16]. This procedure involves: (i) recomputing the centre of mass of the control volumes, (ii) recomputing the coordinates of the vertices of the halo control volumes, (iii) performing the linear extrapolation of the pressure to obtain the pressure in the halo control volumes and

---

[16] Or fewer, when for instance a corner vertex is considered.

(iv) recomputing the surface normal vector at the face over which the pressure integral is to be performed. Then, the pressure integral is evaluated only for the faces affected. The sensitivity with respect to the coordinate of the vertex considered is subsequently found by considering the three components of the non-real part of the dual-number-valued result.

Extension to composite overset discretizations

As discussed in the preceding section, overlap of the surface grids can occur, when composite overset grids are used for the discretization of the flow domain. The zipper grids that have been used to facilitate the accurate evaluation of the surface integrals in these situations, depend on the grid used for the discretization of the flow domain. Therefore, when the sensitivity of the objective function with respect to the coordinate of a vertex on the surface of the geometry must be determined, the zipper grid must be taken into account as well.

The surface area of a triangle of the zipper grid, as well as its surface unit normal vector are only affected by the three vertices of the original surface grid to which the edges of the triangle are connected. The interpolation weight is however also affected by other vertices of the surface grid, since the interpolation weights are determined using the centres of four quadrilateral faces of the surface grid, as shown in picture 4.11 on page 65. Consequently, this influence also needs to be taken into account. Based on this observation, the approach taken to determine the effect of the zipper grid on the grid sensitivity of the objective function is therefore to reconstruct the zipper grid for each vertex that affects either the interpolation weight or the metric information of at least one triangular face. For this purpose, first all vertices are collected, that satisfy this requirement. Subsequently, for one of the collected vertices, the non-real component of the dual number that is an element of $\overline{\mathbb{D}}^3$ is assigned the value 1.0 for each of the coordinate directions — using a different component of the dual number for each coordinate direction. Then, the zipper grid is generated and the non-real components of the resulting interpolation weights and metric data are considered. For non-zero results, the information of the vertex for which a disturbance has been applied is stored, together with the information of the triangle for which the non-zero result was observed and the computed sensitivity information. This procedure is then repeated, until all vertices that were collected have been handled.

Finally, to take into account the contribution of the zipper grid to the grid sensitivity of the objective function, the pressure — with zero valued non-real components — of the quadrilateral faces that are used in the interpolation stencil of a triangular face are multiplied with the sensitivity data that has previously been computed and stored. This result is then added to the sensitivity that was already computed for the vertex that is being considered.

6.8 Design variable sensitivity of objective function $\left(\text{i.e. } \dfrac{\partial \mathcal{I}}{\partial \alpha}\right)$

Some design variables directly affect the value of the objective function for an aerodynamic shape optimization problem. An example of such a design variable is the angle of attack — for instance, when a 2D aerofoil shape optimization problem is considered, with the drag coefficient acting as the objective function. To compute the sensitivity of the objective function with respect to design variables like these, first a dual number copy of the converged flow solution is made, with the non-real part initialized as zero. Subsequently, the non-real part of the design variable considered is assigned the value 1.0. Then, the control volumes for which the flow solution is directly affected by the design variable are recomputed. This situation is usually only applicable to halo control volumes. Finally, the objective function is evaluated. The non-real part of the result gives the sensitivity of the objective function with respect to the design variable considered.

Note, that computing this sensitivity does not require any extension or modification in the case that a composite overset discretization is used instead, apart from the method used for computing the objective function. This modification, however, must also be made, even when computing this sensitivity is not considered.

6.9 Design variable sensitivity of grid $\left(\text{i.e. } \frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}\right)$

The approach taken to compute $\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}$ is to just construct a new grid for a geometric design variable with a non-real part of the dual number equal to one. Construction of a dual number hyperbolic field grid requires modification of the method used for solving the system of linear equations involved in the hyperbolic grid generation algorithm. The external software library PETSc is used for solving this system of linear equations. Although there is a possibility to use PETSc for solving systems of linear equations involving complex numbers, there is no such capability for dual numbers. To address this issue, the following approach is taken. Consider the dual-number-valued system of linear equations

$$\underline{\underline{A}}\boldsymbol{x} = \boldsymbol{b}, \tag{6.17}$$

with $\underline{\underline{A}} \in \mathbb{D}^{\mathrm{n}\times\mathrm{n}}$, $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{D}^{\mathrm{n}}$ and $\mathrm{n} \in \mathbb{N}_1$. By denoting

$$\begin{aligned} \underline{\underline{A}} &= \underline{\underline{A}}_1 + \varepsilon \underline{\underline{A}}_2, && \text{with } \underline{\underline{A}}_1,\ \underline{\underline{A}}_2 \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}, \\ \boldsymbol{b} &= \boldsymbol{b}_1 + \varepsilon \boldsymbol{b}_2, && \text{with } \boldsymbol{b}_1,\ \boldsymbol{b}_2 \in \mathbb{R}^{\mathrm{n}}, \\ \boldsymbol{x} &= \boldsymbol{x}_1 + \varepsilon \boldsymbol{x}_2, && \text{with } \boldsymbol{x}_1,\ \boldsymbol{x}_2 \in \mathbb{R}^{\mathrm{n}}, \end{aligned}$$

this system of equations can also be written as a larger coupled system of the real and non-real part, which now involves only real-valued numbers

$$\begin{bmatrix} \underline{\underline{A}}_1 & \underline{\underline{0}} \\ \underline{\underline{A}}_2 & \underline{\underline{A}}_1 \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix}. \tag{6.18}$$

Note, as observed before, that there is no influence of the non-real part $\boldsymbol{x}_2$ on the real-valued result $\boldsymbol{x}_1$. Moreover, observe that $\boldsymbol{x}_1$ is just the solution of the real-valued grid generation procedure and therefore already available, the equation can therefore be rewritten to

$$\underline{\underline{A}}_1 \boldsymbol{x}_2 = \boldsymbol{b}_2 - \underline{\underline{A}}_2 \boldsymbol{x}_1. \tag{6.19}$$

The result of this strategy is that the dimension of the system of equations that must be solved to obtain the dual number result, does not increase with respect to the real-valued grid generation procedure. Instead, an additional system of linear equations, of the same dimension as the original one, must be solved. However, since the computational effort required to solve a matrix equation is typically quoted as being on the order of the size of the matrix raised to some power $\bar{q} \in \mathbb{R}$ with $(\bar{q} > 1)$ — where the value of the exponent $\bar{q}$ depends on the algorithm used [60, 62] — solving the additional system of linear equations is more efficient than solving the larger system of equation (6.18). Moreover, because matrix $\underline{\underline{A}}_1$ appears both times in the left hand side of the equations, parts of the solution procedure can be reused. When the solution is, for instance, obtained using a preconditioned GMRES iterative procedure, the preconditioner matrix can be reused. Using a direct method instead, by means of LU decomposition, can be even more efficient, since the decomposition needs to be performed only once. Then, the solution of the non-real part is obtained by performing a forward and a backward substitution for respectively the lower and upper triangular matrix. By the application of a dual number with a non-real part consisting of multiple components, as in equation (6.2), this procedure can be repeated for a number of times equal to the dimension of the non-real part, resulting in an even more efficient algorithm. Since this approach involves only real-valued systems of linear equations, existing algorithms for solving real-valued systems can be used.

Once the complete dual number grid has been computed, using the procedure described above, the sensitivity of the grid with respect to the design variable that was given a non-real value of 1.0 is determined by considering the non-real part of the vertex coordinates of the resulting grid.

Extension to composite overset discretizations

The use of a composite overset discretization of the flow domain does not require additional work for the computation of the sensitivity of the grid with respect to a geometrical design variable. In the present approach, however, the product of $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}$ and $\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\alpha}$ is computed by the evaluation of the residual of the flow equations on a dual number grid — i.e. only when a composite overset discretization of the flow domain is used. Therefore, the implicit holecutting method must be carried out with dual numbers as well. Realizing this requirement is straightforward, using the template functions of C++. However, when the solution of a linear system of equations is required for determining the interpolation coefficients of the fringe cells, as discussed in subsection 4.5.4, some additional work must be performed. For this purpose, the same approach is used as for the generation of the dual number grid.

6.10 Solution method for the adjoint equations

The adjoint equations consist of a system of linear equations. The matrix involved in this system of linear equations is the transposed Jacobian matrix. Since transposing a square matrix does not alter its eigenvalues — because the transposed matrix shares the same characteristic polynomial as the untransposed one — the properties of the linear system of equations are the same as for the system of equations involved in Newton's method, used to obtain the flow solution. Therefore, a similar solution strategy can be used for solving the adjoint equations.

For the purpose of solving the adjoint equations, the transpose of the Jacobian matrix based on the first-order discretization of the convective flux is computed as well as the transpose of the Jacobian matrix based on the second-order discretization of the convective flux. The first-order matrix is used for the construction of the preconditioning matrix, for which an ILU decomposition is employed. Subsequently, the adjoint vector is obtained by solving the linear system of equations by means of a preconditioned GMRES method. This procedure is analogous to the method used to compute the increment of the flow solution in Newton's method. However, when the increment of the flow solution is computed, the iterative solution method is usually converged only a few orders of magnitude, while for the adjoint equations a more accurate solution is required, in order to obtain accurate gradients.

For constraint functions that also require the solution of the adjoint equations, the preconditioner matrix — constructed for computing the adjoint vector for the objective function — can simply be reused, since the left hand side matrix in both adjoint equations is exactly the same.

6.11 Verification of adjoint implementation

The accuracy of the implementation of the adjoint equation method is verified by using a different method to compute the gradients and comparing the result with the result obtained by means of the adjoint equation method. For the alternative approach, the flow solution method is treated as a function that has design variables as input parameters and the result of the objective function as output parameter. Subsequently, the dual number method is used to directly compute the derivatives of the objective function with respect to the design variables.

6.11.1 Approach

The approach taken is as follows. First, a fully converged real-valued flow solution is computed, denoted $\boldsymbol{\mathcal{U}}^* \in \mathbb{R}^{\mathrm{n_{cons}} \cdot \mathrm{n_c}}$. Then, one of the design variables is assigned a non-real value of unit magnitude. Subsequently, a new grid is generated and preprocessing of the grid is performed, including determining the overset block connectivity. For the initialization of the flow field for this dual-number-valued grid, the converged real-valued flow solution is used. Then, the residual vector $\boldsymbol{\mathcal{R}} \in \mathbb{D}^{\mathrm{n_{cons}} \cdot \mathrm{n_c}}$ of the governing equations is computed. As a converged flow solution is used for the initialization, the real-valued elements of the residual vector will have a magnitude close to that of machine precision. The non-real part of the solution is now obtained by employing the same approach as used for determining the design variable sensitivity of the grid, discussed in section 6.9. Since the real-valued solution has been fully converged, the system of linear equations does not need to be solved for the real part any more, which also means that the non-real part of the Jacobian matrix is not required. The non-real part of the flow solution — i.e. $\boldsymbol{\mathcal{U}}_2 \equiv \mathfrak{D}\left(\boldsymbol{\mathcal{U}}\right)$, $\boldsymbol{\mathcal{U}} \in \mathbb{D}^{\mathrm{n_{cons}} \cdot \mathrm{n_c}}$ — is obtained by solving

$$\underline{\underline{A}}_1\left(\boldsymbol{\mathcal{U}}^*\right)\boldsymbol{\mathcal{U}}_2 = -\mathfrak{D}\left(\boldsymbol{\mathcal{R}}\left(\boldsymbol{\mathcal{U}}^*\right)\right), \tag{6.20}$$

where $\underline{\underline{A}}_1 \in \mathbb{R}^{\mathrm{n_{cons}} \cdot \mathrm{n_c} \times \mathrm{n_{cons}} \cdot \mathrm{n_c}}$ is the real-valued Jacobian matrix, i.e. the non-real part is a linear problem even though the real part is non-linear. This approach means that both the real-valued Jacobian matrix as well as the preconditioner matrix — required for solving the system of linear equations — only need to be computed once and that

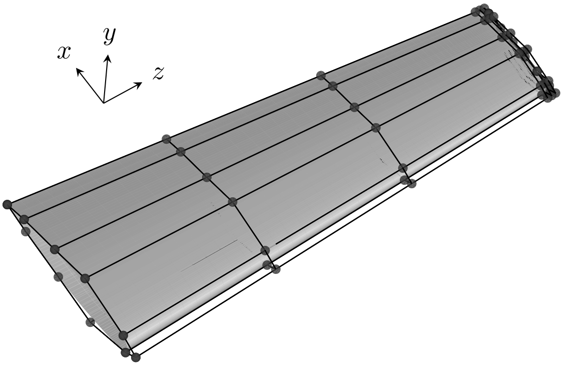


Figure 6.10: Control point mesh and corresponding surface geometry of the ONERA M6 wing, consisting of 11 control points per chordwise section. The wing is defined by 4 chordwise sections and an additional series of 3 × 11 control points is used to specify the tip-cap.

they can be reused for obtaining the non-real part of the flow solution for the subsequent design variables.

Once the flow solution has been obtained, the total derivative of the objective function, with respect to the design variable considered, is determined by evaluating the objective function for the dual-number-valued flow solution on the dual-number-valued grid and considering the non-real part of the result.

For the computation of the derivatives by means of the adjoint equation method, the residual of the iterative method used to solve the adjoint equations was required to converge 15 orders of magnitude with respect to its initial value.

6.11.2 Geometry and flow configuration

For the verification of the derivatives, the flow around the ONERA M6 wing, with a sharp trailing edge, is considered. The geometry is parametrized by means of a NURBS surface, using 11 control points per chordwise section and 4 control points in the spanwise direction of the wing, see figure 6.10. The z-component of the coordinate of every control point is used as an independent design variable, except for the final chordwise cross-section at the tip of the wing for which the z-component is the same for all control points. Moreover, the final cross-section remains symmetric, which means that only six control points are used for the design parameters. The two control points at the trailing edge of the wing coincide. The control point for the trailing edge of the root section remains at a fixed position. For the other control points, the x and y-component of the coordinate of the control point are independent. Apart from the design parameters describing the geometry of the wing, an additional parameter is used to specify the inflow angle of the free-stream with respect to the x-axis. Using the parametrization specified above, the number of design parameters amounts to 128, including the weights associated with the control points.

The flow conditions chosen for the verification are: a free-stream Mach number

Table 6.2: The number of cells used for the discretization of the flow domain around the ONERA M6 wing, used for the verification of the implementation of the adjoint equation method. For the body-fitted grid n_1 is in the circumferential direction, n_2 is in the spanwise direction and n_3 is in the direction normal to the surface of the wing. For the background grid n_3 is in the spanwise direction. For the single block results, only the body-fitted grid is used.

block	n_1	n_2	n_3
body-fitted	64	40	44
Cartesian background grid	50	50	20

of $\mathrm{M}_\infty = 0.30$ at an inflow angle of $\alpha = 3.00$ with respect to the x-axis.

Verification is performed for both a single block grid and for a composite overset grid, consisting of a body-fitted grid and a Cartesian background grid. The verification of consistency of the implementation does not require the same spatial resolution as required to accurately resolve the flow physics [132]. Therefore, coarser grids are used, to alleviate the computational requirements for this verification, since the dual number method is not very efficient for determining such a considerable number of total derivatives. The number of cells used for the discretization is listed in table 6.2.

6.11.3 Results

Functions considered for this verification are the lift and drag coefficient for the wing. The results are presented in table 6.3, by means of two different norms and the quadratic mean of the vector that represents the difference between the result obtained by means of the adjoint equation method and the result obtained by means of the dual number method. These norms are computed for both the absolute as well as for the normalized difference.

Considering the very small differences between the results computed with the two different methods used, it can be concluded that the gradients computed by means of the adjoint equation method are consistent; both for the single block as well as for the composite overset discretization of the domain. Moreover, it should be noted that the results appeared to be dependent on the optimization settings used for the compiler, indicating that the differences observed are probably a reflection of machine precision — for the double-precision floating-point arithmetic used — affecting the results.

6.12 Summary

In this chapter, the method used to efficiently compute the gradient of the objective function has been presented. Efficiency of the method is achieved by:

(i) using the discrete adjoint equation method, which limits the number of flow solutions that must be obtained to only one;

(ii) applying graph vertex colouring, which is used to compile independent sets, such that multiple partial derivatives can be computed simultaneously;

Table 6.3: Norms and the quadratic mean of the difference between the gradient of the force coefficients C_D and C_L, computed using the discrete adjoint equation method and the dual number method. Normalization has been performed using the results of the adjoint equation method.

C_D

discretization		$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_2$	$(\cdot)_\mathrm{rms}$
single block	absolute difference	$9.221 \cdot 10^{-14}$	$2.146 \cdot 10^{-13}$	$1.897 \cdot 10^{-14}$
	normalized difference	$2.749 \cdot 10^{-10}$	$2.982 \cdot 10^{-10}$	$2.635 \cdot 10^{-11}$
overset	absolute difference	$8.824 \cdot 10^{-14}$	$2.166 \cdot 10^{-13}$	$1.914 \cdot 10^{-14}$
	normalized difference	$2.399 \cdot 10^{-10}$	$3.160 \cdot 10^{-10}$	$2.793 \cdot 10^{-11}$

C_L

discretization		$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_2$	$(\cdot)_\mathrm{rms}$
single block	absolute difference	$3.206 \cdot 10^{-13}$	$6.506 \cdot 10^{-13}$	$5.750 \cdot 10^{-14}$
	normalized difference	$6.982 \cdot 10^{-11}$	$9.533 \cdot 10^{-11}$	$8.426 \cdot 10^{-12}$
overset	absolute difference	$3.704 \cdot 10^{-13}$	$6.922 \cdot 10^{-13}$	$6.118 \cdot 10^{-14}$
	normalized difference	$3.039 \cdot 10^{-11}$	$4.764 \cdot 10^{-11}$	$4.211 \cdot 10^{-12}$

(iii) using the dual number method in vector mode to compute the partial derivatives, which reduces the number of arithmetical operations that must be performed to obtain the result.

Note, that although a single flow solution is sufficient, the use of the adjoint equations does require the solution of an additional system of linear equations for the objective function. Moreover, some of the constraint functions also require the solution of a similar system of linear equations. The computational cost of solving the additional system of linear equations is comparable to performing a single Newton iteration, used for obtaining the flow solution. Therefore, when a significant[17] number of design variables is used — which is typically the case for aerodynamic shape optimization problems — the adjoint equation method is far more efficient than a method requiring a number of flow solutions proportional to the number of design parameters.

The discrete adjoint equation method has been chosen, because it provides derivatives that are consistent with the solution of the discretized governing equations and because of the relatively simple implementation, compared to the continuous approach for the adjoint equation method.

To be able to use the dual number method for computing the partial derivatives, the flow solution method has been implemented using the concept of function templates. Function templates are a concept in C++ — the programming language used — that

---

[17] i.e. the number of design parameters is considerably larger than the number of constraint functions requiring the solution of the adjoint equations.

allows for a function to be used with different data types. Therefore, these functions can be used with both ordinary floating-point numbers and with dual numbers.

For checking the correctness of the implementation of the discrete adjoint equation method, the derivatives have also been computed by the direct application of the dual number method, to compute the total derivatives. The results show that the gradients obtained by means of the adjoint equation method are consistent with those obtained by means of the dual number method. Based on this observation, the discrete adjoint equation method is considered to be correctly implemented in the flow solution method.

7

Optimization method and results

"Evolution has ensured that our brains just aren't equipped to visualise 11 dimensions directly. However, from a purely mathematical point of view it's just as easy to think in 11 dimensions, as it is to think in three or four."

— Stephen W. Hawking (1942 – present)

In the preceding chapters, the development of the different components of the optimization framework has been described. Moreover, the verification of the implementation of these components has been performed and the corresponding results have been presented. This chapter shows how these different parts are combined to a method that can be used for solving aerodynamic shape optimization problems. The capabilities of the optimization method are subsequently demonstrated by solving two different shape optimization problems. Thereafter, a discussion on solving an aerodynamic shape optimization problem involving a rotor blade is presented.

7.1 Introduction

A gradient-based approach is used in the present research for solving aerodynamic shape optimization problems. The computational costs associated with obtaining a discrete approximation to the solution of the partial differential equations that govern the flow, combined with the observation that a considerable number of design parameters is required to accurately describe a 3D geometry, resulted in the choice of using the discrete adjoint equation method to compute the gradients of the objective function and constraint functions. Figure 7.1 presents a flow chart of the optimization procedure that corresponds to the approach taken in the present research.

The optimization procedure starts with the definition of the optimization problem. This definition involves the specification of the objective function and the constraints that the design must satisfy. Moreover, since a gradient-based approach is employed, the choice of the initial design is important. This initial design provides the value of the design parameters that are fed into the optimization loop. Also the flow conditions for which the design is considered need to be specified.

A grid is generated based on the design parameters provided and the flow domain specified. This discrete representation of the flow domain is subsequently used to obtain a flow solution. Once the flow solution has been obtained, the objective function and constraint functions can be evaluated. Moreover, the partial derivatives, that are required to assemble the adjoint equations and the expression for the gradient involving the adjoint

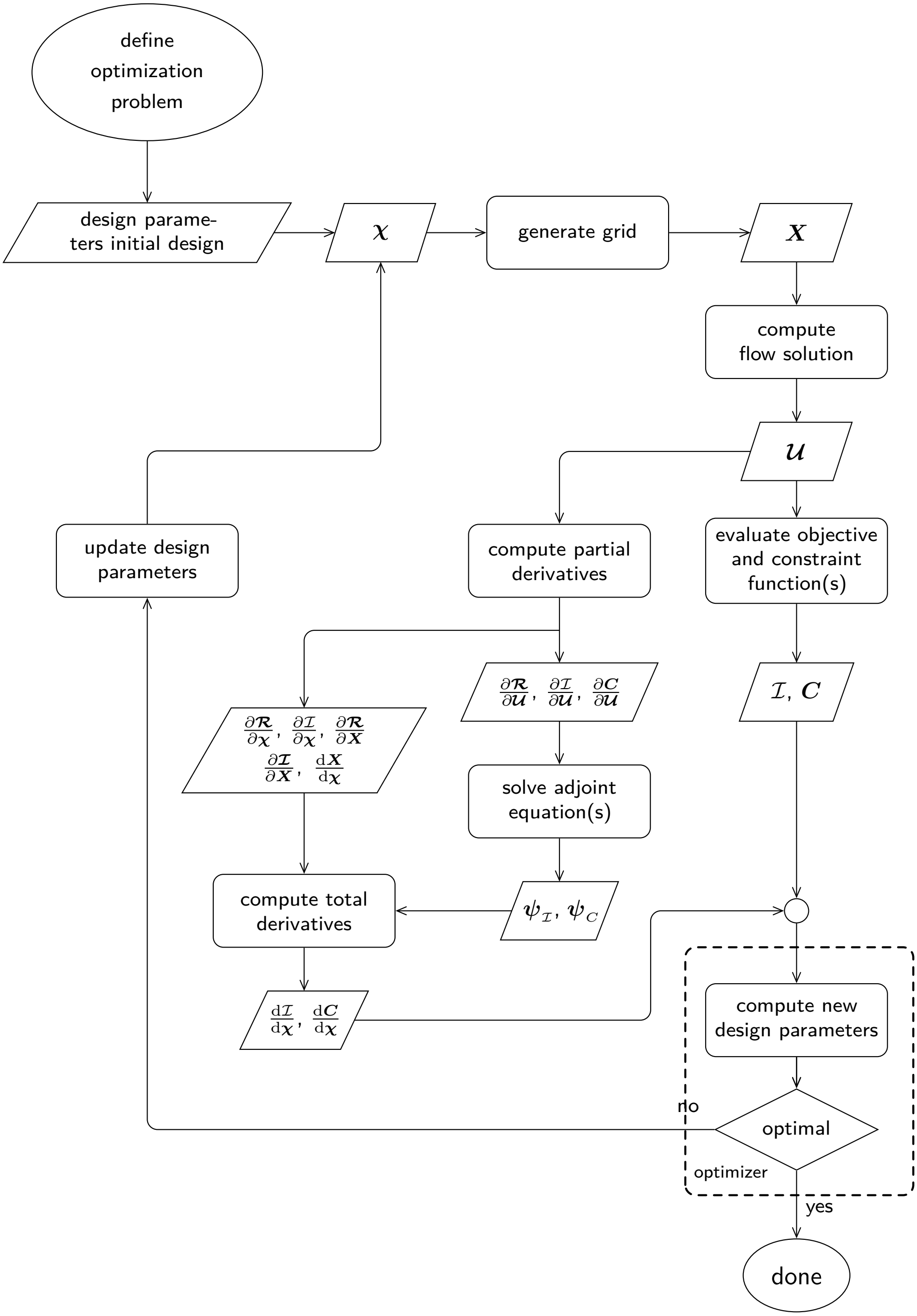

define optimization problem
design parameters initial design
$\boldsymbol{\chi}$
generate grid
$\boldsymbol{X}$
compute flow solution
$\boldsymbol{\mathcal{U}}$
update design parameters
compute partial derivatives
evaluate objective and constraint function(s)
$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\mathcal{U}}}, \frac{\partial \mathcal{I}}{\partial \boldsymbol{\mathcal{U}}}, \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\mathcal{U}}}$
$\mathcal{I}, \boldsymbol{C}$
$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\chi}}, \frac{\partial \mathcal{I}}{\partial \boldsymbol{\chi}}, \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{X}}, \frac{\partial \mathcal{I}}{\partial \boldsymbol{X}}, \frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}\boldsymbol{\chi}}$
solve adjoint equation(s)
compute total derivatives
$\boldsymbol{\psi}_{\mathcal{I}}, \boldsymbol{\psi}_{C}$
$\frac{\mathrm{d}\mathcal{I}}{\mathrm{d}\boldsymbol{\chi}}, \frac{\mathrm{d}\boldsymbol{C}}{\mathrm{d}\boldsymbol{\chi}}$
compute new design parameters
no
optimal
optimizer
yes
done

Figure 7.1: Flow chart of an adjoint-based aerodynamic shape optimization method. A square box with rounded corners represents a process or task, while a rhomboid-shaped box represents an input or output.

vector — i.e. equation (6.9) and (6.8), respectively — can now be computed. Once the adjoint equation for the objective function has been assembled, its solution is obtained iteratively using a GMRES method. The same applies for additional adjoint equations associated with each of the non-linear constraint functions that also depend on the flow solution. With the adjoint vector or vectors determined, the total derivative of the objective function can be computed.

The values computed for the objective function and for the constraint functions are then provided to the optimizer, together with the total derivatives. This information is subsequently used by the optimizer to calculate a new value for the design parameters and to determine if the optimization procedure has converged. In case the convergence criteria have not been met, the design parameters are updated and the whole procedure is repeated.

The optimization algorithm employed in the present research has been developed at Stanford University and is called SNOPT, which is an acronym for Sparse Nonlinear Optimizer [71]. This optimizer is a sequential quadratic programming (SQP) method that uses a smooth augmented Lagrangian merit function [70] to solve large-scale non-linear optimization problems. For the approximation of the Hessian matrix, required for performing the line search in the quadratic programming method, a BFGS quasi-Newton method is used [24, 25, 49, 64, 73, 163]. Although SNOPT is used in the present research as a 'black-box' with the objective function and constraint functions as input, together with their corresponding gradients, some details of the method are discussed here. This is necessary to explain the convergence histories of the optimization problems considered later on in this chapter.

The convergence of the optimization is measured by a quantity denoted as *optimality*, which represents the ratio between the largest component of the gradient of the Lagrangian merit function and the ℓ^2-norm of the vector of Lagrange multipliers. Moreover, the level of compliance to the constraints is denoted as *feasibility*. The feasibility is represented by the ratio of the largest violation of one of the non-linear constraints and the ℓ^2-norm of the vector of design variables. For details on the exact formulation of these two quantities, the original description of the algorithm can be consulted [71].

7.2 General aspects

This chapter first presents two different optimization problems, which have been solved to demonstrate the capabilities of the optimization method that has been developed in the present research. In the first optimization problem considered, the drag of a swept wing, subject to transonic flow conditions, is minimized for a fixed lift coefficient. In the second optimization problem, the wing span efficiency of a swept wing is maximized, also for a fixed lift coefficient. For both optimization problems, the initial geometry used is the same, being the ONERA M6 wing, which is a geometry that has been utilized before in this thesis. Because common aspects exist for both optimization problems considered, these general aspects are treated first.

7.2.1 Initial geometry and parametrization

For the initial geometry, a NURBS surface is used that approximates the ONERA M6 wing. The NURBS surface is the same as the one used for the verification of the implementation

Table 7.1: The number of cells used for the discretization of the flow domain around the swept wing, used in the optimization. For the body-fitted grid n_1 is the number of cells around a chordwise section, n_2 is the number of cells in the spanwise direction and n_3 is the number of cells in the direction normal to the surface of the wing. For the background grids n_1 is in the chordwise direction, n_3 is in the spanwise direction and n_2 is in the third direction, perpendicular to both the chordwise and spanwise direction. The bottom row lists the total number of control volumes used for the discretization.

block	n_1	n_2	n_3
body-fitted	192	80	44
uniform Cartesian	50	50	50
refined Cartesian	60	60	30
total		908 840	

of the adjoint equation method, which is depicted in figure 6.10 on page 142. Therefore, the wing geometry is represented by 4 different chordwise sections, each defined by 11 control points. The control points of each section have the same z-component. To close the wing at the tip, a tip-cap is added. Since the geometry of the wing changes during the optimization procedure, the geometry of the tip-cap must also adapt accordingly, in order to have the tip-cap fit properly to the wing. For that purpose, the geometry of the tip-cap is also represented by the same NURBS surface that represents the wing geometry, by adding additional sections of control points to the NURBS surface definition. The locations of the control points that represent the tip-cap are determined based on the geometry of the wing, in particular the geometry of the last section that specifies the wing geometry. Since the control points specifying the tip-cap directly depend on the rest of the control points of the NURBS surface, the coordinates of the control points of the tip-cap are not actual design variables. Consequently, when a non-real disturbance is applied to one of the control points of the wing — in order to compute the sensitivity with respect to that control point — which affects the geometry of the tip-cap, the method used for determining the location of the control points that specify the tip-cap must also be applied; in order to obtain the correct value of the sensitivity.

7.2.2 Flow domain and discretization

For the discretization of the flow domain, a composite overset grid is used, similar to the one used for the validation of the flow solution method, discussed in section 5.9.1 on page 114. The discretization of the flow domain consists of a body-fitted grid, which extends to 4 times the mean aerodynamic chord length from the wing surface and two Cartesian background grids. The far-field boundaries of the second Cartesian background grid are situated at about 5 times the wing span from the tip of the wing and 100 times the mean aerodynamic chord upstream and downstream of the wing. The block dimensions, in terms of the number of cells, for the different blocks, are listed in table 7.1.

7.2.3 Geometrical constraints

In order to obtain a feasible result from the optimization, a number of constraints are imposed. These constraints concern the geometry itself, but also constraints have been imposed on the control points. The latter have been used to prevent the occurrence of extreme, unrealistic geometries, which might result in the optimization procedure to terminate; either because the grid generation method fails to construct a valid body-fitted grid or because the flow solution method fails to obtain a converged flow solution. For this purpose, the distance between the control points must be larger than or equal to a certain minimum value. Moreover, the angle between two consecutive line segments of the control polygon — see figure 2.2 on page 24 — of the chordwise sections that define the wing shape, should be smaller than or equal to a certain value, for every control point.

Concerning the further geometrical constraints, the planform area of the wing is forced to be between an upper and a lower bound. The internal volume of the wing is constrained to be larger than or equal to the volume of the initial wing. The maximum curvature occurring somewhere on the geometry is constrained to be smaller than a specified value. This constraint has been imposed to prevent the leading edge from becoming too sharp. Moreover, the trailing edge angle is constrained to be larger than or equal to a specified value. This constraint function is evaluated for each section used in the parametrization of the geometry and has been imposed to prevent the geometry from becoming too thin towards the trailing edge.

Volume constraint

For the computation of the volume constraint, the surface definition of the wing geometry by a NURBS surface is exploited. Gauss' divergence theorem states that

$$\int\limits_{V} \nabla \cdot \boldsymbol{F}(\boldsymbol{x})\, \mathrm{d}V = \int\limits_{\partial V} \boldsymbol{F}(\boldsymbol{x}) \cdot \boldsymbol{n}\, \mathrm{d}S, \tag{7.1}$$

where $\boldsymbol{n}$ is the outward pointing unit normal vector and $\boldsymbol{F}(\boldsymbol{x})$ an arbitrary continuously differentiable vector field defined on a neighbourhood of V. Note, that the NURBS surface representing the wing is open at the symmetry plane. Since the only non-zero component of the normal vector of the symmetry plane is in the z-direction, $\boldsymbol{F}(\boldsymbol{x})$ is chosen to be $(x, y, 0)^T$ so that $\boldsymbol{\nabla} \cdot \boldsymbol{F} = 2$. This choice for $\boldsymbol{F}(\boldsymbol{x})$ warrants that the open part of the NURBS surface has no contribution to the surface integral and does therefore not need to be taken into account. Using the divergence theorem and the present choice for $\boldsymbol{F}(\boldsymbol{x})$, the volume of the wing can be expressed as

$$V_{\text{wing}} = \frac{1}{2} \int\limits_{\partial V} (x n_{\mathrm{x}} + y n_{\mathrm{y}})\, \mathrm{d}S. \tag{7.2}$$

Transformation of this surface integral to the parametric variables of the NURBS surface, yields

$$V_{\text{wing}} = \frac{1}{2} \int\limits_{v=0}^{1} \int\limits_{u=0}^{1} (x n_{\mathrm{x}} + y n_{\mathrm{y}}) \left\| \frac{\partial \boldsymbol{S}^{\mathrm{pq}}}{\partial u} \times \frac{\partial \boldsymbol{S}^{\mathrm{pq}}}{\partial v} \right\| \mathrm{d}u\, \mathrm{d}v. \tag{7.3}$$

This expression is evaluated by means of numerical integration. For this purpose, the parameter space is uniformly discretized in both directions with 200 elements. Subsequently, a four-point Gauss-Legendre quadrature rule [2] is used, to compute the contribution of the integral for each discrete surface element.

7.3 Drag minimization in transonic flow

This optimization problem considers the transonic flow around a swept wing. For the initial design — for which the ONERA M6 wing is used — shock waves are present at the suction side of the wing, which have a significant contribution to the total drag experienced by the wing. The aim of the present optimization is therefore to reduce the drag. The drag minimization of a swept wing with the ONERA M6 wing as initial geometry is widely used for testing aerodynamic optimization methods [31, 112, 179]. Although the transonic flow regime encountered in this situation is not representative for the flow around a wind turbine rotor, it perfectly serves the purpose of demonstrating the capabilities of the present method for solving aerodynamic shape optimization problems.

7.3.1 Definition of optimization problem

For the present optimization problem transonic flow conditions are considered, with a free-stream Mach number of $\mathrm{M}_\infty = 0.84$. The flow is at an initial inflow angle of $\alpha = 3.00^\circ$. These conditions are close to the conditions used for the validation case, presented in chapter 5.

The objective function considered concerns the minimization of the drag coefficient, defined by equation (5.80). Furthermore, an equality constraint is imposed on the lift coefficient, defined by equation (5.79), also on page 117. For this constraint the value of the lift coefficient is the value obtained for the initial design at the specified flow conditions and the discretization used, i.e. $C_\mathsf{L} = 2.8297 \cdot 10^{-1}$. The latter is done, because the grid used for solving the optimization problem is coarser than required for obtaining a grid-converged result.

The design variables used are the x and y-component of the coordinates of the control points. To prevent the optimization method from failing — for reasons previously mentioned — bounds have been imposed on these design variables. The x-component of the coordinate of each control point has been limited to take a value between $(x_0 - 0.05\, \bar{c}_{\mathsf{loc}})$ and $(x_0 + 0.05\, \bar{c}_{\mathsf{loc}})$, where $\bar{c}_{\mathsf{loc}}$ is the chord length of the section considered and x_0 is the initial value at the start of the optimization. Similarly, the y-component of the coordinate of each control point has been limited to take a value between $(y_0 - 0.015\, \bar{c}_{\mathsf{loc}})$ and $(y_0 + 0.015\, \bar{c}_{\mathsf{loc}})$. For the angle of attack the constraint is that it stays within the range $-4^\circ \leq \alpha \leq 4^\circ$. The number of design parameters used in this optimization amounts to 71.

The default settings for obtaining the flow solution are to first perform 16 000 Runge-Kutta iterations. Subsequently, Newton's method is used to obtain the fully converged flow solution, for which a default of 12 iterations is used. The system of linear equations for a Newton iteration is converged two orders of magnitude.

For the optimization the convergence criteria have been set to 10^{-5} for the optimality tolerance and 10^{-6} for the feasibility tolerance.

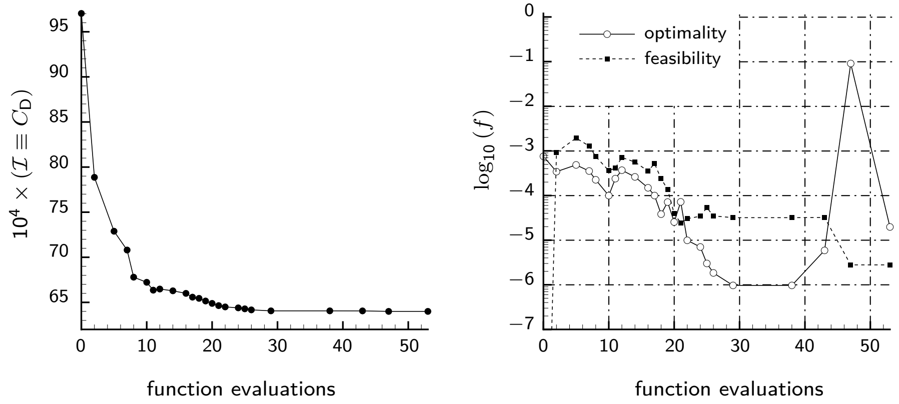


Figure 7.2: Convergence history for the minimum drag coefficient optimization problem. Left plot shows the evolution of the objective function. Right plot depicts both the optimality and feasibility of the design. Only major iterations are represented by a marker.

Table 7.2: Breakdown of an optimization iteration in different tasks with the corresponding timing. Calculation performed on two Intel Xeon X5650 processors, with each 6 cores, running at 2.67 $[\mathrm{GHz}]$. The presented wall clock time has been averaged over 10 optimization iterations.

component	absolute time $[\mathrm{s}]$		fraction	
grid	57.96		0.0040	
flow solution	9 721.38		0.6681	*1.000*
Runge-Kutta		*8 984.13*		*0.924*
Newton		*737.25*		*0.076*
Jacobian matrix	15.11		0.0010	
adjoint vectors	803.08		0.0552	
geometrical constraint sensitivity	15.22		0.0010	
$\partial\mathcal{I}/\partial\boldsymbol{X}$ and $\partial\mathcal{C}/\partial\boldsymbol{X}$	1.40		0.0001	
$(\partial\boldsymbol{\mathcal{R}}/\partial\boldsymbol{X})\,(\mathrm{d}\boldsymbol{X}/\mathrm{d}\boldsymbol{\chi})$ (design variable sensitivity of residual)	3 919.82		0.2694	
miscellaneous	17.10		0.0012	
total	14 551.07		1.0000	

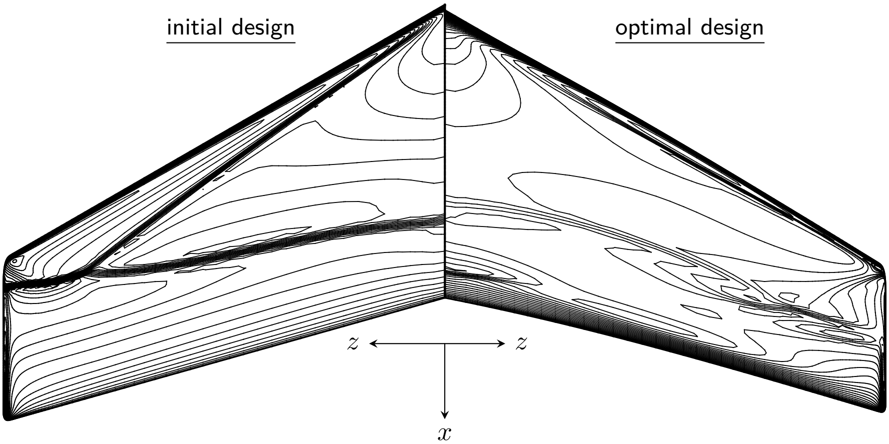
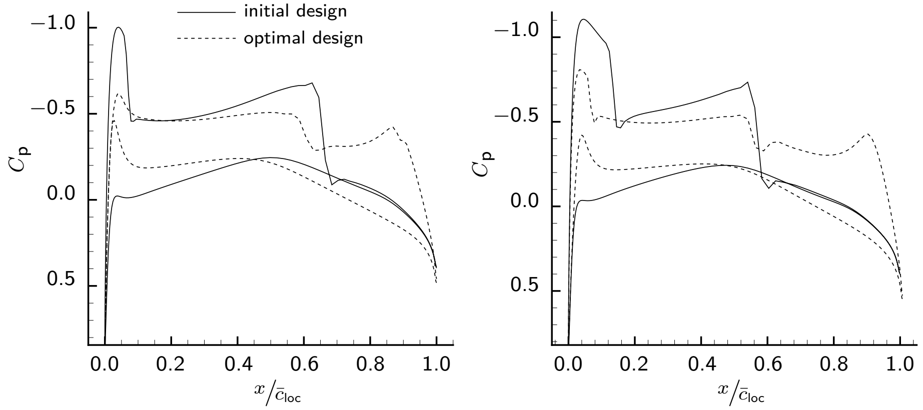


Figure 7.3: Iso-C_p contours on the surface of the wing. Free-stream Mach number of $\mathrm{M}_\infty = 0.84$, $C_\mathsf{L} = 2.8297 \cdot 10^{-1}$.

7.3.2 Results and discussion

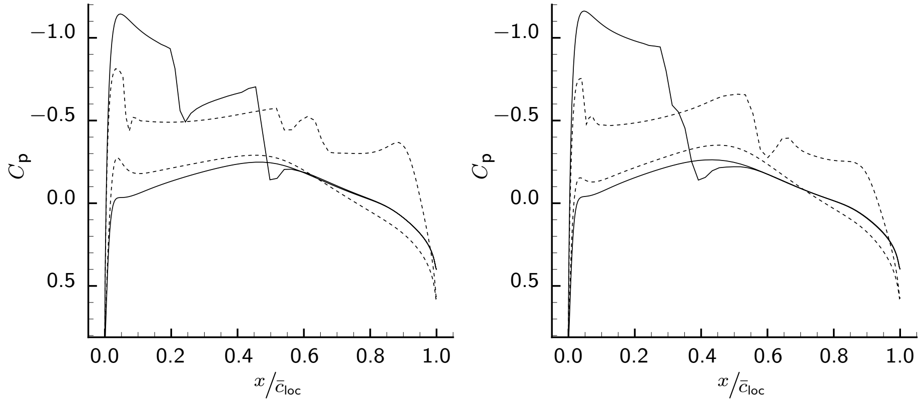
Figure 7.2 shows the convergence behaviour of the optimization problem. The left plot shows the convergence history for the objective function. The abscissa represents the number of evaluations of the objective function, which means that the flow solution must be recomputed. Although the optimizer does not require the gradients to be computed for every new evaluation of the objective function, usually the gradients are required to be recomputed as well. When the gradients are required, two adjoint equations are solved, one for the objective function itself and an additional adjoint equation for the lift coefficient, which acts as an equality constraint. The optimization algorithm sometimes requires the objective function to be evaluated for the so-called minor iterations as well. However, for these minor iterations the value of the objective function is not plotted in the graph; only the value of the objective function for the major iterations is depicted. The convergence history shows that in the first major iteration a large improvement in the value of the objective function is achieved. Subsequently, in the remaining 22 major iterations, a more gradual reduction is observed. In the optimization the drag coefficient reduces from its initial value of $9.703 \cdot 10^{-3}$ to $6.393 \cdot 10^{-3}$ for the optimal solution, which is a reduction of 34.1%. The right plot in figure 7.2 shows the convergence history for both the optimality and the feasibility. The convergence of the feasibility indicates that some difficulties are encountered in satisfying the constraints. Investigation revealed that the curvature constraint is the cause for this behaviour.

Figure 7.3 compares the planform shape and the iso-contour lines of the pressure coefficient for the suction side of the wing, for the initial design and for the geometry corresponding to the optimal design. These results show that the strong shocks present for the initial design have nearly vanished for the optimal result. Only a weak shock halfway the chord close to the tip of the wing remains. Moreover, close to the trailing edge a weak shock is present, that ranges from the root of the wing to about 60 % of the span. A more detailed view of the pressure coefficient distribution can be found in figure 7.4, which compares the pressure coefficient distribution for different spanwise locations. These graphs also clearly show the significant reduction of the shock on the



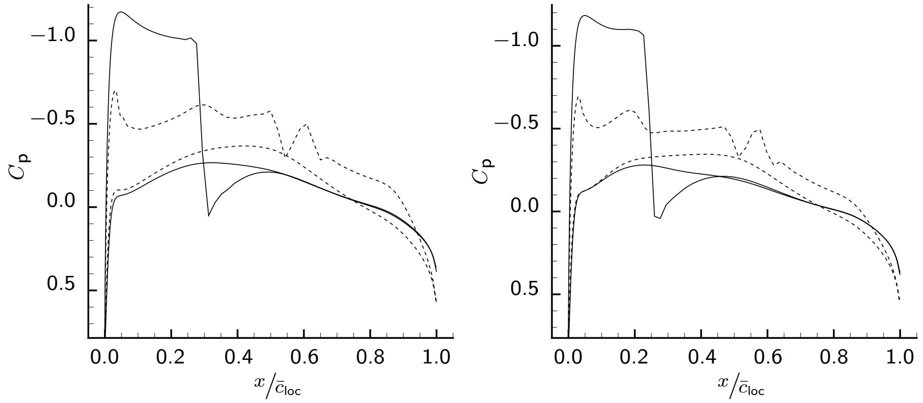
(a) $2z/b = 0.20$

(b) $2z/b = 0.44$



(c) $2z/b = 0.65$

(d) $2z/b = 0.80$



(e) $2z/b = 0.90$

(f) $2z/b = 0.95$

Figure 7.4: Comparison between the numerical results for the flow around a swept wing subject to a free-stream Mach number of $\mathrm{M}_\infty = 0.84$ and lift constraint of $C_\mathrm{L} = 2.8297 \cdot 10^{-1}$, for the initial design and for the result of solving the optimization problem of minimizing the drag coefficient.

suction side of the wing.

Table 7.2 on page 153 lists a breakdown in different tasks of an optimization iteration and the corresponding timing for each of the tasks. The results have been obtained by averaging over 10 separate optimization iterations, that required both the objective function and the gradient to be computed. These results show that computing the flow solution has a major contribution in the overall time required for a single optimization iteration. In particular performing the explicit Runge-Kutta pseudo-time-steps takes a long time. Therefore, it might be more suitable to use a different globalization strategy, i.e. the method used to determine a suitable initial guess for Newton's method, that is more efficient in terms of computation time required. Moreover, the computation of the grid sensitivity has also an important contribution in the computing time. Solving the two adjoint equations, on the other hand, has only a minor contribution to the total computing time, which indicates that it has been a good choice to use the adjoint equation method for the gradient computation.

7.4 Wing span efficiency maximization

In order to also consider an optimization problem for which the flow conditions are more representative for the flow around a wind turbine rotor, the following optimization problem for subcritical flow conditions is considered. The optimization problem concerns the subcritical flow around a swept wing and the objective is to maximize the wing span efficiency e.

7.4.1 Definition of optimization problem

The wing span efficiency, used in this optimization to be maximized, is defined as [7]

$$e := \frac{C_{\mathsf{L}}^2 S}{\pi b^2 C_{\mathsf{D}}}, \tag{7.4}$$

where b denotes the full wing span and S represents the planform area of the full wing. The subcritical flow conditions used are a free-stream Mach number of $\mathrm{M}_\infty = 0.30$ and the initial inflow angle is $\alpha = 3.00^\circ$. For the maximization of the wing span efficiency, an equality constraint is imposed on the lift coefficient. As in the preceding optimization problem, for this constraint the value is taken equal to the value of the lift coefficient obtained for the initial design at the specified flow conditions and the discretization used, i.e. $C_{\mathsf{L}} = 2.0200 \cdot 10^{-1}$.

The design variables used are the y-component of the coordinates of the control points and the z-component for the 11 control points of each section, resulting in a total of 39 design parameters, including the angle of attack. To prevent the optimization method from failing — for reasons previously mentioned — bounds have been imposed on these design variables. The y-component has been limited to take a value between $(y_0 - 0.05\ \bar{c}_{\mathsf{loc}})$ and $(y_0 + 0.05\ \bar{c}_{\mathsf{loc}})$. Similarly, the z-component has been limited to take a value between $(z_0 - 0.25\ b)$ and $(z_0 + 0.25\ b)$, where b is the wing span for the initial design. Since the z-component is a design variable, an additional geometrical constraint is imposed on the wing span, for which the value of the semi-span should be between 1.00 $[\mathrm{m}]$ and 1.30 $[\mathrm{m}]$. For the angle of attack the permitted range is $-5^\circ \leq \alpha \leq 5^\circ$.

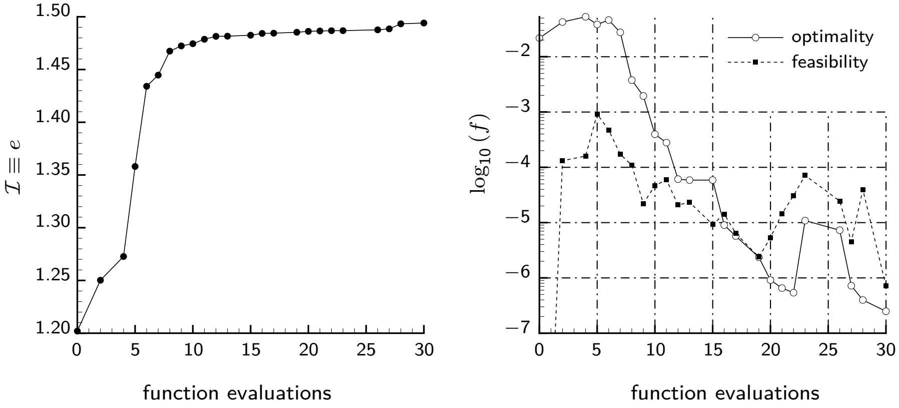


Figure 7.5: Convergence history for the maximum wing span efficiency optimization problem. Left plot shows the evolution of the objective function. Right plot depicts the optimality and feasibility of the design. Only major iterations are represented by a marker.

The default settings for obtaining the flow solution are to first perform 16 000 Runge-Kutta iterations. Subsequently, Newton's method is used to fully converge the steady flow solution, for which a default of 16 iterations is used. Again, the system of linear equations for a Newton iteration is converged two orders of magnitude.

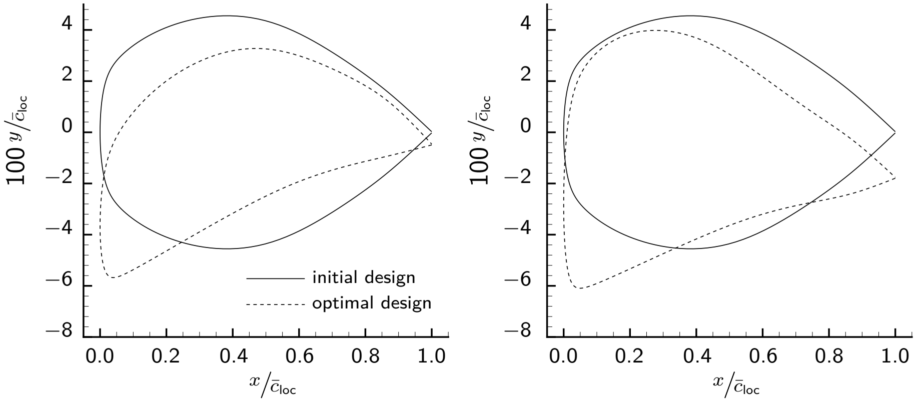
For solving the optimization problem, the same convergence criteria have been used as for the previous optimization problem, i.e. optimality tolerance of 10^{-5} and feasibility tolerance of 10^{-6}.

7.4.2 Results and discussion

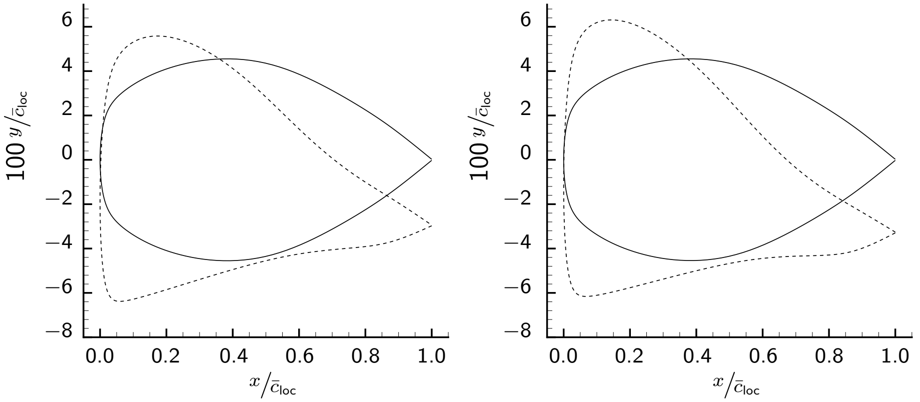
The convergence behaviour for the present optimization problem is presented in figure 7.5. Considering the left plot, similar behaviour as for the transonic drag-minimization problem is observed. In the first few iterations a large change in the value of the objective function is realized, while towards the end only very minor improvements are observed. Regarding the convergence history of the feasibility, no problems in satisfying the constraints are observed for this optimization problem.

As with the transonic drag optimization problem, two adjoint equations are solved for computing the gradients. One adjoint vector is required for efficiently computing the gradient of the lift coefficient. The gradient for the objective function is, however, not computed using an efficiency factor specific adjoint vector. Instead, the gradient of the drag coefficient is computed by means of the adjoint equation method. The product rule and chain rule of differentiation are subsequently used to compute the gradient of the wing span efficiency. With this approach, the already implemented and verified methods for computing the gradient of C_D could be reused, which saved time required for the implementation and verification of the method.

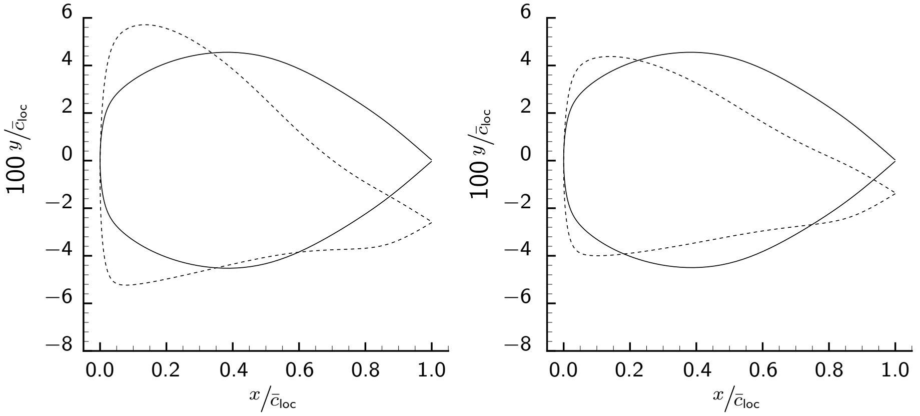
A striking result of the solution of this optimization problem is that the final value of the objective function considerably exceeds 1.0. Based on lifting line theory for a high-aspect-ratio low-sweep thin wing, the value of 1.0 is considered optimal. However, some assumptions in lifting line theory, such as the wake being planar, do not apply to



(a) $^{2z}/_{b} = 0.05$ (b) $^{2z}/_{b} = 0.20$



(c) $^{2z}/_{b} = 0.40$ (d) $^{2z}/_{b} = 0.60$



(e) $^{2z}/_{b} = 0.80$ (f) $^{2z}/_{b} = 0.95$

Figure 7.6: Comparison of the geometry of the initial design and the geometry resulting from solving the optimization problem. For the maximization of the wing span efficiency at a free-stream Mach number of $\mathrm{M}_\infty = 0.30$ and lift constraint of $C_{\mathrm{L}} = 2.0200 \cdot 10^{-1}$. Note, that for b the value of the initial design is used.

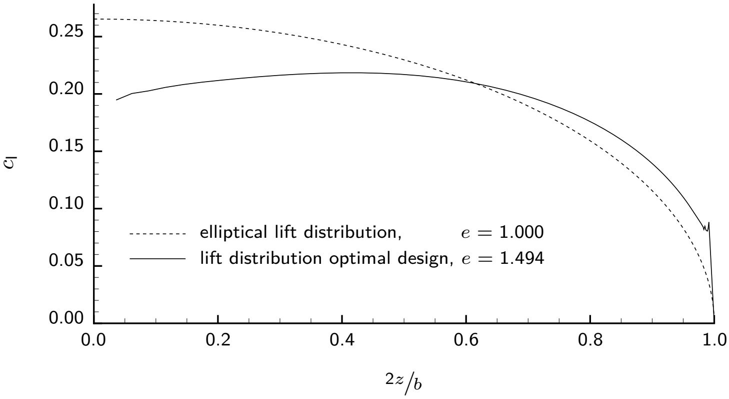


Figure 7.7: Lift distribution of optimal design for the wing span efficiency maximization problem, $\mathrm{M}_\infty = 0.30$ and $C_\mathrm{L} = 2.0200 \cdot 10^{-1}$. Shown is also the elliptical lift distribution with identical lift coefficient.

the present design. Therefore, obtaining a value exceeding 1.0 is possible, although the present value is still considered quite large. A possible cause for this result may be the location of the formation of the tip vortex not matching the one that will appear in true viscous flow, as suggested by Hicken [83].

When the spanwise lift distribution is considered, depicted in figure 7.7, it is clear that the lift does not show an elliptical distribution; which was already clear from the wing span efficiency not being equal to 1.0. More lift is produced towards the tip, while near the root the lift is lower than for an elliptical loading. Therefore, the root bending moment is larger for the present optimal design, which is not a desirable property from a structural point of view. To prevent such behaviour, it is recommended to include a constraint on the root bending moment in the optimization.

Figure 7.6 compares the aerofoil sections of the initial design and the ones from the final shape of the wing at different spanwise locations. This figure shows that the geometry has been changed considerably. Twist of the wing has been introduced in the root section. Moreover, halfway the span of the wing, the aerofoil sections have become thicker towards the leading edge, while thickness towards the trailing edge has decreased. A further decrease in thickness was limited by the constraints on the design variables, in particular the y-component of the coordinates of the control points. Although, a number of geometrical constraints have been imposed to enforce a certain thickness, such as the trailing edge angle constraint and the volume constraint, they did allow for the wing to be locally very thin. Therefore, instead of limiting the design variables, it might be more direct to employ constraints on the chordwise thickness distribution that can also locally enforce sufficient thickness of the wing at all locations.

Figure 7.8 compares the planform shape and the iso-contour lines of the pressure coefficient for the suction side of the wing, for the initial design and for the geometry corresponding to the optimal design. These results show that the wing span has increased; to the value specified by the constraint. Moreover, the leading edge and trailing edge are no longer straight lines. One more remark on the feasibility of the current optimal design concerns the considerable amount of twist introduced. A strongly twisted wing has a limited applicability for a real airplane, because the twist unloads the wing near

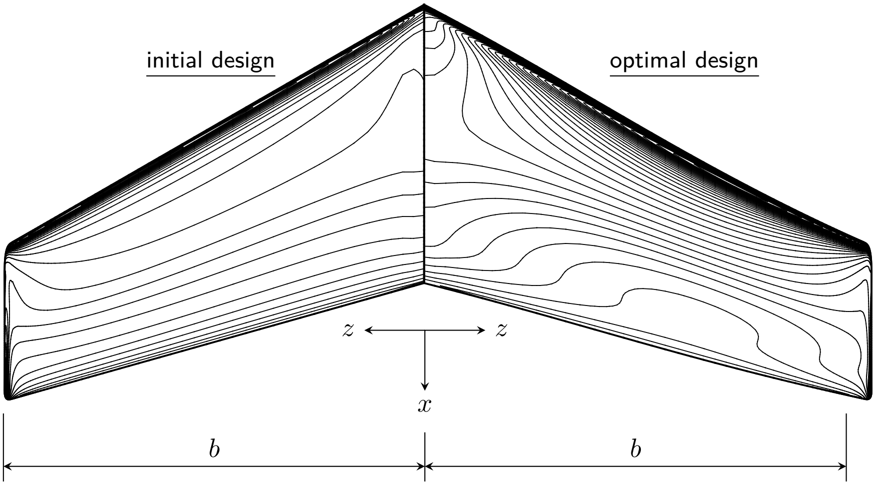


Figure 7.8: Iso-C_p contours on the surface of the wing. Free-stream Mach number of $\mathrm{M}_\infty = 0.30$, $C_\mathrm{L} = 2.0200 \cdot 10^{-1}$. Note, that the optimal design has a larger span than the initial design.

the root while it undesirably increases the loading near the tip. This effect might lead to premature tip stall [168].

7.5 On solving a rotor blade optimization problem

The results presented in the two preceding sections demonstrate the capabilities of the optimization method developed in the present research. They show that the method can be used successfully for solving aerodynamic shape optimization problems for 3D geometries, using composite overset grids for the discretization of the flow domain. The next step is to apply the method for solving an aerodynamic shape optimization problem involving a blade of a rotating wind turbine rotor.

7.5.1 Geometry and parametrization

For this optimization problem, an isolated rotor is considered. The geometry used is based on the blade of the NREL phase VI rotor [68]. This rotor has been tested in the 24.4 $[\mathrm{m}] \times 36.6$ $[\mathrm{m}]$ NASA Ames wind tunnel. The rotor is part of a 10.06 $[\mathrm{m}]$ diameter stall regulated wind turbine, for which amongst others, blade surface pressures and local angles were measured at five spanwise locations. The actual geometry of the rotor used in the wind tunnel tests is quite complex; especially in the hub region, because of the instruments and control mechanisms used. These detailed features have been discarded and an ellipsoidal spinner is introduced instead. This approach prevents requiring the discretization of an unnecessarily complex domain. Moreover, the configuration with an ellipsoidal spinner more accurately resembles the geometry of a modern commercial horizontal axis wind turbine rotor.

The rotor blade has linear taper and is twisted. The cross-section for the blade is from root to tip defined by the NREL S809 aerofoil [181]. At the hub, the blade has a

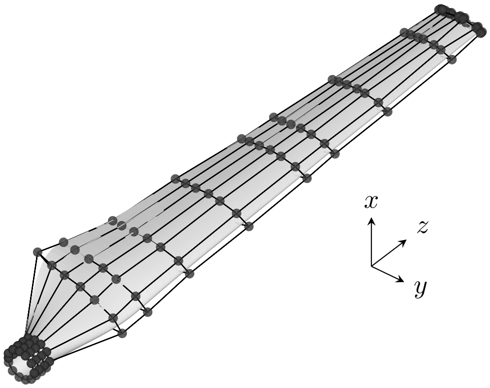


Figure 7.9: Control points and corresponding surface geometry of the NREL Phase VI rotor blade, consisting of 13 control points per chordwise section. The blade is defined using 13 sections. An additional series of 3 × 13 control points is used to specify the tip-cap.

circular cross-section, which extends to a radius of 0.724 $[\mathrm{m}]$. Then, there is a linear transition from the circular cross-section to the NREL S809 aerofoil from 0.724 $[\mathrm{m}]$ to a radius of 1.257 $[\mathrm{m}]$. The blade has a finite chord length at the tip. A rounded tip-cap is added to the tip.

For this blade geometry, a NURBS surface representation has been obtained, using the NURBS surface fitting method discussed in chapter 2. The NURBS surface uses 13 control points per cross-section. The blade is defined by 13 different cross-sections. The coordinates of the control points of a single section have the same z-component. Moreover, for the definition of the tip-cap an additional 3 sets of control points are used. The NURBS surface with the corresponding grid of control points is presented in figure 7.9. There are four sections used to represent the circular section near the hub. The control points of these four sections are not used as design variables in the optimization. These four sections are required to make sure that the geometry remains circular in the region where the blade intersects with the spinner when the location of the control points of the fifth cross-section is modified during the optimization process. For the construction of the surface grid on the blade and on the spinner, the intersection between the blade and the spinner is determined. The curve representing this intersection is used as one of the boundary curves for the surface grid.

7.5.2 Flow conditions

The flow conditions used for the present optimization correspond to standard sea level conditions. The inflow velocity is 10 $\left[\mathrm{ms}^{-1}\right]$ and the rotor plane is perpendicular to the uniform inflow. The rotor blade has a pitch angle of 0° and rotates at an angular velocity of 7.5 $\left[\mathrm{rads}^{-1}\right]$, which corresponds to approximately 72 $[\mathrm{rpm}]$. At this angular velocity the tip speed is 37.7 $\left[\mathrm{ms}^{-1}\right]$.

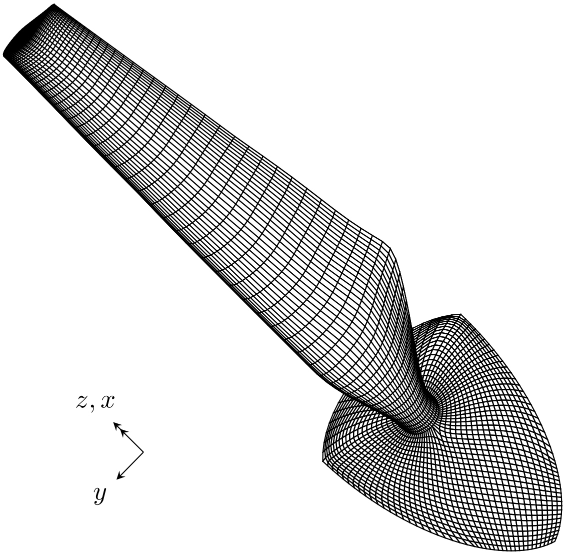


Figure 7.10: Surface grids on the rotor blade and the spinner. The blade surface grid consists of 128 × 130 cells. The spinner surface grid consists of 176 × 40 cells. Only every other grid line is shown.

7.5.3 Flow domain and discretization

The flow conditions specified in the preceding subsection, will result in a rotationally periodic flow solution. It is therefore permitted to compute the flow solution for a sector containing a single rotor blade with rotationally periodic boundary conditions at its sides. For this single blade, a body-fitted grid is constructed, that extends to approximately 0.80 $[\mathrm{m}]$ from the surface of the blade. The surface grid consists of 128 cells around each chordwise section and 130 cells in spanwise direction. For the spinner a body-fitted grid is constructed, which extends to 2.00 $[\mathrm{m}]$ from the surface of the spinner. The surface grid on the blade and the spinner are shown in figure 7.10. In the direction upstream of the spinner, a rhombus-shaped grid is used, that extends all the way to the inflow boundary, which is situated at approximately 20 times the rotor radius away from the centre of the blade. Downstream of the spinner is a point-matching cylindrical grid, that extends to approximately 20 rotor radii, to the location where the downstream boundary is situated. A so-called collar grid [136] is used to accurately represent the flow domain in the region of the intersection of the blade with the spinner. Additional cylindrical background grids are used in radial direction for the discretization of the domain towards the far-field boundary, which is situated at 9 rotor radii from the tip of the blade. Including the body-fitted grids, the total number of blocks used for the discretization of the domain amounts to 11, which have a combined number of 2 736 260 cells. To accommodate the evaluation of surface integrals and for the ray-casting, two zipper grids are constructed. These zipper grids connect the surface grid for the collar grid to the surface grid on the rotor blade and to the surface grid on the spinner, similar as in figure 4.10 on page 63.

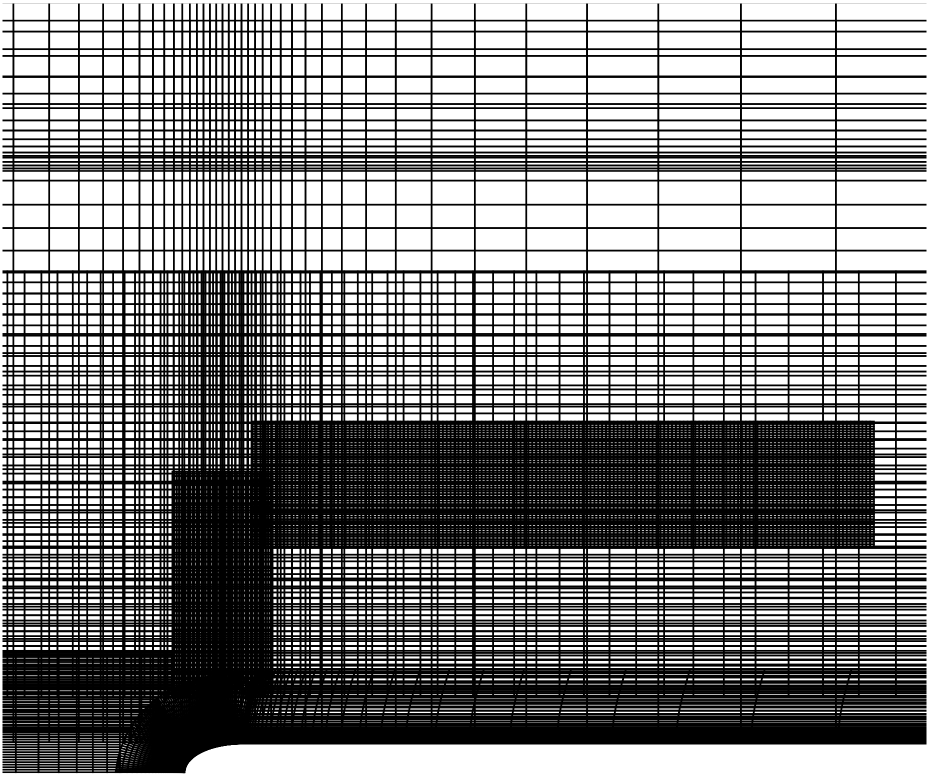


Figure 7.11: Detail of the grid on the periodic boundary. Note the higher grid resolution used to accurately resolve the helical vortex.

Subsonic outflow boundary conditions have been imposed for the outer boundary downstream of the rotor and the outer boundary in the radial direction. For the boundaries on the sides of the sector of the circular cylinder, periodic boundary conditions have been imposed. Upstream of the rotor, far-field boundary conditions are used, to represent the uniform incoming wind. A schematic representation of the flow domain is depicted in figure 7.13.

7.5.4 Objective function and constraints

One of the most important aspects of the optimization problem is the objective function for which the optimization is performed. The objective function defines — together with the parametrization employed — the design space in which the optimum is searched for. Moreover, the objective function used largely dictates the properties of the final design that is obtained by solving the optimization problem.

Considering the purpose of a wind turbine, i.e. extracting energy from the wind, it has been chosen to optimize for the power extracted from the wind. This power can be represented in non-dimensional form by the power coefficient, which is defined as the

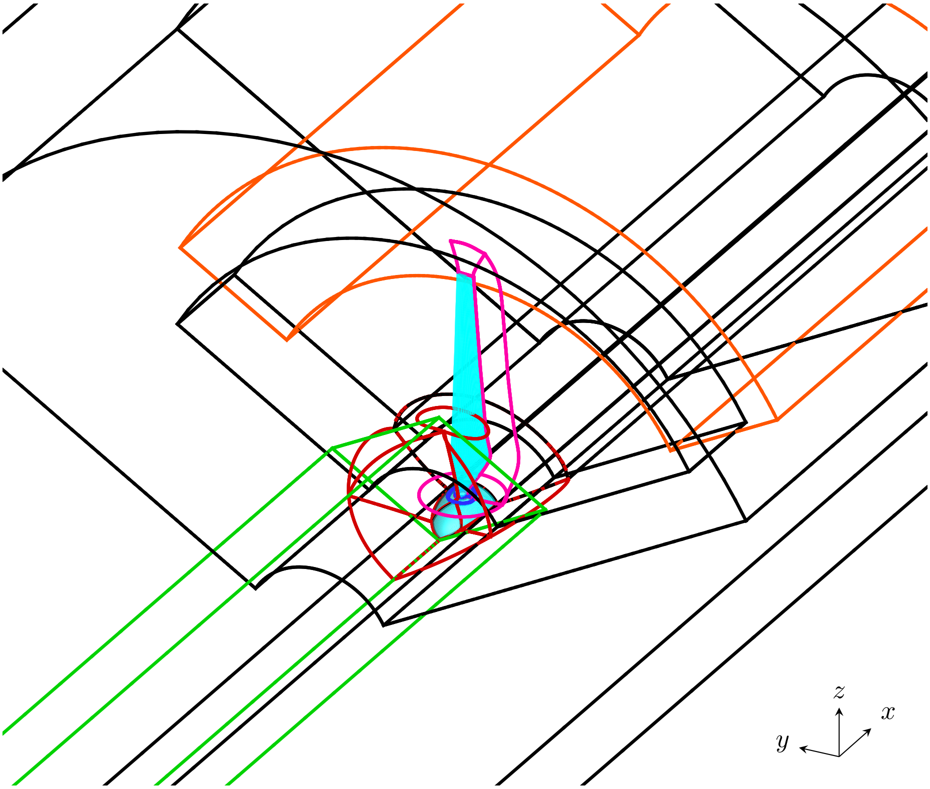


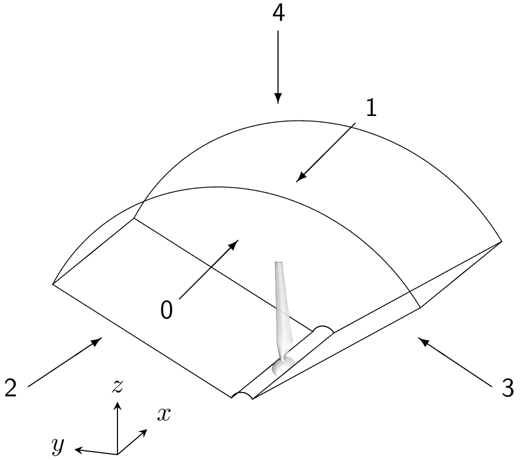
Figure 7.12: Detail of the block topology of the grid used for the discretization of the flow domain around the rotor blade. Surface of the blade and the spinner are depicted in cyan. Edges of: the body-fitted grid of the blade are in pink; the body-fitted grid of the spinner in red; rhombus-shaped background grid in green; collar grid in blue; cylindrical background grid for resolving helical vortex in orange; other cylindrical background grids in black.

ratio of the power extracted by the rotor and the power present in the wind, i.e.

$$C_{\text{power}} := \frac{\bar{P}}{\frac{1}{2}\pi\rho R^2 V_{\text{wind}}^3}, \tag{7.5}$$

where $\bar{P}$ is the power extracted by the rotor, V_{wind} the wind speed and R is the radius of the rotor. From an aerodynamic point of view, using the power coefficient as objective function might be a good option. However, using it without considering the implications of the resulting design on the costs for production and maintenance of the wind turbine most likely results in a design that is not economically feasible. Actually, the key factor in the design of a wind turbine is not the aerodynamic performance of the rotor, but the minimum cost of energy [66]. Therefore, this factor should be properly taken into account, either implicitly via the formulation of constraints or by a direct representation in the objective function itself.

To determine the cost of wind energy a number of aspects must be considered. These aspects include the costs for the construction of a wind turbine, but also the costs for



boundary	boundary condition
0	free-stream
1	subsonic outflow
2	rotational periodic
3	rotational periodic
4	subsonic outflow

Figure 7.13: The NREL Phase VI rotor blade together with an elliptical spinner and a schematic representation of the flow domain with the imposed boundary conditions indicated. Note, that the dimensions of the flow domain are not to scale and that rhombus-shaped part upstream of the spinner is not shown.

its operation and maintenance. Another prominent aspect to consider is the amount of energy that is produced. Because of the large number of factors involved in determining the actual cost of energy and its multidisciplinary character — which involves not only technical disciplines, but also, for instance, finance and politics — it is not easy to formulate an objective function that is suited for a gradient-based optimization method. It is, however, possible to discuss some of these factors and their implications for the cost of energy. For that purpose, a number of possible objective functions are discussed in the context of employing them for the optimization method that has been developed in the present research.

Blade mass

For the capital costs associated with a wind turbine, the rotor blades have an important contribution. A large part of the costs for a blade is proportional to its mass, since the mass is related to the amount of material used. The mass of the blade can be estimated, using for example the analytical model developed by Ashuri et al. [9]. Such a model has, however, not been implemented yet in the present optimization method. On the other hand, the blade mass is heavily correlated to the root bending moment [63] experienced by the rotor blade. An objective function based on the root bending moment might therefore be used instead. Since the root bending moment can be computed based on the aerodynamic forces on the blade, which follow from the flow solution, it is feasible to use the root bending moment in the present optimization method as objective function in a minimization problem.

Rotor thrust

Also the tower has a significant contribution in the total costs of a wind turbine. At rated conditions, the most prominent contribution of the loads exerted on the tower is associated with the thrust exerted by the rotor [166]. A lower rotor thrust therefore allows for a lighter tower construction, which in turn reduces the capital costs associated with the tower. Since the rotor thrust can be directly determined from the solution of the flow around the rotor, it can be used in the objective function in a minimization problem, in the present optimization framework. However, to prevent obtaining an optimal result that does not extract energy from the wind, a constraint should be imposed on the power coefficient.

Annual energy production

The annual energy production can be used, together with the expected lifetime of the wind turbine, to estimate the total amount of energy that will be produced by the wind turbine. To determine the annual energy production of a wind turbine, the performance of the rotor needs to be considered for the range of wind conditions the turbine will operate in. Therefore taking into account the probability density distribution for the wind speed at the site considered — represented by for instance a Weibull distribution [201] — and the capacity factor, an estimate can be made of the total amount of energy that can be produced in one year.

Considering this approach, it is observed that in order to determine the annual energy production, multiple flow solutions for different flow conditions are required. Moreover, each flow solution requires a solution of the adjoint equations for the power coefficient. The resulting adjoint vector can subsequently be used to efficiently determine the total derivative of the power coefficient with respect to the design variables, for the wind speed considered. When these total derivatives have been determined, the gradient of the objective function can be computed by means of either the dual number method or by using the product rule and chain rule of differentiation. Because the number of flow and adjoint solutions is proportional to the number of different wind conditions considered, using the annual energy production as objective function in a maximization problem will be very time consuming employing the flow model used in the present research. However, smart use of a large parallel computer can bring a solution, since the different flow solutions required can be carried out in parallel.

Power coefficient

One of the cost of energy related objective functions would be the most relevant choice to be used in the optimization problem. However, for testing the implementation of the present method, the economically less relevant power coefficient appears to be a better choice. For the power coefficient the optimal value for an unshrouded wind turbine is known — known as the Betz limit [17][18]. By using the power coefficient as objective function, the effectiveness of the optimization method can be assessed. Therefore, the power coefficient will be used as objective function.

---

[18] It appears that the limit has independently been derived by three different leading aerodynamic researchers. It has therefore been proposed to recognize their contribution as well and rename the limit as the Lanchester–Betz–Joukowsky limit [191].

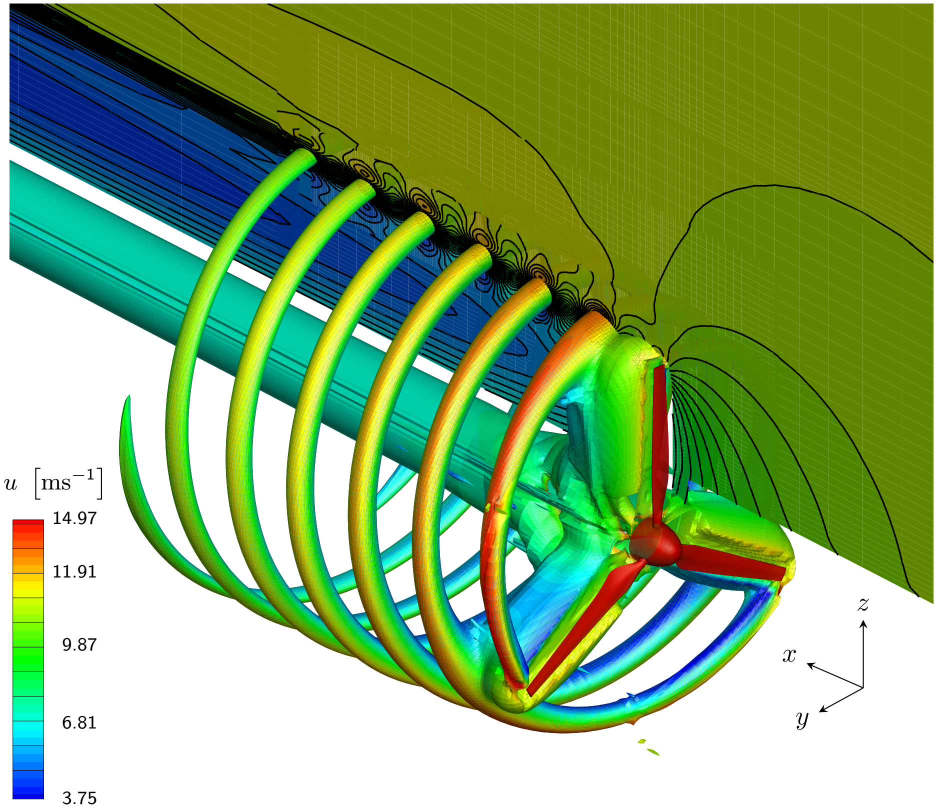


Figure 7.14: Iso-density surface for the flow around an isolated rotor rotating at 7.5 $\left[\mathrm{rads}^{-1}\right]$ for a wind speed of 10 $\left[\mathrm{ms}^{-1}\right]$. Iso-contours on a cross-section of the flow domain show the x-component of the velocity vector. The blades and spinner have been given a red colour.

Design variables and constraints

The design variables used in the present optimization are the x and y-components of the coordinates of the control points. The same regularity constraints have been imposed on the control points as in the wing optimization problems considered earlier. Moreover, no constraints have been imposed on, for instance, the root bending moment or the axial force exerted on the rotor. The pitch angle, which is also used as design variable, is limited to be in the range from -10° to 10°.

7.5.5 Approach and discussion

Figure 7.14 presents the solution for the flow around the rotor for the initial design, for the flow conditions specified in subsection 7.5.2 and the discretization that has been described in subsection 7.5.3. In this figure the helical vortex is visualized using an iso-density surface of 0.999886 of the free-stream density. Moreover, a cross-section of the flow domain is depicted, showing the iso-lines for the x-component of the velocity vector.

Considering the result, it is clear that the helical vortex generated by the rotor is preserved well over quite some distance downstream of the rotor. The helical vortex being no longer resolved at little over one rotor diameter from the rotor plane is due to the fine cylindrical background grid ending at that location, as can be observed in figure 7.11 on page 163. In the coarser cylindrical background grid used for the discretization further downstream, the vorticity is quickly dissipated.

The corresponding convergence history, is presented in figure 7.15. From this figure it is clear that a convergence of about 14 orders of magnitude is achieved. First, $40 \cdot 10^3$ Runge-Kutta iterations are performed to obtain the initial solution for Newton's method. To be able to obtain a solution for the system of linear equations — by means of the preconditioned GMRES method — involved in Newton's method, the application of damping was required. Damping has been applied by adding an additional term to the diagonal part of the Jacobian matrix. This additional term was chosen to be proportional to the inverse of the local time-step, based on a specified CFL number. To prevent divergence of the solution procedure, even intermediate altering of the CFL number was required. Eventually, after a considerable number of Newton iterations have been performed, convergence of the flow solution has been achieved, as can be observed in the convergence history. It is, however, not clear whether the resulting solution is truly steady, because the troublesome convergence might be caused by the flow solution exhibiting pseudo-unsteady behaviour. That a steady state solution is still found, can be caused by a number of factors. For instance due to the occurrence of artificial dissipation, caused by the discretization of the governing equations, that damps the unsteadiness. Another factor is the pseudo-time-integration by means of Newton's method, which can also have a strong damping effect. A likely source of the suggested unsteadiness is flow separation occurring at the tip of the rotor blade.

Subsequently using the flow solution for assembling the adjoint equations for the power coefficient and solving this system of linear equations did not lead to a result. Trying to use a so-called defect-correction approach [101, 173] for solving the adjoint equations did not give any solace either. An option might be to use the recursive projection method [28,126,165] instead. In this method, the dominant eigenvalues are identified and a separate solution method is applied for the corresponding eigenspace, resulting in a better overall convergence of the solution. However, in the present research, no attempt has been made in implementing such a method.

Moreover, if the troublesome convergence behaviour is really caused by the flow solution exhibiting unsteady flow effects, a solution of the adjoint equations would not provide accurate gradients. Krakos investigated the effect of small-scale output unsteadiness on the adjoint based sensitivity [103, 104]. Results of that investigation showed that even when a stationary-point solution is obtained, computing the sensitivity by means of the steady adjoint equations does not provide an accurate result, compared to the result of using the unsteady adjoint equations for a time-accurate flow solution. Therefore, it might be necessary to consider using an approach based on the unsteady adjoint equations.

Due to the challenges that emerged in attempting to solve the present optimization problem, unfortunately, no optimization results could be obtained for the rotor blade optimization problem presently considered. Apart from the suggestions already made, the introduction of viscous flow effects in the flow model might already prevent part of the problems from occurring, because these viscous flow effects could effectively damp the unsteady flow behaviour that is not damped in the inviscid flow model. Moreover, the

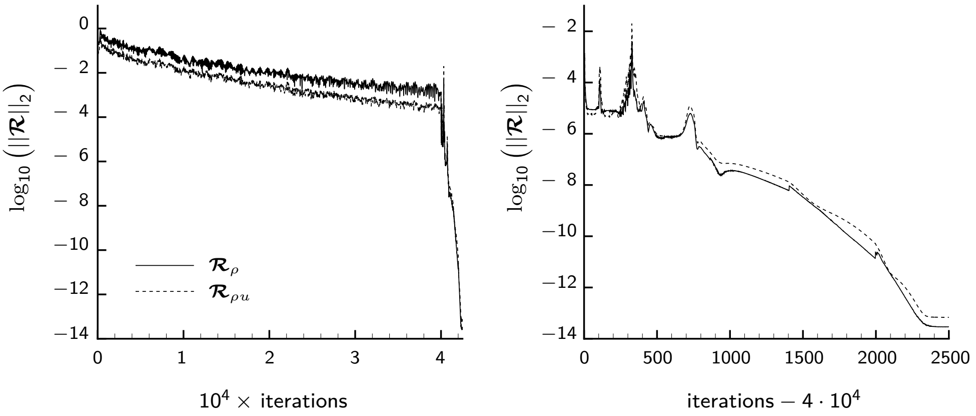


Figure 7.15: Convergence history for the flow around the NREL Phase VI rotor. Left plot shows full convergence history. Right plot shows the part of the convergence history corresponding to the use of Newton's method.

introduction of a viscous flow model can also serve the purpose of increasing the validity of the flow model. This increased accuracy with which flow physics is represented might, however, introduce a new source of unsteadiness by means of vortex shedding from the cylindrical part of the blade near the hub of the rotor. Furthermore, it should be noted, that either handling viscous flow effects — for instance by using a RANS model — or computing a time-accurate flow solution, combined with an unsteady adjoint equation method, both incur a significant increase in the computational requirements for the optimization method.

7.6 Concluding remarks

This chapter presented the optimization method that has been developed by coupling the different parts of the method realized in the present research to a gradient-based optimizer. Two different aerodynamic shape optimization problems have been solved and the results have been presented. These results show that the optimization method can be used successfully for solving aerodynamic shape optimization problems for different flow conditions and objective functions considered.

For the minimization of the drag for a swept wing in transonic flow, a reduction of 34.1% in drag coefficient was achieved for a fixed value of the lift coefficient. A breakdown of the time required for the different tasks in the optimization showed that the computing of the flow solution, in particular the explicit pseudo-time-integration, takes the largest portion of the computing time, while solving the adjoint equations only has a minor contribution to the overall time required for a single optimization iteration.

Solving the wing span efficiency maximization problem resulted in a design with a wing span efficiency significantly exceeding 1.0. The reason for such a large value requires further investigation. Moreover, in both optimizations, a number of geometrical constraints have been imposed to enforce a feasible result of the optimization problem. However, it was found that these constraints still allow geometries that can present challenges for the structural integrity of the design. It is therefore recommended to add additional geometrical constraints to the optimization to prevent this behaviour. An even better alternative would be to use structural constraints instead, which take the influence of the local thickness of the wing on its structural properties into account.

In attempting to solve an aerodynamic shape optimization problem involving a rotor blade, a number of challenges arose that are yet to be addressed. One of the challenges concerns the convergence behaviour for the flow solution. Application of damping and a considerable number of implicit time-iterations was required to obtain a converged result. Subsequently trying to solve the adjoint equations was not successful. A number of solutions have been proposed, to face the challenges encountered.

8

CONCLUDING REMARKS AND RECOMMENDATIONS

> *"In summary, the idea is to try to give all the information to help others to judge the value of your contribution; not just the information that leads to judgment in one particular direction or another."* [59]
>
> — RICHARD P. FEYNMAN (1918 – 1988)

THE design of a wind turbine comprises a multidisciplinary challenge, of which the aerodynamic design of the rotor blades forms an important aspect. The research presented in this thesis focussed on the development of a method for the aerodynamic design of a rotor blade for a horizontal axis wind turbine. For the choices made in this process, the multidisciplinary character of the task has been anticipated. The conclusions drawn from the present research are listed below. This chapter concludes with recommendations for directions of further research and further development of the optimization method.

8.1 Concluding remarks

Based on research presented in this thesis, the following conclusions can be drawn for the different aspects of the aerodynamic shape optimization method considered.

Shape parametrization

A NURBS surface is considered a good choice for the parametrization of the wind turbine blade geometry used in solving an aerodynamic shape optimization problem, because:

- it requires a limited number of design parameters to:
  - accurately represent a typical wind turbine rotor blade geometry;
  - cover the design space sufficiently in order to find an optimal solution for the aerodynamic shape optimization problem.
- it is compatible with computer aided design methods, which facilitates the incorporation of the aerodynamic shape optimization method in the multidisciplinary design process of a wind turbine rotor blade;
- it can conveniently be used for the surface grid generation method as well.

Moreover, it has been found that:

- for a typical wind turbine rotor blade geometry 13 × 13 control points is adequate for achieving a sufficiently accurate fit;
- a smaller number of control points — than required for an accurate fit — is still sufficient for the optimization method to reach an optimum, for the 2D model optimization problem considered.

Flow domain discretization

For the discretization of the flow domain, a hyperbolic grid generation method has been implemented, which was found to:

- be an efficient means for providing a high quality discretization of the flow domain;
- be robust enough to be used for solving aerodynamic shape optimization problems;
- provide good grid quality throughout the optimization procedure.

Moreover, it has been found that surface grid generation:

- by means of linear transfinite interpolation provides a good grid quality in most situations;
- by solving equations for elliptical grid generation can be used when linear transfinite interpolation does not provide satisfactory grid quality.

Overset block connectivity

Methods have been implemented that can handle composite overset grids. The use of composite overset grids allows for:

- using a complex topology of the flow domain, without further complicating the grid generation;
- locally increasing grid resolution by introduction of additional blocks.

Moreover, a method for the generation of so-called zipper grids has been implemented to realize a closed non-overlapping surface grid, which can subsequently be used for

- the accurate evaluation of surface integrals;
- elimination of control volumes that are outside the physical flow domain, by means of a ray-casting method.

Flow model and solution method

The flow model used is the one based on the Euler equations. The partial differential equations have been discretized by means of a cell-centred finite volume method with a second-order spatial accuracy of the flux discretization.

- implicit time integration by means of Newton's method provides an efficient means for computing a steady-state solution;

    - construction of the ILU(k) preconditioning matrix based on the first-order Jacobian matrix results in better convergence properties of the Krylov subspace method — used for computing the update of the flow solution;

- verification of the implementation of the method by considering the steady 2D flow around a NACA 0012 aerofoil, showed that:
    - the observed spatial order of convergence corresponds with the expected order, although different parameters can have a strong effect on this result, such as the method used for:
        - the extrapolation of the pressure to halo control volumes representing a solid wall boundary condition;
        - the discretization of the convective flux discretization;

      and
        - the far-field of the flow domain.
    - the order of convergence observed for a composite overset grid discretization is consistent with the observed order of convergence using a single block discretization.
- validation of the method has been performed by considering the flow around an ONERA M6 wing at transonic flow conditions, for which there are well documented experimental results available. The good agreement of the computational results with the results from these experiments, showed that the method can be used to accurately predict high Reynolds number attached flows.

Sensitivity analysis

The adjoint equations have been used to efficiently compute the derivatives of the objective function with respect to a large number of design variables. A discrete adjoint equation method has been implemented for this purpose. With respect to this method, the following conclusions can be drawn:

- the use of template functions in conjunction with the dual number method:
    - allows for reusing already implemented functions, which saves development time;
    - makes the method used for computing derivatives fully consistent with the underlying function, which renders derivatives accurate to machine accuracy.
- derivatives computed using the discrete adjoint equation method have been shown to be consistent with derivatives computed using an equally accurate but less efficient method.

Optimization method

An optimization framework has been developed by coupling the different components implemented in the present research to a gradient-based optimizer. Subsequently, different optimization problems have been considered to demonstrate the capabilities of this method. Solving the test problems showed that:

- the optimization method can be used successfully for solving aerodynamic shape optimization problems for a range of flow conditions and objective functions;
- computing the flow solution is the most time consuming part of performing an optimization iteration, in particular the explicit pseudo-time-integration part;
- solving the adjoint equations has only a minor contribution to the overall time required for an optimization iteration, which shows that using the adjoint equations is a good choice for efficiently computing gradients, specifically for the case of a large number of design variables;
- in the definition of a constrained aerodynamic shape optimization problem, a thorough consideration of the constraints to be used and their corresponding formulation is important.

An optimization problem involving a rotor blade has been considered. In the process of performing an optimization iteration, the following has been noted:

- a fully converged flow solution has been obtained, however:
    - application of damping was required to be able to obtain a converged solution using the iterative solution method for solving the system of linear equations that is part of Newton's method;
    - a solution to the adjoint equations could not be obtained.
- the suggestion has been made that unsteady flow behaviour might be the cause for the troublesome convergence of the flow solution.

8.2 Recommendations

It is demonstrated that the optimization method developed in the present research is able to successfully solve aerodynamic shape optimization problems. There is however still room for improvement of the method. Moreover, the insights gained during its development have also given rise to new ideas for extending the optimization method. Therefore, this section is dedicated to recommend extensions and further improvement of the method.

Parametrization

In the present research a NURBS surface has been used for the direct parametrization of the aerodynamic shape. This approach helps to incorporate the aerodynamic shape optimization method in the multidisciplinary design of a wind turbine rotor blade. Moreover, the parametrization by means of a NURBS surface facilitates the surface grid generation. The aerodynamic shape optimization problems solved show that the current parametrization method is adequate for this purpose. However, other researches indicate that a different approach of the parametrization method can result in a better convergence of the optimization procedure [114]. Therefore, it is recommended to investigate whether or not this claim also holds for the optimization method used in the present research. This goal can be accomplished by considering the effect of using such an alternative parametrization method; for instance, using the decomposed approach, as discussed in section 2.1.3. This investigation can be performed by simply trying such an alternative parametrization method and comparing the performance of the optimization procedure with the performance of using the present parametrization method.

However, an alternative approach, is to first perform a so-called epistasis analysis [135], to identify the interaction between design parameters and their mutual influence on the objective function. The results of such an analysis can give insight in the suitability of a certain set of design parameters to be used for the optimization.

Flow domain discretization

In the present implementation, a single processor core is used for the construction of a single block of the grid. This strategy does in general not provide an optimal distribution of the computational effort required for the generation of the grid. Moreover, the difference in effort required for computing a body-fitted grid, compared to creating a background grid, is not taken into account. It is therefore recommended, to take the required effort into account, when the distribution of the blocks over the available processor cores is performed. Moreover, multiple cores should be used — if available — for the generation of the body-fitted grids.

As an alternative for using structured hyperbolic field grids in conjunction with an overset block connectivity method, similar geometrical flexibility could be achieved using unstructured grids. Unstructured grids have the additional advantage that less additional work is involved when a different topology is considered, compared to using composite overset grids. Although the advantages of using structured grids, mentioned in chapter 1, are still valid.

Flow model and solution method

The results of the inviscid flow model, used in the present research, showed good agreement with experimental results. However, it has also been observed, that some discrepancies between results from the experiment and the numerical results can be attributed to leaving out viscous flow effects. Therefore, to further increase the accuracy of the flow model, it is recommended to include these effects of viscosity. Moreover, since high Reynolds number flows are considered, it also means that the modelling of turbulence must be taken into account. For this purpose, the use of a Reynolds-averaged Navier-Stokes eddy viscosity model [142, 203] is recommended.

For the numerical simulation of the flow about a horizontal axis wind turbine, it is important to accurately capture the helical vortex, formed at the blade tips, because of its inductive effect on the flow in the rotor plane. For the standard second-order spatial accuracy, this vortical motion quickly dissipates. To preserve the helical vortex further downstream of the rotor, without using excessive grid refinement, it is recommended to use reconstruction methods that can realize a higher than second-order spatial accuracy. The use of a so-called WENO scheme [110] might be a good option for this purpose, as well as a high-order finite-difference scheme [30] or a discontinuous Galerkin finite element method [13, 134].

As obtaining an initial guess for Newton's method by means of explicit Runge-Kutta pseudo-time-integration proved to consume the largest portion of the time required to compute the flow solution, considering the use of a different globalization strategy is recommended. Based on the investigation presented by Pawlowski et al. [137] an inexact Newton backtracking method [56] might be a promising option.

A compressible flow model has been used to represent the flow. For a full-scale wind turbine, the flow velocities in the tip region of the blade are such that compressible flow effects are not negligible. However, there are also regions in the flow domain for which compressible flow effects are negligible. These regions can cause convergence problems for the solution method, when a compressible flow model is used. This effect might be another reason for the troublesome convergence for the wind turbine case considered. A possible remedy is using low Mach number preconditioning [190], that might increase the convergence speed.

Overset block connectivity

When viscous flow effects are included in the flow model, accurately resolving the flow in the boundary layer regions becomes important. Having block overlap in the boundary layer region near curved surfaces can result in using incorrect donor information for the interpolation of the flow solution. This issue can be dealt with by application of a curvature correction when determining the interpolation coefficients. Using the curvature correction method presented by Schwarz [160, 161] would be a good choice.

If higher than second-order spatial accuracy is considered, the use of linear interpolation to obtain the solution for fringe control volumes is no longer sufficient to maintain the higher-order spatial accuracy throughout the domain. The use of a higher-order interpolation would be required. Based on the work of Lee on the simulation of rotorcraft aerodynamics [107], the use of Hermitian cubic interpolation is recommended.

Sensitivity analysis

The method used for the calculation of the sensitivities with respect to the grid by means of the construction of a grid with dual number arithmetic, has a computational effort proportional to the number of design variables. For a large number of design variables, the computation of the grid sensitivities provides a considerable contribution to the total time required for performing a single optimization iteration. It is therefore recommended to use a method that is independent of the number of design variables. As suggested by Carpentieri [31] and Dwight [53] and demonstrated, among others, by Hicken [82], an adjoint equation method can be used for this purpose as well.

To limit the memory requirements for solving the adjoint equations, it might be an option to apply a Jacobian-free adjoint approach. Especially, when higher-order methods would be used, the Jacobian matrix becomes less sparse, increasing the memory requirements of the method. In the present approach, the computation of the Jacobian matrix is still required for the construction of the preconditioning matrix. This preconditioning matrix is, however, generally based on a low-order discretization, which is more sparse. To apply a Jacobian-free adjoint approach, products $\underline{\underline{A}}^T \boldsymbol{q}$ must be computed for the Krylov-subspace method. Evaluation of these products requires the reverse mode of algorithmic differentiation, applied to the function that computes the residual of the flow equations. Although dual numbers cannot be used for this purpose, it still is possible to exploit the template implementation of that function by using a different data type, which can provide the reverse mode of algorithmic differentiation. An example of such a data type has been presented by Hogan [85].

To alleviate the restriction of requiring a steady flow solution to compute the gradient, it might be worthwhile to pursue an unsteady adjoint approach. Such an approach does incur a considerable increase in computational requirements, for solving an aerodynamic shape optimization problem, because of the time-accurate flow solution required. After a time-accurate solution is obtained, subsequently, the unsteady adjoint equation must be solved for every time-step, in reverse order.

Computational resources

The use of the adjoint equation method provides an efficient means for computing the gradients, leaving computing the flow solution as the most time consuming part of the gradient-based optimization procedure. The time required for obtaining the flow solution can be reduced, for instance by using convergence acceleration techniques, like grid sequencing or a multigrid [91] method. However, an alternative approach is to use different hardware than CPUs for computing the flow solution.

One option is to use a graphical processing unit (GPU) for this purpose. Because of their original application, GPUs are designed to handle floating-point data in a massively parallel fashion. This capability makes them well suited for performing general purpose computations involving floating-point data as well — for instance, computing flow solutions. Examples of using GPUs for obtaining flow solutions [34, 99] show that a significant speed-up can be achieved. However, in order to use a GPU for obtaining the flow solution, large parts of the implementation of the flow solution method must be rewritten, to take advantage of the different hardware.

A different alternative is to use reconfigurable hardware to further reduce the computational time required for obtaining the flow solution. With reconfigurable hardware, by means of a so-called field programmable gate array (FPGA), the hardware can be configured such that it is specialized for performing a specific task. This specialization has the advantage that the task at hand can be performed faster than when a combination of general operations is performed to realize the same task. The latter approach is customary for non-reconfigurable hardware, like a CPU or a GPU. Although, the use of reconfigurable hardware for performing CFD calculations is not very common yet, a significant speed-up is expected from this approach, some estimates state a speed-up of up to two orders of magnitude [8]. One of the disadvantages of using reconfigurable hardware is that the configurations must be specified using a so-called hardware description language. Such a language is inherently different from the programming languages CFD methods are commonly implemented in. However, a recent development among major FPGA vendors is the embracement of high level languages, such as OpenCL [6], which can significantly reduce the effort required for the implementation of a CFD method for reconfigurable hardware [148]. An additional advantage of using reconfigurable hardware is the significantly lower power consumption compared to power consumption by CPUs and GPUs [6, 8].

Optimization method

It is common knowledge that a gradient-based optimization method in general returns a local optimum as the result of an optimization problem. However, as pointed out, for the number of design variables considered and costs for a single evaluation of the objective function, a gradient-based method is the only feasible option. Nevertheless, a zeroth-order method and a low-fidelity flow model could still be used to obtain an initial guess in the neighbourhood of the global optimum. Subsequently, the gradient-based method developed in the present research can be used to quickly converge to the actual global optimum. Note, that this approach requires that the design space represented by the low-fidelity flow model sufficiently accurately resembles the design space represented by the high-fidelity flow model. For the purpose of increasing the accuracy of the result from the low-fidelity model, a response surface method can be employed, that represents the difference between the results of the high and low-fidelity model [42]. Moreover, the zeroth-order method must be able to find the neighbourhood of the global optimum, which can still be quite challenging if a considerable number of design variables is considered.

Apart from including more of the physics in the model for the flow, it is also worthwhile to consider other physical aspects in the optimization method as well. For instance the weight of the rotor blade or the elastic deformation of the rotor blade due to the aerodynamic loads.

Moreover, since a wind turbine generally does not operate at only a single wind condition, it is recommended to perform multipoint optimizations, such that the performance at off-design conditions is still reasonable. Another subject to consider is the discrepancy that inherently exists between the numerical representation of a geometry and its actual physical realization. These discrepancies can for instance be caused by manufacturing tolerances [77] or approximations in the numerical model. The existence of these discrepancies can be taken into account by applying an optimization under uncertainty [87, 149] approach. By using such an approach, the result of solving an optimization problem is more likely to show optimal performance in reality as well.

BIBLIOGRAPHY

THE page numbers between the square brackets at the end of each entry indicate the pages where reference occurs.

[1] ABBOTT, I. AND VON DOENHOFF, A. (1959) *Theory of Wing Sections*. Dover Books on Aeronautical Engineering Series, Dover Publications. [Page 203].

[2] ABRAMOWITZ, M. AND STEGUN, I. A. (1965) *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications Inc., New York. [Page 152].

[3] AFTOSMIS, M. J. (1997) *Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries*. Von Karman Institute for Fluid Dynamics, Rhode-Saint-Genése, Belgium, lecture Series 1997-02. [Page 66].

[4] AL-MOHY, A. H. AND HIGHAM, N. J. (2010) The complex step approximation to the Fréchet derivative of a matrix function. *Numerical Algorithms*, 53, pp. 133–148. [Page 95].

[5] VAN ALBADA, G. D., VAN LEER, B. AND ROBERTS JR, W. W. (1982) A comparative study of computational methods in cosmic gas dynamics. *Astronomy and Astrophysics*, 108, pp. 76–84. [Pages 27 and 84].

[6] ALTERA CORPORATION (2013), Implementing FPGA Design with the OpenCL Standard. [Page 177].

[7] ANDERSON, J. D. (2001) *Fundamentals of Aerodynamics*. McGraw-Hill, New York, 3 edn. [Page 156].

[8] ANDRÉS, E., CARRERAS, C., CAFFARENA, G., DEL CARMEN MOLINA, M., NIETO-TALADRIZ, O. AND PALACIOS, F. (2008) A Methodology for CFD Acceleration Through Reconfigurable Hardware. *46th AIAA Aerospace Science Meeting and Exhibit*, Reno, NV, no. AIAA 2008–481. [Page 177].

[9] ASHURI, T., VAN BUSSEL, G. J. W., ZAAYER, M. B. AND VAN KUIK, G. A. M. (2010) An Analytical Model to Extract Wind Turbine Blade Structural Properties for Optimization and Up-scaling Studies. *Torque 2010*. [Page 165].

[10] ATTA, E. H. AND VADYAK, J. (1981) Component-adaptive grid interfacing. *AIAA 19th Aerospace Science Meeting*, St. Louis, MO, no. AIAA 81–0382. [Page 53].

[11] ATTA, E. H. AND VADYAK, J. (1982) A grid interfacing zonal algorithm for three-dimensional transonic flows about aircraft configurations. *AIAA/ASME 3rd Joint Thermophysics Fluids, Plasma and Heat Transfer Conference*, St. Louis, MO, no. AIAA 82–1017. [Page 53].

[12] BALAY, S. ET AL. (2012) *PETSc Users Manual*. Tech. Rep. ANL-95/11 - Revision 3.3, Argonne National Laboratory. [Page 92].

[13] BASSI, F. AND REBAY, S. (1997) A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations. *Journal of Computational Physics*, 131, pp. 267 – 279. [Page 175].

[14] BAUMSLAG, B. (2000) *Fundamentals of Teaching Mathematics at University Level*. Imperial College Press. [Page 121].

[15] BECKER, G., SCHÄFER, M. AND JAMESON, A. (2011) An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations. *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, no. AIAA 2011–891. [Page 30].

[16] BENEK, J. A., STEGER, J. L. AND DOUGHERTY, F. C. (1983) A Flexible Grid Embedding Technique with Application to the Euler Equations. *AIAA 6th Computational Fluid Dynamics Conference*, Danvers, MA, no. AIAA 83–1944. [Page 53].

[17] BETZ, A. (1920) Das maximum der theoretisch möglichen ausnützung des windes durch windmotoren. *Zeitschrift für das gesamte Turbinenwesen*, 26, pp. 307–309. [Page 166].

[18] BIEDRON, R. T. AND THOMAS, J. L. (2009) Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications. *47th AIAA Aerospace Sciences Meeting & Exhibit*, Orlando, FL, no. AIAA 2009–1360. [Page 54].

[19] BIEDRON, R. T., VATSA, V. N. AND ATKINS, H. L. (2005) Simulation of Unsteady Flows Using an Unstructured Navier-Stokes Solver on Moving and Stationary Grids. *17th AIAA Computational Fluid Dynamics Conference*, Toronoto, ON, no. AIAA 2005–5093. [Page 54].

[20] BLAZEK, J. (2005) *Computational Fluid Dynamics: Principles and Applications*. Elsevier, Oxford. [Pages 35, 81 and 87].

[21] BONDY, J. A. AND MURTY, U. S. R. (1976) *Graph Theory With Applications*. Elsevier Science Publishing, New York, U.S.A. [Page 128].

[22] BONET (1991) An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31, pp. 1–17. [Page 55].

[23] BRACEWELL, R. N. (1999) *The Fourier transform and its applications*. McGraw-Hill, New York, 3rd edn. [Page 22].

[24] BROYDEN, C. G. (1969) A new double-rank minimization algorithm. *Notices American Mathematical Society*, 16, pp. p. 670. [Page 149].

[25] BROYDEN, C. G. (1970) The Convergence of Single-Rank Quasi-Newton Methods. *Mathematics of Computation*, 24, pp. pp. 365–382. [Page 149].

[26] BUCKLEY, H. P., ZHOU, B. Y. AND ZINGG, D. W. (2009) Airfoil Optimization Using Practical Aerodynamic Design Requirements. *27th AIAA Applied Aerodynamics Conference*, San Antonio, TX, no. AIAA 2009–3516. [Page 19].

[27] CAMBIER, L., GAZAIX, M., S. HEIB, S., PLOT, M. P., VEUILLOT, J.-P., BOUSSUGE, J.-F. AND MONTAGNAC, M. (2011) CFD Platforms and Coupling: An Overview of the Multi-Purpose elsA Flow Solver. *Aerospace Lab*, Issue 2. [Page 54].

[28] CAMPOBASSO, M. S. (2004) *Effects of Flow Instabilities on the Linear Harmonic Analysis of Unsteady Flow in Turbomachinery*. PhD thesis, University of Oxford. [Page 168].

[29] CAREY, G. F. (1997) *Computational Grids: Generation, Adaptation and Solution Strategies*. Series in Computational and Physical Processes in Mechanics, Taylor & Francis. [Pages 4, 33 and 98].

[30] CARPENTER, M. H., GOTTLIEB, D. AND ABARBANEL, S. (1994) Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111, pp. 220–236. [Page 175].

[31] CARPENTIERI, G. (2009) *An Adjoint-Based Shape-Optimization Method for Aerodynamic Design*. PhD thesis, Technische Universiteit Delft. [Pages 7, 15, 18, 19, 114, 152 and 176].

[32] CASELLA, G. AND BERGER, R. (2002) *Statistical inference*. Duxbury advanced series in statistics and decision sciences, Thomson Learning. [Page 214].

[33] CASTONGUAY, P. AND NADARAJAH, S. (2007) Effect of Shape Parameterization on Aerodynamic Shape Optimization. *45th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, no. AIAA 2007–0059. [Page 20].

[34] CASTONGUAY, P., WILLIAMS, D. M., VINCENT, P. E., LOPEZ, M. AND JAMESON, A. (2011) On the Development of a High-Order, Multi-GPU Enabled, Compressible Viscous Flow Solver for Mixed Unstructured Grids. *20th AIAA Computational Fluid Dynamics Conference*, Honolulu, HI, no. AIAA 2011–3229. [Page 177].

[35] CHAN, M. K. (2003) *Supersonic Aircraft Optimization for Minimizing Drag and Sonic Boom*. PhD thesis, Stanford University. [Page 6].

[36] CHAN, W. M. (1999) Hyperbolic Methods for Surface and Field Grid Generation. *Handbook of Grid Generation [185]*, chap. 5, CRC Press. [Pages 16, 36, 45 and 46].

[37] CHAN, W. M. (2009) Enhancements to the Hybrid Mesh Approach to Surface Loads Integration on Overset Structured Grids. *19th AIAA Computational Fluid Dynamics Conference*, San Antonio, TX, no. AIAA 2009–3990. [Pages 16 and 58].

[38] CHAN, W. M. (2009) Overset Grid Technology Development at NASA Ames Research Center. *Computers & Fluids*, 39, pp. 496–503. [Page 53].

[39] CHAN, W. M. AND STEGER, J. L. (1992) Enhancement of a Three-dimensional hyperbolic Grid Generation Scheme. *Applied Mathematics and Computation*, 51, pp. 181–205. [Pages 16, 45, 47 and 199].

[40] CHISHOLM, T. T. (2006) *A Fully Coupled Newton-Krylov Solver with a One-Equation Turbulence Model*. PhD thesis, University of Toronoto. [Page 93].

[41] CHIU, I. T. AND MEAKIN, R. L. (1995) On automating domain connectivity for overset grids. *33rd Aerospace Sciences Meeting & Exhibit*, Reno, NV, no. AIAA 95–0854. [Page 65].

[42] CHOI, S., ALONSO, J. J., KROO, I. M. AND WINTZER, M. (2008) Multifidelity Design Optimization of Low-Boom Supersonic Jets. *Journal of Aircraft*, 45, pp. 106–118. [Page 178].

[43] CHUNG, H.-S., CHOI, S. AND ALONSO, J. J. (2003) Supersonic Business Jet Design using a Knowledge-Based Genetic Algorithm with an Adaptive, Unstructured Grid Methodology. *AIAA 21st Applied Aerodynamics Conference*, Orlando, FL, no. AIAA 2003–3791. [Page 6].

[44] CNOSSEN, J. M. (2007) *Goal-oriented modelling-error estimation for hierarchical models of a different type*. PhD thesis, Technische Universiteit Delft. [Page 126].

[45] CORLISS, G. AND GRIEWANK, A. (1993) Operator Overloading as an Enabling Technology for Automatic Differentiation. *Proceedings of the First Annual Object-Oriented Numerics Conference*, Sunriver, OR. [Page 124].

[46] COURANT, R., FRIEDRICHS, K. AND LEWY, H. (1928) Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100, pp. 32–74. [Page 90].

[47] DAVIS, B. (2008) *IEEE Standard for Floating-Point Arithmetic*. Standard 754-2008, Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY 10016-5997, USA. [Page 9].

[48] DAVOUDZADEH, F., MCDONALD, H. AND THOMPSON, B. (1995) Accuracy evaluation of unsteady CFD numerical schemes by vortex preservation. *Computers & Fluids*, 24, pp. 883–895. [Page 213].

[49] DENNIS, J. E. AND MORÉ, J. J. (1977) Quasi-Newton Methods, Motivation and Theory. *SIAM Review*, 19, pp. 46–89. [Page 149].

[50] DERKSEN, R. W. AND ROGALSKY, T. (2010) Bezier-PARSEC: An optimized aerofoil parametrization for design. *Advances in Engineering Software*, 41, pp. 923–930. [Page 19].

[51] DESTARAC, D. (2011) Investigating Negative Drag in Grid Convergence for Two-Dimensional Euler Solutions. *Journal of Aircraft*, 48, pp. 1468–1470. [Pages 104 and 105].

[52] DURAND, W. F. (1935) *Aerodynamic Theory*. Springer, Berlin, Germany. [Page 184].

[53] DWIGHT, R. P. (2006) *Efficiency improvements of RANS-based analysis and optimization using implicit and adjoint methods on unstructured grids*. PhD thesis, University of Manchester. [Pages 3, 15 and 176].

[54] EASTHAM, M. S. P. (1961) On the Definition of Dual Numbers. *The Mathematical Gazette*, 45, pp. 232–233. [Page 10].

[55] EINSTEIN, A. (1905) Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik*, 323, pp. 639–641. [Page 76].

[56] EISENSTAT, S. C. AND WALKER, H. F. (1994) Choosing the Forcing Terms in an Inexact Newton Method. *SIAM Journal on Scientific Computing*, 17, pp. 16–32. [Pages 95 and 176].

[57] FENG, Y. T. AND OWEN, D. R. J. (2002) An augmented spatial digital tree algorithm for contact detection in compuational mechanics. *International Journal for Numerical Methods in Engineering*, 55, pp. 159–176. [Page 55].

[58] FERDMAN, M., WENISCH, T. F., AILAMAKI, A., FALSAFI, B. AND MOSHOVOS, A. (2008) Temporal instruction fetch streaming. *41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–10. [Page 123].

[59] FEYNMAN, R. P. AND ROBBINS, J. (1999) *The Pleasure of Finding Things Out: The Best Short Works of Richard P.Feynman*. Art of Mentoring Series, Perseus Books. [Page 171].

[60] FIKE, J. A. (2012) *Multi-Objective Optimization Using Hyper-Dual Numbers*. PhD thesis, Stanford University. [Pages 10, 92 and 140].

[61] FIKE, J. A. AND ALONSO, J. J. (2011) The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations. *49th AIAA Aerospace Sciences Meeting*, Orlando, FL, no. AIAA 2011–886. [Pages 10 and 92].

[62] FIKE, J. A., JONGSMA, S. H., ALONSO, J. J. AND VAN DER WEIDE, E. T. A. (2011) Optimization with Gradient and Hessian Information Calculated Using Hyper-Dual Numbers. *AIAA 29th Applied Aerodynamics Conference*, Honolulu, HI, no. AIAA 2011–3807. [Page 140].

[63] FISCHER, G. R., KIPOUROS, T. AND SAVILL, A. M. (2014) Multi-objective optimisation of horizontal axis wind turbine structure and energy production using aerofoil and blade properties as design variables. *Renewable Energy*, 62, pp. 506 – 515. [Page 165].

[64] FLETCHER, R. (1970) A New Approach to Variable Metric Algorithms. *Compututer Journal*, 13, pp. 317–322. [Page 149].

[65] FUDGE, D., ZINGG, D. AND HAIMES, R. (2005) A CAD-Free and a CAD-Based Geometry Control System for Aerodynamic Shape Optimization. *43rd AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, no. AIAA 2005–0451. [Page 17].

[66] FUGLSANG, P. AND MADSEN, H. (1999) Optimization method for wind turbine rotors. *Journal of Wind Engineering and Industrial Aerodynamics*, 80, pp. 191 – 206. [Page 164].

[67] GAUGER, N., WALTHER, A., MOLDENHAUER, C. AND WIDHALM, M. (2008) Automatic differentiation of an entire design chain for aerodynamic shape optimization. Tropea, C., Jakirlic, S., Heinemann, H.-J., Henke, R. and Hönlinger, H. (eds.), *New Results in Numerical and Experimental Fluid Mechanics VI*, vol. 96 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pp. 454–461, Springer Berlin / Heidelberg. [Page 19].

[68] GIGUÈRE, P. AND SELIG, M. S. (1999) *Design of a Tapered and Twisted Blade for the NREL Combined Experiment Rotor*. Technical Report NREL/SR-500-26173, National Renewable Energy Laboratory. [Page 160].

[69] GILES, M. B. AND PIERCE, N. A. (2000) An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, 65, pp. 393–415. [Page 126].

[70] GILL, P. E., MURRAY, W. AND SAUNDERS, M. A. (1986) *Some Theoretical Properties of an Augmented Lagrangian Merit Function*. Technical Report SOL 86-6R, Stanford University. [Page 149].

[71] GILL, P. E., MURRAY, W. AND SAUNDERS, M. A. (2005) SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47, pp. 99–131. [Pages 18, 26 and 149].

[72] GLAUERT, H. (1935) Airplane propellers. *Aerodynamic Theory [52]*, chap. L, Springer. [Page 2].

[73] GOLDFARB, D. (1970) A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation*, 24, pp. 23–26. [Page 149].

[74] GOLDFARB, D. (1984) Optimal Estimation of Jacobian and Hessian Matrices That Arise in Finite Difference Calculations. *Mathematics of Computation*, 43, pp. 69–88. [Page 129].

[75] GRIEWANK, A. (2000) *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania. [Pages 10 and 123].

[76] GRIMALDI, R. P. (2003) *Discrete and Combinatorial Mathematics: An Applied Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. [Page 55].

[77] GUMBERT, C. R., NEWMAN, P. A. AND HOU, G. J.-W. (2002) Effect of Random Geometric Uncertainty on the Computational Design of a 3-D Flexible Wing. *20th AIAA Applied Aerodynamics Conference*, St. Louis, MO, no. AIAA 2002–2806. [Page 178].

[78] HARTEN, A. (1997) High Resolution Schemes for Hyperbolic Conservation Laws. *Journal of Computational Physics*, 135, pp. 260–278. [Page 82].

[79] HARTKAMP, R. M. (2009) *Optimization of Airfoils in Inviscid Transonic Flow using Discrete Adjoints*. Master's thesis, Universiteit Twente. [Pages 26 and 27].

[80] HASCOËT, L. (2007) *Automatic Differentiation by Program Transformation*. Technical report, INRIA Sophia-Antipolis. [Page 11].

[81] HENSHAW, W. D. (1996) Automatic Grid Generation. *Acta Numerica*, 5, pp. 121–148. [Pages 4, 34 and 35].

[82] HICKEN, J. E. (2009) *Efficient algorithms for future aircraft design: contributions to aerodynamic shape optimization*. PhD thesis, University of Toronoto. [Pages 15, 51, 95, 114 and 176].

[83] HICKEN, J. E. AND ZINGG, D. W. (2010) Induced-drag minimization of nonplanar geometries based on the Euler equations. *AIAA Journal*, 48, pp. 2564–2575. [Page 159].

[84] HICKS, R. M. AND HENNE, P. A. (1978) Wing Design by Numerical Optimization. *Journal of Aircraft*, 15, pp. 407–412. [Page 19].

[85] HOGAN, R. J. (2014) Fast reverse-mode automatic differentiation using expression templates in C++, *ACM Transactions on Mathematical Software*, in press. [Page 177].

[86] HUA, Y., SHEN, W. Z., SØRENSEN, J. N. AND ZHU, W. J. (2010) Determination fo the Angle of Attack on the Mexico Rotor using Experimental Data. *Torque 2010*. [Page 2].

[87] HUYSE, L., PADULA, S. L., LEWIS, M. AND LI, W. (2002) Probabilistic Approach to Free-Form Airfoil Shape Optimization Under Uncertainty. *AIAA Journal*, 40, pp. 1764–1772. [Page 178].

[88] INTEL CORPORATION (2013) *Intel® 64 and IA-32 Architectures Optimization Reference Manual*. No. 248966-028. [Page 215].

[89] JAMESON, A. (1985) Transonic flow calculations. Brezzi, F. (ed.), *Numerical Methods in Fluid Dynamics*, vol. 1127 of *Lecture Notes in Mathematics*, pp. 156–242, Springer-Verlag, Wien. [Page 90].

[90] JAMESON, A. (1988) Aerodynamic design via control theory. *Journal of Scientific Computing*, 3, pp. 233–260. [Pages 14, 19 and 26].

[91] JAMESON, A. AND BAKER, T. (1984) Multigrid Solution of the Euler Equations for Aircraft Configurations. *AIAA 22nd Aerospace Sciences Meeting*, no. AIAA 84–0093. [Page 177].

[92] JAMESON, A., SCHMIDT, W. AND TURKEL, E. (1981) Numerical solutions of the Euler equation by finite volume methods using Runge-Kutta time stepping schemes. *AIAA 14th Fluid and Plasma Dynamic Conference*, Palo Alto, CA, no. AIAA 1981–1259. [Pages 3 and 80].

[93] JAMESON, A. AND VASSBERG, J. C. (2000) Studies of Alternative Numerical Optimization Methods Applied to the Brachistochrone Problem. *CFD Journal*, pp. 281–296. [Pages 7 and 18].

[94] JAMESON, A. AND VASSBERG, J. C. (2001) Computation Fluid Dynamics for Aerodynamic Design: Its Current and Future Impact. *39th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, no. AIAA 2001–0538. [Page 6].

[95] KELLENERS, P. (2007) *An Edge-based Finite Volume Method for Inviscid Compressible Flow with Condensation*. PhD thesis, Universiteit Twente. [Pages 82, 83, 86 and 105].

[96] KHAMAYSEH, A. AND KUPRAT, A. (1999) Surface Grid Generation Systems. *Handbook of Grid Generation [185]*, chap. 9, CRC Press. [Pages 39, 40 and 41].

[97] KIM, S., ALONSO, J. J. AND JAMESON, A. (2000) Two-dimensional High-Lift Aerodynamic Optimization Using the Continuous Adjoint Method. *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, no. AIAA 2000–4741. [Page 19].

[98] Kinsey, D. W. and Barth, T. J. (1984) *Description of a hyperbolic grid generation procedure for arbitrary two dimensional bodies*. Technical Report TM 84-191-FIMM, AFWAL. [Page 45].

[99] Klöckner, A. (2010) *High-Performance High-Order Simulation of Wave and Plasma Phenomena*. PhD thesis, Brown University. [Page 177].

[100] Koop, A. H. (2008) *Numerical simulation of Unsteady Three-Dimensional Sheet Cavitation*. PhD thesis, Universiteit Twente. [Page 78].

[101] Koren, B. (1989) *Multigrid and Defect Correction for the Steady Navier-Stokes Equations*. PhD thesis, Technische Universiteit Delft. [Page 168].

[102] Koren, B. (1995) Upwind discretization of the steady Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 11, pp. 99–117. [Page 83].

[103] Krakos, J. A. and Darmofal, D. L. (2009) Effect of Small-Scale Unsteadiness on Adjoint-Based Output Sensitivity. *19th AIAA Computational Fluid Dynamics*, San Antonio, TX, no. AIAA 2009–4274. [Page 168].

[104] Krakos, J. A. and Darmofal, D. L. (2010) Effect of Small-Scale Output Unsteadiness on Adjoint-Based Sensitivity. *AIAA Journal*, 48, pp. 2611–2623. [Page 168].

[105] Kulfan, B. M. (2008) Universal Parametric Geometry Representation Method. *Journal of Aircraft*, 45, pp. 142–158. [Page 20].

[106] Kundu, P. K. and Cohen, I. M. (2004) *Fluid Mechanics*. Elsevier Academic Press, San Diego. [Page 76].

[107] Lee, Y. (2008) *On Overset Grids Connectivity and Vortex Tracking in Rotorcraft CFD*. PhD thesis, University of Maryland. [Pages 16, 53, 68 and 176].

[108] Lépine, J. (1999) *Optimisation de la représentation de profils d'ailes pour application au design aérodynamique*. Master's thesis, École Polytechnique de Montréal. [Page 26].

[109] Lépine, J., Guibault, F., Trépanier, J.-Y. and Pépin, F. (2001) Optimized Nonuniform Rational B-Spline Geometrical Representation for Aerodynamic Design of Wings. *AIAA Journal*, 39, pp. 2033–2041. [Pages 19 and 26].

[110] Liu, X.-D., Osher, S. and Chan, T. (1994) Weighted Essentially Non-oscillatory Schemes. *Journal of Computational Physics*, 115, pp. 200–212. [Page 175].

[111] Lyness, J. N. (1967) Numerical Algorithms based on the theory of complex variable. *Proceedings of the 1967 22nd national conference*. [Page 9].

[112] Lyu, Z., Kenway, G., Paige, C. and Martins, J. R. R. A. (2013) Automatic Differentiation Adjoint of the Reynolds-Averaged Navier-Stokes Equations with a Turbulence Model. *43rd AIAA Fluid Dynamics Conference and Exhibit*, San Diego, CA, no. AIAA 2013–2581. [Pages 128, 135 and 152].

[113] Lyu, Z. and Martins, J. R. R. A. (2013) Aerodynamic Shape Optimization of a Blended-Wing-Body Aircraft. *51st AIAA Aerospace Sciences Meeting*, Grapevine, TX, no. AIAA 2013–0283. [Pages 6 and 120].

[114] MARINUS, B. G. (2010) Influence of parametrization and optimization method on the optimum airfoil. *ICAS 2010*. [Pages 17, 20, 21 and 174].

[115] MARTA, A. C. (2007) *Rapid development of discrete adjoint solvers with applications to magnetohydrodynamic flow control*. PhD thesis, Stanford University. [Pages 3, 7, 10, 11, 13 and 15].

[116] MARTINS, J. R. R. A. (2002) *A Coupled-Adjoint Method for High-fidelity Aero-Structural Optimization*. PhD thesis, Stanford University. [Pages 6, 12, 15 and 124].

[117] MARTINS, J. R. R. A., STURDZA, P. AND ALONSO, J. J. (2001) The Connection Between The Complex-Step Derivative Approximation And Algorithmic Differentiation. *39th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, no. AIAA 2001–0921. [Page 124].

[118] MARTINS, J. R. R. A., STURDZA, P. AND ALONSO, J. J. (2003) The Complex-Step Derivative Approximation. *ACM Transactions on Mathematical Software*, 23, pp. 245–262. [Page 9].

[119] MCHUGH, P., KNOLL, D. AND KEYES, D. (1998) Application of Newton-Krylov-Schwarz Algorithm to Low-Mach-Number Compressible Combustion. *AIAA Journal*, 36, pp. 290–292. [Page 94].

[120] MEAKIN, R. L. (1999) Composite Overset Structured Grids. *Handbook of Grid Generation [185]*, chap. 11, CRC Press. [Page 5].

[121] MEAKIN, R. L. (2001) Object X-Rays for Cutting Holes in Composite Overset Structured Grids. *15th AIAA Computational Fluid Dynamics Conference*, Anaheim, CA, no. AIAA 2001–2537. [Page 65].

[122] MILNE-THOMSON, L. M. (1973) *Theoretical aerodynamics*. Dover Publications, New York. [Pages 99 and 204].

[123] VON MISES, R. (1959) *Theory of Flight*. Dover Books on Aeronautical Engineering Series, Dover Publications, New York. [Page 204].

[124] MÖBIUS, A. F. (1827) *Der barycentrische Calcul: Ein Neues Hülfsmittel zur Analytischen Behandlung der Geometrie*. Johann Ambrosius Barth, Leipzig, Germany. [Page 67].

[125] MOHAMMADI, B. (1997) A New Optimal Shape Design Procedure for Inviscid and Viscous Turbulent Flows. *International Journal for Numerical Methods in Fluids*, 25, pp. 183–203. [Page 19].

[126] MÖLLER, J. (2005) *Aspects of The Recursive Projection Method applied to Flow Calculations*. PhD thesis, Kungliga Tekniska högskolan. [Page 168].

[127] MORTON, K. W. AND MAYERS, D. F. (2005) *Numerical Solution of Partial Differential Equations*. Cambrige University Press, Cambridge, United Kingdom, 2nd edn. [Pages 47 and 78].

[128] MOUSAVI, A., CASTONGUAY, P. AND NADARAJAH, S. (2007) Survey of Shape Parameterization Techniques and its Effect on Three-Dimensional Aerodynamic Shape Optimization. *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, no. AIAA 2007–3837. [Page 20].

[129] NICHOLS, R. H., TRAMEL, R. W. AND BUNING, P. G. (2006) Solver and Turbulence Model Upgrades to OVERFLOW 2 for Unsteady and High-Speed Applications. *24th AIAA Applied Aerodynamics Conference*, San Francisco, CA, no. AIAA 2006–2824. [Page 54].

[130] NIELSEN, E. J. (1998) *Aerodynamic Design Sensitivities on an Unstructured Mesh using the Navier-Stokes equations and a discrete adjoint formulation*. PhD thesis, Virginia State University. [Pages 3 and 15].

[131] NIELSEN, E. J. (2009) Discrete Adjoint-Based Design Optimization of Unsteady Turbulent Flows on Dynamic Unstructured Grids. *19th AIAA Computational Fluid Dynamics*, San Antonio, TX, no. AIAA 2009-3802. [Page 15].

[132] NIELSEN, E. J. AND DISKIN, B. (2009) Discrete Adjoint-Based Design for Unsteady Turbulent Flows On Dynamic Overset Unstructured Grids. *50th AIAA Aerospace Sciences Meeting*, Nashville, TN, no. AIAA 2012–0554. [Pages 15 and 143].

[133] OBERKAMPF, W. L. AND TRUCANO, T. G. (2002) Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, 38, pp. 209–272. [Pages 102, 107 and 114].

[134] OLIVER, T. A. (2008) *A High-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations*. PhD thesis, Massachusetts Institute of Technology. [Page 175].

[135] OYAMA, A., OBAYASHI, S. AND NAKAHASHI, K. (1999) Fractional Factorial Design of Genetic Coding for Aerodynamic Optimization. *14th Computational Fluid Dynamics Conference*, Norfolk, VA, no. AIAA 99–3298. [Pages 19, 20 and 175].

[136] PARKS, S. J., BUNING, P. G., STEGER, J. L. AND CHAN, W. M. (1991) Collar grids for intersecting geometric components within the Chimera overlapped grid scheme. *10th AIAA Computational Fluid Dynamics*, Honolulu, HI, no. AIAA 91–1587-CP. [Page 162].

[137] PAWLOWSKI, R. P., SHADID, J. N., SIMONIS, J. P. AND WALKER, H. F. (2004) *Globalization techniques for Newton-Krylov methods and apllications to the fully-coupled solution of the Navier-Stokes equations*. Technical Report SAND2004-1777, Sandia National Laboratories. [Page 176].

[138] PETRINI, E., EFRAIMSSON, G. AND NORDSTRÖM, J. (1998) *A Numerical Study of the Introduction and Propagation of a 2-D Vortex*. Technical Report FFA TN 1998-66, The Aeronautical Research Institute of Sweden, Bromma, Sweden. [Page 213].

[139] PIEGL, L. AND TILLER, W. (1997) *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA. [Page 22].

[140] PIRONNEAU, O. (1973) On optimum profiles in Stokes flow. *Journal of Fluid Mechanics*, 59, pp. 117–128. [Page 14].

[141] PIRONNEAU, O. (1974) On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64, pp. 97–110. [Page 14].

[142] POPE, S. B. (2005) *Turbulent Flows*. Cambridge University Press, Cambridge. [Page 175].

[143] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992) *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA. [Page 26].

[144] Pryce, J. D. and Reid, J. K. (1998) *AD01, a Fortran 90 code for automatic differentiation*. Technical Report RAL-TR-1998-057, Rutherford Appleton Laboratory. [Page 12].

[145] Richardson, L. F. (1911) The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam. *Philosophical Transactions of the Royal Society of London*, 210, pp. pp. 307–357. [Page 98].

[146] Rizzi, A. (1978) Numerical implementations of solid-body boundary conditions for the euler equations. *Zeitschrift für angewandte Mathematik und Mechanik*, 58, pp. 301–304. [Page 86].

[147] Roe, P. L. (1981) Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 135, pp. 250–258. [Pages 3, 27 and 80].

[148] Román, D. S. (2011) *An Assessment of Aeronautical CFD Acceleration on Reconfigurable Computers using High Level Languages and Floating Point Arithmetic*. Master's thesis, Universidad Autónoma de Madrid. [Page 177].

[149] Rumpfkeil, M. P. (2013) Optimizations Under Uncertainty Using Gradients, Hessians, and Surrogate Models. *AIAA Journal*, 51, pp. 444 – 451. [Page 178].

[150] Saad, Y. (2003) *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edn. [Pages 42, 91 and 93].

[151] Saad, Y. and Schultz, M. H. (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7, pp. 856–869. [Page 91].

[152] Sacher, P. (1985) Numerical solutions for three-dimensional cases — Swept wings. *AGARD Advisory Report No. 211, Test Cases for Inviscid Flow Field Methods*, chap. 7. [Page 114].

[153] Saff, E. B. and Snider, A. D. (2003) *Fundamentals of complex analysis with applications to Engineering and Science*. Prentice-Hall Englewood Cliffs, NJ, Upper Saddle River, New Jersey, 3rd edn. [Page 9].

[154] Salas, M. D. (2006) Some observations on grid convergence. *Computers & Fluids*, 35, pp. 688–692. [Page 98].

[155] Samareh, J. A. (2001) Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization. *AIAA Journal*, 39, pp. 877–884. [Page 20].

[156] Samet, H. (1990) *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. [Page 55].

[157] Schepers, J. G. (2012) *Engineering models in wind energy aerodynamics*. PhD thesis, Technische Universiteit Delft. [Page 2].

[158] Schmitt, V. and Charpin, F. (1979) Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers. *AGARD Advisory Report No. 138, Experimental data base for computer program assessment*, chap. B1. [Pages 114 and 119].

[159] Schwamborn, D., Gerhold, T. and Heinrich, R. (2006) The DLR TAU-Code: Recent Applications in Research and Industry. *Proceedings of European Conference on Computational Fluid Dynamics ECCOMAS CFD*. [Page 54].

[160] Schwarz, T. (2005) *Ein blockstrukturiertes Verfahren zur Simulation der Umströmung komplexer Konfigurationen*. PhD thesis, Technischen Universität Carolo-Wilhelmina. [Pages 16, 54 and 176].

[161] Schwarz, T. (2009) An Interpolation Method Maintaining the Wall Distance for Structured and Unstructured Overset Grids. *CEAS 2009*. [Page 176].

[162] Schwarz, T., Spiering, F. and Kroll, N. (2010) Grid coupling by means of Chimera interpolation techniques. *Second Symposium: Simulation of Wing and Nacelle Stall*. [Page 54].

[163] Shanno, D. F. (1970) Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24, pp. 647–656. [Page 149].

[164] Sherman, J. and Morrison, W. J. (1950) Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics*, 21, pp. 124–127. [Page 48].

[165] Shroff, G. M. and Keller, H. B. (1993) Stabilization of Unstable Procedures: The Recursive Projection Method. *SIAM Journal on Numerical Analysis*, 30, pp. 1099–1120. [Page 168].

[166] Sieros, G., Chaviaropoulos, P., Sørensen, J. D., Bulder, B. H. and Jamieson, P. (2012) Upscaling wind turbines: theoretical and practical aspects and their impact on the cost of energy. *Wind Energy*, 15, pp. 3–17. [Page 166].

[167] Smagorinsky, J. (1963) General Circulation Experiments with the Primitive Equations. *Monthly Weather Review*, 91, pp. 99–164. [Page 3].

[168] Smith, S. C. (1996) *A Computational and Experimental Study of Nonlinear Aspects of Induced Drag*. Technical Report 3598, NASA. [Page 160].

[169] Sobieczky, H. (1998) Parametric airfoils and wings. Fuji, K. and Dulikravich, G. S. (eds.), *Notes on Numerical Fluid Mechanics*, vol. 68, pp. 71–88, Vieweg Verlag Wiesbaden. [Page 19].

[170] Sohn, M. and Lee, K. (2000) Bézier Curve Application in the Shape Optimization of Transonic Airfoils. *18th Applied Aerodynamics Conference*, Denver, CO, no. AIAA 2000–4523. [Page 19].

[171] Song, W. and Keane, A. J. (2004) A study of shape parameterisation methods for airfoils optimisation. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, no. AIAA 2004–4482. [Page 20].

[172] Sonneveld, P. and Van Leer, B. (1984) *A Minimax Problem Along the Imaginary Axis*. Department of Mathematics and Informatics, University of Technology. [Page 90].

[173] SPEKREIJSE, S. P. (1987) *Multigrid Solution of the Steady Euler Equations*. PhD thesis, Technische Universiteit Delft. [Page 168].

[174] SPEKREIJSE, S. P. (1995) Elliptic Grid Generation Based on Laplace Equations and Algebraic Transformations. *Journal of Computational Physics*, 118, pp. 38–61. [Pages 34 and 41].

[175] SQUIRE, W. AND TRAPP, G. (1998) Using Complex Variables to Estimate Derivatives of Real Functions. *SIAM Review*, 40, pp. 110–112. [Page 9].

[176] STEGER, J. L. (1989) *Generation of Three-dimensional Body-Fitted Grids by Solving Hyperbolic Partial Differential Equations*. Technical Report 101069, NASA. [Pages 16 and 43].

[177] STEGER, J. L. AND CHAUSSEE, D. S. (1980) Generation of Body-Fitted Coordinates using Hyperbolic Partial Differential Equations. *SIAM Journal on Scientific and Statistical Computing*, 1, pp. 431–437. [Pages 16 and 34].

[178] STEGER, J. L., DOUGHERTY, F. C. AND BENEK, J. A. (1983) A Chimera grid scheme. *Advances in grid generation*. [Page 53].

[179] STRAATHOF, M. H. (2012) *Shape Parametrization in Aircraft Design: A Novel Method, Based on B-Splines*. PhD thesis, Technische Universiteit Delft. [Pages 20, 114 and 152].

[180] STROUSTRUP, B. (2013) *The C++ Programming Language*. Addison-Wesley Professional, Boston, MA, USA, 4th edn. [Page 122].

[181] TANGLER, J. L. AND SOMERS, D. M. (1995) *NREL Airfoil families for HAWTs*. Technical Report NREL/TP-442-7109, National Renewable Energy Laboratory. [Page 160].

[182] THE MPI FORUM (1993), MPI: A Message Passing Interface. [Page 14].

[183] THOMAS, L. H. (1949) *Elliptic Problems in Linear Differential Equations over a Network*. Tech. rep., Columbia University. [Page 47].

[184] THOMAS, P. D. AND LOMBARD, C. K. (1979) Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17, pp. 1030–1037. [Page 97].

[185] THOMPSON, J., SONI, B. AND WEATHERILL, N. (1999) *Handbook of Grid Generation*. CRC Press, Boca Raton, FL, USA. [Pages 181, 185 and 187].

[186] TILLER, W. (1983) Rational B-Splines for Curve and Surface Representation. *IEEE Computer Graphics and Applications*, 3, pp. 61–69. [Pages 19 and 22].

[187] TORO, E. F. (2009) *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer, Berlin, 3rd edn. [Pages 78, 82 and 88].

[188] TRÉPANIER, J.-Y., LÉPINE, J. L. AND PÉPIN, F. (2000) An Optimized Geometric Representation for Wing Profiles Using NURBS. *Canadian Aeronautics and Space Journal*, 46, pp. 12–19. [Pages 26 and 28].

[189] TURING, A. M. (1950) Computing machinery and intelligence. *Mind*, 59, pp. 433–460. [Page 1].

[190] Turkel, E. (1987) Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of Computational Physics*, 72, pp. 277 – 298. [Page 176].

[191] van Kuik, G. A. (2007) The Lanchester–Betz–Joukowsky limit. *Wind Energy*, 10, pp. 289–291. [Page 166].

[192] Vanderplaats, G. N., Hicks, R. N. and Murman, E. M. (1975) Application of Numerical Optimization Techniques to Airfoil Design. *NASA Conference on Aerodynamic Analysis Requiring Advanced Computers, NASA SP-347*. [Page 14].

[193] Van Leer, B. (1979) Towards the Ultimate Conservative Difference Scheme, V. A Second-Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32, pp. 101–136. [Pages 27, 79 and 84].

[194] Vassberg, J. C. and Jameson, A. (2009) In Pursuit of Grid Convergence, Part 1: Two-Dimensional Euler Solutions. *27th AIAA Applied Aerodynamics Conference*, San Antonio, TX, no. AIAA 2009–4114. [Page 98].

[195] Vassberg, J. C. and Jameson, A. (2010) In Pursuit of Grid Convergence for Two-Dimensional Euler Solutions. *Journal of Aircraft*, 47, pp. 1152–1166. [Pages 28, 98, 102, 112, 113, 203 and 204].

[196] Venner, C. and Lubrecht, A. (2000) *Multilevel Methods in Lubrication*. Tribology Series, Elsevier. [Page 42].

[197] Verhoeff, A. J. J. (2005) *Aerodynamics of wind turbine rotors*. PhD thesis, Universiteit Twente. [Page 96].

[198] Vinokur, M. (1983) On One-Dimensional Stretching Functions for Finite-Difference Calculations. *Journal of Computational Physics*, 50, pp. 215–234. [Pages 36 and 38].

[199] De Vries, H. (2013) *On Synthetic Jet Actuation for Aerodynamic Load Control*. PhD thesis, Universiteit Twente. [Pages 81, 88 and 96].

[200] Waggoner, E. G., Burt, M., Lekoudis, S., Kaynak, U., Hirschel, E. H. and Körner, H. (1994) CFD Requirements for Code Validation. *AGARD-AR-303 A Selection of Experimental Test Cases for the Validation of CFD Codes*, chap. 2. [Page 33].

[201] Weibull, W. (1951) A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18, pp. 293–297. [Page 166].

[202] Van der Weide, E. T. A. (1998) *Compressible Flow Simulation on Unstructured Grids using Multi-dimensional Upwind Schemes*. PhD thesis, Technische Universiteit Delft. [Pages 88 and 114].

[203] Wilcox, D. C. (1993) *Turbulence Modeling for CFD*. DCW Industries, Inc., La Cañada. [Page 175].

[204] Wu, H. Y., Yang, S., Lui, F. and Tsai, H. M. (2003) Comparison of three geometric representations of airfoils for aerodynamic optimization. *16th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, no. AIAA 2003–4095. [Page 20].

[205] Yee, H., Sandham, N. and Djomehri, M. (1999) Low-Dissipative High-Order Shock-Capturing Methods Using Characteristic-Based Filters. *Journal of Computational Physics*, 150, pp. 199 – 238. [Page 213].

[206] Zahle, F. (2007) *Wind Turbine Aerodynamics Using an Incompressible Overset Grid Method*. PhD thesis, Imperial College London. [Pages 16, 46 and 54].

[207] Zhang, H., Reggio, M., Trépanier, J. and Camarero, R. (1993) Discrete form of the GCL for moving meshes and its implementation in CFD schemes. *Computers & Fluids*, 22, pp. 9 – 23. [Page 97].

INDEX

A
A Brief History of Time *see* Hawking, Stephen
Abbott, Edwin
  quote 33
adjoint equation method 12–13, 125–127
  discrete 26, 125–126
  grid 176
  Jacobian-free 176–177
  unsteady 168, 169, 177
  verification 141–143
adjoint equations 126
  solution method 140–141
adjoint vector 126
ADT *see* alternating binary tree
algorithmic differentiation 10–12
  forward mode 11
  operator overloading 11–12, 124
  reverse mode 11, 176–177
  source code transformation 11
alternating binary tree 55–58
  generation 55–56
  search procedure 56
  terminal leaf 56
alternating digital tree *see* alternating binary tree
analytical differentiation 7
annual energy production 166
anti-dissipative effect 104, 105
Argand diagram 93
automatic differentiation *see* algorithmic differentiation

B
background grids 49–50
  Cartesian 49–50
  cylindrical 50
  rhombus 50
barycentric coordinate 67
basis spline curve 21–24
  knot vector 21–23
  order 21
BEM *see* blade element momentum theory
Bessel function 8
Betz limit 166
blade element momentum theory 2
blade mass 165
bounding box
  Cartesian 55
  transformed 69, *70*
boxcar function 22

C
C++ 122, 140, 144
cache miss 123
CAD *see* computer aided design
calorically perfect gas 77
cancellation of errors 102, 107
capacity factor 166
Cauchy-Riemann equations 9
ceiling operation 214
central-difference approximation 8
centrifugal force 96
chromatic number 128, 129
co-rating frame of reference *see* non-inertial frame of reference
colour 127
colouring
  $\partial \boldsymbol{R}/\partial \boldsymbol{X}$ 135
  Jacobian matrix 128, 129
complex-step finite-difference method 9, 15
composite overset grid *see* overset grid
computational coordinate 39
compute block *14*, 55
computer aided design 7, 171
condition number 93
conformal mapping 99, 203
conservation laws 76
conserved variabe
  vector 77
constraint
  curvature 151
  internal volume 151–152
  planform area 151
  trailing edge angle 151
continuum hypothesis 76
control point 21
control polygon 24
control volume 76, 77
  centre of mass 79

  volume 79
control-volume-averaged value 77
convective flux discretization
  JST scheme .... *see* Jameson-Schmidt-Turkel scheme
  Roe scheme . *see* Roe's approximate Riemann solver
convective flux tensor 77
convergence acceleration method
  grid sequencing 177
  multigrid 177
Coriolis force 96
covariance 214
Cox-de Boor recursion formula 22
curvature correction 176
curvature-weighted arc length 37
curve
  arc length 36
  curvature 36
Cynegetica .... *see* Oppian

D

defect-correction 168
direct numerical simulation 3
discontinuous Galerkin method 175
discrete adjoint equations . . *see* adjoint equation method
DNS .... *see* direct numerical simulation
domain connectivity 68–74
donor cell *54*, 69
double-precision arithmetic 9
drag coefficient
  aerofoil 100
  wing 117
dual number 121–125
  definition 10
  method 10
    approach 124–125
    vector mode 123–125, 213–216
  non-real part 10

E

epistasis analysis 175
Euler equations *77*
  hyperbolic nature 34, 76
  rotational invariance property 78
  stencil 128
Euler's identity 122
Euler, Leonhard
  quote 121

F

feasibility *149*
Feynman, Richard
  quote 171
field cell 54
field grid generation 43–49
  boundary conditions 48–49
    periodic 48–49
    splay 49
    symmetry 48
  dissipation coefficient 45–47
  finite-difference operators 45
  grid spacing 46
  hyperbolic equations 43
    discretization 43–45
  implicit weighting factor 45
  orthogonality measure 51
  solution method 47–48
Field programmable gate array .... *see* reconfigurable hardware
finite volume method 77–79
  boundary conditions 85–89
    far-field 87
    overset grids 88–89
    periodicity 86
    solid wall 85–86
    subsonic outflow 88
    symmetry 87
finite-difference approximation 7–8
first-order optimization method .... *see* gradient-based optimization method
Flatland .... *see* Abbott, Edwin
floor operation 129
force coefficient
  drag 100, 117
  lift 100, 117
FPGA .... *see* reconfigurable hardware
Fréchet derivative 95
fringe cell *54*, 68
  incomplete .... *see* incomplete fringe cell
function template .... *see* template function

G

Galerkin-Petrov condition 91
Gauss' divergence theorem 151
Gauss-Legendre quadrature 152
GCL .... *see* geometric conservation law
generalized complex number . . . *see* dual number
geometric conservation law 97
ghost control volume . . . *see* halo control volume
global optimization method .... *see* zeroth-order optimization method
GMRES method 91–92
  matrix-free 95–96
  restarted 94
GPU .... *see* graphical processing unit
gradient-based optimization method 6, 178
graph vertex colouring 127–129
  proper *127*
graphical processing unit 177
grid generation
  considerations 34–35
  elliptic 34
  field grid .... *see* field grid generation
  hyperbolic 34–35
  requirements 34
  surface grid ...... *see* surface grid generation

  volume grid . . . . *see* field grid generation
Gulliver's Travels . . . . *see* Swift, Jonthan

H

halo control volume 85
Hawking, Stephen
  quote 75, 147
helical vortex .167, 168, 175
Hermitian cubic interpolation 176
Hessenberg matrix 92
Hessian matrix 149
hole cell 54

I

ILU decomposition . . *see* incomplete lower-upper decomposition
imaginary part 9
incomplete fringe cell 72
incomplete lower-upper decomposition 93
inexact Newton backtracking method 176
interpolation coefficients 70–71

J

Jacobian matrix 92, 129–134
  first-order *92*
  verification 133–134
Jameson-Schmidt-Turkel scheme 80–81
  background dissipation term 80
  dissipation coefficients 81
  pressure sensor 81
  shock capturing term 80
JST scheme . . . . *see* Jameson-Schmidt-Turkel scheme

K

Kármán-Trefftz conformal transformation . . . 99, 204
knot multiplicity 22–23
knot vector 21–23
Krylov subspace methods 91–92

L

Lagrange multiplier .12, 149
large-eddy simulation 3
lift coefficient
  aerofoil 100
  wing 117
low Mach number preconditioning 176

M

Maclaurin series 122
magnetohydrodynamics 15
maximum principle 83, *84*
Message Passing Interface 14
MHD . . . . *see* magnetohydrodynamics
midpoint quadrature rule 99, 100
moment coefficient
  aerofoil 100
MPI . . . . *see* Message Passing Interface
MUSCL-type reconstruction . *see* reconstruction

N

NACA 0012 . . .27, 28, 75, 98, 99, 101, 204, 207
NACA 4-digit series 203
Navier-Stokes equations 76
  dimensional analysis 76–77
Newtonian fluid 76
non-inertial frame of reference 96–97
non-uniform rational basis spline
  surface 7, 21–25
NREL phase VI rotor 160
NREL S809 . . . . .160, 161
numerical flux 78, 80–82
NURBS . . . *see* non-uniform rational basis spline

O

odd-even decoupling 80
ONERA D 114
ONERA M6 15, 75, 114–119, 142, 143, 149, 152
operator overloading .11–12
Oppian
  quote 53
optimality *149*
optimization . . . . *see* shape optimization
overset block connectivity . . . . *see* domain connectivity
overset grid 4–5, 35

P

parallel implementation 14
parametrization method . . . . *see* shape parametrization
perfect gas law 77
Petersen graph 128
PETSc 92, 139
pitching moment 100
power coefficient 164
preconditioning 93–94
pseudo-time-integration 89–96
  Newton's method 90–96
    quadratic convergence 133–134
    solution strategy .94–96
  Runge-Kutta method 89–90

Q

quote
  Abbott, Edwin (*Flatland*) 33
  Euler, Leonard 121
  Feynman, Richard 171
  Hawking, Stephen .75, 147
  Oppian (*Cynegetica*) 53
  Swift, Jonathan (*Gulliver's Travels*) 17
  Turing, Alan 1
  Whitehead, Alfred 217

R

RANS . . . . *see* Reynolds-averaged Navier-Stokes equations

ratio of specific heats 77
rational basis spline curve .24
ray-casting 65–67
  approach 65–66
  concept 65
reconfigurable hardware 177
reconstruction 83–85
  about centre of mass 83
  Cartesian grid assumption 84
  MUSCL-type 27, 84–85
recursive projection method 168
Reynolds-averaged Navier-Stokes equations . . . 3
Richardson extrapolation 98
Riemann-invariant 88
Roe's approximate Riemann solver . . . 27, 81–82
  entropy fix 82
Roe-averaged variable 82
root bending moment 159, 165, 167
rotor thrust 166

S

sensitivity analysis 7–13
shape optimization 5–7
  aerofoil 25–28
  flow chart
    adjoint based 148
    general 6
  multipoint 178
  under uncertainty 178
shape parametrization 6–7
  Chebychev polynomials 19
  Class function/ Shape function Transformation 20
  CST . . . . . *see* Class function/ Shape function Transformation
  direct parametrization .21
  Hicks-Henne bump functions 19
  indirect parametrization 21
  mesh points 19
  PARSEC 19–20
  splines 19
  wind turbine rotor blade 29–30
signed-volume check 66–67
SNOPT 26, 149
spatial order of convergence 98
spurious oscillations 84
stochastic variable 214
subtractive cancellation error 8
surface grid generation 35–42
  elliptical 40–42
    control function 41
    solution procedure 41–42
  transfinite interpolation 38–40
Swift, Jonathan
  quote 17

T

Taylor series expansion
  complex-valued analytical function 9
  dual-number-valued analytical function 10
  real-valued analytical function 7
template function 122–123, 144
time-integration . . . . *see* pseudo-time-integration
tip-cap 150
tree 55
tri-linear transformation 69–70, *71*
Turing, Alan
  quote 1

V

Van Albada limiter 27
variance 215
vertex distribution 35–38
  curvature based 36–37
  Vinokur's stretching function 38
volume grid generation . *see* field grid generation
vortex shedding 169
vortex-correction boundary condition . . 102, 112, 113

W

Weibull distribution 166
WENO scheme 175
Whitehead, Alfred
  quote 217
wind energy 1
wing span efficiency 156

Z

zeroth-order optimization method 5–6, 178
zipper grid 58–64
  generation 58–62
  use 63–64
  verification 62–63

Appendix $\mathcal{A}$

METHOD FOR COMPUTING GRID GENERATION DISSIPATION COEFFICIENT

THE dissipation coefficient — given by equation (3.37) and equation (3.38) on page 47 — is used as a smoothing mechanism in the hyperbolic field grid generation method. Moreover, its application improves the numerical stability of the solution method. The amount of smoothing required for this purpose is dependent on the local properties of the grid. Therefore, this observation is taken into account, by making the dissipation coefficient spatially varying, with the value dependent on these properties. This appendix presents the method that is used to determine the value of the dissipation coefficient. The content of this appendix is based on the original description of the method to determine the dissipation coefficient, presented by Chan and Steger [39].

Apart from the user-definable smoothing parameter ϵ_i, the dissipation coefficient depends on four properties of the grid:

(i) local grid spacing;

(ii) occurrence of converging grid lines;

(iii) normal distance from the body surface;

(iv) occurrence of concave corners.

How each of these properties is accounted for in the computation of the dissipation coefficient is explained in the following sections.

Local grid spacing

The local grid spacing is represented by the approximation of the local matrix norms of $||\underline{\underline{C}}^{-1}\underline{\underline{A}}||$ and $||\underline{\underline{C}}^{-1}\underline{\underline{B}}||$, denoted by N_ξ and N_η, respectively. These approximations yield

$$N_\xi = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\xi^2 + y_\xi^2 + z_\xi^2}}\,, \qquad N_\eta = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\eta^2 + y_\eta^2 + z_\eta^2}}\,. \tag{A.1}$$

For the derivatives involved in this equation, a finite-difference approximation can be employed.

Converging grid lines

In regions of converging grid lines, additional dissipation is required to prevent grid lines from crossing. Converging grid lines are detected with the grid point distribution sensor functions $\tilde{d}^{\xi}_{\mathrm{i,j,k}}$ and $\tilde{d}^{\eta}_{\mathrm{i,j,k}}$, which read

$$\tilde{d}^{\xi}_{\mathrm{i,j,k}} = \max\left\{\left(d^{\xi}_{\mathrm{i,j,k}}\right)^{2/S_{\mathrm{k}}}, 0.1\right\}, \tag{A.2}$$

$$\tilde{d}^{\eta}_{\mathrm{i,j,k}} = \max\left\{\left(d^{\eta}_{\mathrm{i,j,k}}\right)^{2/S_{\mathrm{k}}}, 0.1\right\}, \tag{A.3}$$

where

$$d^{\xi}_{\mathrm{i,j,k}} = \frac{|\boldsymbol{r}_{\mathrm{i+1,j,k-1}} - \boldsymbol{r}_{\mathrm{i,j,k-1}}| + |\boldsymbol{r}_{\mathrm{i-1,j,k-1}} - \boldsymbol{r}_{\mathrm{i,j,k-1}}|}{|\boldsymbol{r}_{\mathrm{i+1,j,k}} - \boldsymbol{r}_{\mathrm{i,j,k}}| + |\boldsymbol{r}_{\mathrm{i-1,j,k}} - \boldsymbol{r}_{\mathrm{i,j,k}}|}, \tag{A.4}$$

$$d^{\eta}_{\mathrm{i,j,k}} = \frac{|\boldsymbol{r}_{\mathrm{i,j+1,k-1}} - \boldsymbol{r}_{\mathrm{i,j,k-1}}| + |\boldsymbol{r}_{\mathrm{i,j-1,k-1}} - \boldsymbol{r}_{\mathrm{i,j,k-1}}|}{|\boldsymbol{r}_{\mathrm{i,j+1,k}} - \boldsymbol{r}_{\mathrm{i,j,k}}| + |\boldsymbol{r}_{\mathrm{i,j-1,k}} - \boldsymbol{r}_{\mathrm{i,j,k}}|}. \tag{A.5}$$

i.e. the ratio of the distance between grid vertices from two subsequent grid layers, i.e. layer $(\mathrm{k}-1)$ and layer k. This ratio is higher in concave regions, resulting in the application of more dissipation in these regions. In equation (A.2) and equation (A.3) S_{k} denotes the scaling function, which scales for the normal distance to the wall.

Normal distance

The normal distance to the wall is accounted for by means of scaling function S_{k}, which is expressed as

$$S_{\mathrm{k}} = \begin{cases} \sqrt{\frac{\mathrm{k}}{\mathrm{n}_3-1}} & \text{if} \quad 1 \le \mathrm{k} \le \mathrm{k}_{\text{trans}}, \\ \sqrt{\frac{\mathrm{k}_{\text{trans}}}{\mathrm{n}_3-1}} & \text{if} \quad \mathrm{k}_{\text{trans}}+1 \le \mathrm{k} < \mathrm{n}_3, \end{cases} \tag{A.6}$$

where n_3 is the number of vertices in the third direction. Parameter $\mathrm{k}_{\text{trans}}$ is restricted to the range $[3/4, 1] \times (\mathrm{n}_3 - 1)$ and is determined based on the local properties of the grid. The value of $\mathrm{k}_{\text{trans}}$ is set to the value of the current grid layer, if one of the following conditions is true:

$$\max_{\mathrm{i,j}} d^{\xi}_{\mathrm{i,j,k}} - \max_{\mathrm{i,j}} d^{\xi}_{\mathrm{i,j,k-1}} < 0, \tag{A.7}$$

$$\max_{\mathrm{i,j}} d^{\eta}_{\mathrm{i,j,k}} - \max_{\mathrm{i,j}} d^{\eta}_{\mathrm{i,j,k-1}} < 0, \tag{A.8}$$

which is a measure for a sufficient decrease in the convergence of the grid lines. Once $\mathrm{k}_{\text{trans}}$ is assigned a value, it remains unchanged. Application of the scaling function makes sure that the dissipation close to the body surface is small, such that orthogonality of the grid lines is maintained in that region.

Concave corners

The grid angle functions $\tilde{a}^{\xi}_{\mathrm{i,j,k}}$ and $\tilde{a}^{\eta}_{\mathrm{i,j,k}}$ are used to detect the occurrence of a concave corner in the grid. For that purpose, define the vectors pointing in the positive and negative ξ-direction, respectively, as:

$$\boldsymbol{r}^{+}_{\mathrm{i}} := \boldsymbol{r}_{\mathrm{i+1,j,k}} - \boldsymbol{r}_{\mathrm{i,j,k}}, \qquad \boldsymbol{r}^{-}_{\mathrm{i}} := \boldsymbol{r}_{\mathrm{i-1,j,k}} - \boldsymbol{r}_{\mathrm{i,j,k}}. \tag{A.9}$$

The vectors pointing in the positive and negative η-direction are defined in a similar fashion and the corresponding unit vectors are denoted by a hat. Then, the local unit normal vector is computed by the following cross product

$$\hat{\boldsymbol{n}}_{\mathrm{i,j,k}} = \frac{\left(\boldsymbol{r}^{+}_{\mathrm{i}} - \boldsymbol{r}^{-}_{\mathrm{i}}\right) \times \left(\boldsymbol{r}^{+}_{\mathrm{j}} - \boldsymbol{r}^{-}_{\mathrm{j}}\right)}{\left|\left(\boldsymbol{r}^{+}_{\mathrm{i}} - \boldsymbol{r}^{-}_{\mathrm{i}}\right) \times \left(\boldsymbol{r}^{+}_{\mathrm{j}} - \boldsymbol{r}^{-}_{\mathrm{j}}\right)\right|}. \tag{A.10}$$

The dot product of $\hat{\boldsymbol{r}}^{+}_{\mathrm{i}}$ with this unit normal vector equals the cosine of the angle between both vectors, i.e.

$$\cos\left(\alpha_{\mathrm{i,j,k}}\right) = \hat{\boldsymbol{n}}_{\mathrm{i,j,k}} \cdot \hat{\boldsymbol{r}}^{+}_{\mathrm{i}} \equiv \hat{\boldsymbol{n}}_{\mathrm{i,j,k}} \cdot \hat{\boldsymbol{r}}^{-}_{\mathrm{i}}. \tag{A.11}$$

The resulting value for α_{ijk} is subsequently used to compute the grid angle function, according to

$$\tilde{a}^{\xi}_{\mathrm{i,j,k}} = \begin{cases} \left[1 - \cos^2\left(\alpha_{\mathrm{i,j,k}}\right)\right]^{-1} & \text{if} \quad 0 \leq \alpha_{\mathrm{i,j,k}} \leq \frac{\pi}{2}, \\ 1 & \text{if} \quad \frac{\pi}{2} < \alpha_{\mathrm{i,j,k}} \leq \pi. \end{cases} \tag{A.12}$$

Following the same procedure for $\hat{\boldsymbol{r}}^{+}_{\mathrm{j}}$ and $\hat{\boldsymbol{r}}^{-}_{\mathrm{j}}$, the value for grid angle function $\tilde{a}^{\eta}_{\mathrm{i,j,k}}$ is obtained.

Dissipation coefficient function

Each of the aforementioned properties of the grid has been accounted for. Next, the expression for the local grid property dependent dissipation coefficient for smoothing of the implicit part of the hyperbolic field grid generation equations is obtained by taking the product of the functions presented above, resulting in

$$\left(\epsilon_{\xi}\right)_{\mathrm{i,j,k}} = \epsilon_{\mathrm{i}} N_{\xi} S_{\mathrm{k}} \tilde{d}^{\xi}_{\mathrm{i,j,k}} \tilde{a}^{\xi}_{\mathrm{i,j,k}}, \tag{A.13}$$

$$\left(\epsilon_{\eta}\right)_{\mathrm{i,j,k}} = \epsilon_{\mathrm{i}} N_{\eta} S_{\mathrm{k}} \tilde{d}^{\eta}_{\mathrm{i,j,k}} \tilde{a}^{\eta}_{\mathrm{i,j,k}}. \tag{A.14}$$

Appendix $\mathcal{B}$

Kármán-Trefftz O-grid generation

For the verification of the spatial order of convergence of the numerical flow solution method, an algebraic grid generation method has been used, which is based on the concept of conformal mapping. The grid resulting from using this method is of high quality, both in terms of orthogonality of the grid lines, as well as the aspect ratio of the cells. The method presented in this appendix is based on the method described by Vassberg and Jameson [195].

NACA 0012 aerofoil geometry

The geometry used for the verification of the numerical method is based on the symmetric NACA 4-digit series of aerofoils. The shape of the upper surface of these aerofoils is defined by

$$\frac{y\left(x\right)}{\bar{c}}=\frac{\bar{t}}{0.2}\left[0.2969\sqrt{\frac{x}{\bar{c}}}-0.1260\left(\frac{x}{\bar{c}}\right)\right.$$
$$\left.-0.3516\left(\frac{x}{\bar{c}}\right)^2+0.2843\left(\frac{x}{\bar{c}}\right)^3-0.1015\left(\frac{x}{\bar{c}}\right)^4\right], \quad \text{(B.1)}$$

for $^{x}/_{\bar{c}} \in [0,1]$ and where $\bar{c}$ is the chord length of the aerofoil and $\bar{t}$ is the relative maximum thickness. Since the aerofoil is symmetric, the lower surface is obtained by mirroring the geometry with respect to the x-axis. This geometry definition yields a non-zero value for y at $x \equiv \bar{c}$. However, for the purpose of having a sharp trailing edge, the geometry is extended to the point where the curve crosses the x-axis. Following this approach, the trailing edge of the aerofoil is now situated at

$$\frac{\left(x_{\text{te}}-1\right)}{\bar{c}}=8.930411365142709\cdot 10^{-3}.$$

Properties required for performing the conformal transformation, which is discussed in the next section, are the leading edge radius and the trailing edge angle. The leading edge radius is given by [1]:

$$\frac{r_{\text{le}}}{\bar{c}}=1.1019\left(\frac{\bar{t}}{\bar{c}}\right)^2. \quad \text{(B.2)}$$

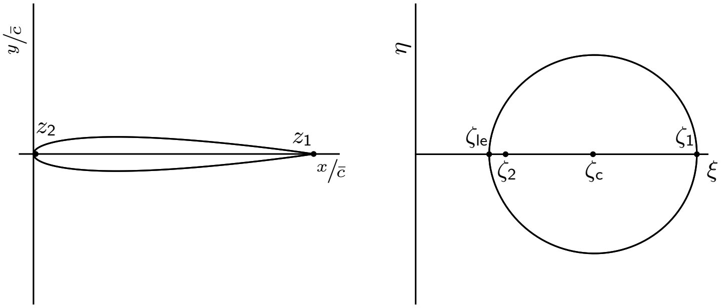


Figure B.1: The NACA 0012 aerofoil, defined by equation (B.1), is transformed to a quasi-circle, using the conformal transformation, defined by equation (5.73). The singular points corresponding to this transformation are ζ_1 and ζ_2 corresponding to z_1 and z_2 respectively, together with two points used in the grid generation procedure $\zeta_{\text{le}} \equiv \zeta(0)$ and ζ_{le}.

The trailing edge angle, for the extended aerofoil, is computed with

$$\tau_{\text{te}} = 2 \arctan \left(\left| \frac{1}{\bar{c}} \left. \frac{\mathrm{d}y}{\mathrm{d}x} \right|_{x_{\text{te}}} \right| \right) = 2.81872535015023 \cdot 10^{-1} [\text{rad}] \approx 16.15^{\circ}. \tag{B.3}$$

Kármán-Trefftz conformal transformation

The Kármán-Trefftz conformal transformation $\mathbb{C} \to \mathbb{C}$ is used to map the NACA 0012 aerofoil to a quasi-circle, such that the discretization can be performed in the ζ-plane. Subsequently, the obtained grid is transformed back to the physical space, resulting in a high quality discretization of the flow domain. The Kármán-Trefftz conformal transformation reads [122, 123]

$$\frac{\zeta - \zeta_1}{\zeta - \zeta_2} = \left(\frac{z - z_1}{z - z_2} \right)^P, \quad P = \frac{\pi}{2\pi - \tau_{\text{te}}}, \tag{5.73}$$

where $z = x + iy$ and $\zeta = \xi + i\eta$. For mapping the NACA 0012 aerofoil, the following singular points in the physical space are placed on the sharp trailing edge location and at half the leading edge radius from the leading edge [195], i.e.

$$z_1 = x_{\text{te}}, \qquad z_2 = \frac{1}{2} r_{\text{le}}.$$

To map the aerofoil to a quasi-circle, the singular points in the mapped space correspond to [195]:

$$\zeta_1 = 0.77043505, \qquad \zeta_2 = 0.24642903.$$

Figure B.1 shows both the NACA 0012 geometry and its shape in the mapped space, after performing the conformal transformation.

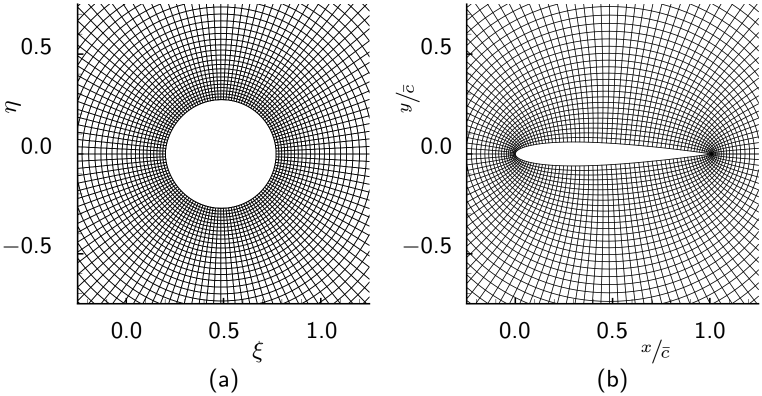


Figure B.2: Flow domain used for studying spatial convergence characteristics of flow solution method. Grid depicted consists of 128 × 128 control volumes. Subfigure (a) shows detail of the grid in the mapped space, (b) shows detail of the grid in physical space after inverse transformation.

Grid generation

For the generation of the grid, the centre of the circle in the mapped space is taken to be at the middle of the line connecting leading edge to trailing edge, i.e.

$$\zeta_{\mathrm{c}} = \frac{\zeta(0) + \zeta(x_{\mathrm{te}})}{2} = 4.85915653521317 \cdot 10^{-1}.$$

Subsequently, the upper half of the quasi-circle is discretized, by placing $\left(\frac{1}{2}\mathrm{N_c} + 1\right)$ vertices in the quasi-circle, with a constant angular spacing of

$$\Delta\theta = \frac{2\pi}{\mathrm{N_c}}.$$

Here $\mathrm{N_c}$ denotes the number of control volumes used for the discretization of the whole aerofoil geometry. For the discretization of the domain with cells that have an aspect ratio of approximately one, the spacing in the radial direction is taken equal to $r\Delta\theta$, where r is the local radius, with respect to ζ_{c}. This goal is achieved by creating concentric circles with radius

$$R_{\mathrm{j}} = R_0 \mathrm{e}^{\frac{2\pi\mathrm{j}}{\mathrm{N_c}}}, \qquad 0 \leq \mathrm{j} \leq \mathrm{N_c} \tag{B.4}$$

around the centre point. In this equation R_0 is the approximate radius of the quasi-circle that represents the aerofoil. It is computed by dividing the perimeter of the quasi-circle by 2π. To account for the fact that the quasi-circle is not perfectly circular, the actual radius on which each vertex is placed is computed according to

$$r_{\mathrm{ij}} = \frac{r_{\mathrm{i0}}\left(R_{\mathrm{N_c}} - R_0\right) + R_{\mathrm{N_c}}\left(R_{\mathrm{j}} - R_0\right)}{R_{\mathrm{N_c}} - R_0}, \quad 0 \leq \mathrm{i} \leq \frac{1}{2}\mathrm{N_c}, \quad 0 \leq \mathrm{j} \leq \mathrm{N_c}. \tag{B.5}$$

The value of r_{i0} can be computed from the discretized upper half of the quasi-circle. For the purpose of creating a grid that is symmetric, the method described above only constructs the upper half of the grid. Subsequently, the lower half of the grid is obtained by mirroring the upper half with respect to the ξ-axis. Figure B.2 (a) shows a part of the grid, in the mapped plane, near the quasi-circle for $\mathrm{N_c} = 128$. Performing the transformation of this grid from the mapped plane back to the physical plane results in the grid depicted in figure B.2 (b).

A family of grids is obtained by creating a grid for the finest discretization to be considered. Subsequent other grids are obtained by removing every other vertex in both directions. This family of grids can then be used for the investigation of the spatial convergence characteristics of the method. Such an investigation has been performed for the flow solution method employed in the present research. The approach and results of this investigation are discussed in section 5.8 starting on page 97.

Appendix C

RESULTS SPATIAL ORDER OF CONVERGENCE STUDY

In section 5.8 the method used for verifying the spatial order of convergence of the flow model has been discussed and a summary of the results of this investigation has been presented. This appendix provides a survey of all results. The flow configuration involves an extended NACA 0012 aerofoil subject to a subcritical flow with a free-stream Mach number of 0.5 and an angle of attack of 1.25°. Several parameters have been varied in the investigation; the parameters used to obtain the results presented are listed in the caption of the corresponding table. A discussion of the results has been given in section 5.8 starting at page 97.

The first column of each table lists the grid resolution, represented by the number of control volumes $\mathrm{N_c}$ used for the discretization of the flow domain in the circumferential direction of the aerofoil. The same number of control volumes is also used for discretization in the radial direction, for the far-field at approximately 150 chord lengths, $\bar{c}$. For the large domain, with a radius of approximately $80 \cdot 10^3\ \bar{c}$, double that number of control volumes is used in the radial direction. The subsequent columns list the lift coefficient c_l, the drag coefficient c_d and the moment coefficient c_m. The bottom two rows of each table list the extrapolated value ϕ^* and corresponding order of convergence $\bar{p}$. These estimates are usually obtained using the three finest grid resolutions. However, in the cases indicated by a dagger, non-monotonic convergence behaviour is observed when the result of the finest grid is included. Therefore, in these situations, the finest grid solution is not considered and the result of the grid containing 512 control volumes in one direction was used for computing the estimate for $h = 0$. The decimal places that have been underlined, correspond to the value estimated for infinite grid resolution.

Table C.1: Results for the single block discretization of the domain, obtained with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = {}^{1}\!/_{3}$; far-field at 150 $\bar{c}$.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.69881209 \cdot 10^{-1}$	$4.11298649 \cdot 10^{-3}$	$3.44522046 \cdot 10^{-3}$
64	$1.\underline{7}5339576 \cdot 10^{-1}$	$5.93413424 \cdot 10^{-4}$	$2.17870475 \cdot 10^{-3}$
128	$1.\underline{7}8563843 \cdot 10^{-1}$	$3.05472095 \cdot 10^{-5}$	$2.17687754 \cdot 10^{-3}$
256	$1.\underline{79}526247 \cdot 10^{-1}$	$-8.82167828 \cdot 10^{-6}$	$2.\underline{2}4151706 \cdot 10^{-3}$
512	$1.\underline{797}35881 \cdot 10^{-1}$	$2.90705235 \cdot 10^{-6}$	$2.\underline{2}5920254 \cdot 10^{-3}$
1024	$1.\underline{797}74289 \cdot 10^{-1}$	$9.34057594 \cdot 10^{-6}$	$2.\underline{262}13760 \cdot 10^{-3}$
2048	$1.\underline{797}79950 \cdot 10^{-1}$	$1.14818243 \cdot 10^{-5}$	$2.\underline{2623}5858 \cdot 10^{-3}$
4096	$1.\underline{79780}313 \cdot 10^{-1}$	$1.\underline{2}1035183 \cdot 10^{-5}$	$2.\underline{262}29847 \cdot 10^{-3}$
ϕ^*	$1.79780338 \cdot 10^{-1}$	$1.23578715 \cdot 10^{-5}$	$2.26237657 \cdot 10^{-3}$
$\bar{p}$	3.961	1.784	3.731 †

Table C.2: Results for the single block discretization of the domain, obtained with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = {}^{1}\!/_{3}$; far-field at 150 $\bar{c}$; modified pressure boundary condition used.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.\underline{7}1225365 \cdot 10^{-1}$	$3.55901040 \cdot 10^{-3}$	$3.35618754 \cdot 10^{-3}$
64	$1.\underline{7}5813493 \cdot 10^{-1}$	$3.21773867 \cdot 10^{-4}$	$2.19042437 \cdot 10^{-3}$
128	$1.\underline{7}8699265 \cdot 10^{-1}$	$-5.88352224 \cdot 10^{-5}$	$2.18598167 \cdot 10^{-3}$
256	$1.\underline{79}558697 \cdot 10^{-1}$	$-3.43448668 \cdot 10^{-5}$	$2.\underline{2}4412609 \cdot 10^{-3}$
512	$1.\underline{797}43102 \cdot 10^{-1}$	$-3.90881356 \cdot 10^{-6}$	$2.\underline{2}5976891 \cdot 10^{-3}$
1024	$1.\underline{797}75848 \cdot 10^{-1}$	$7.57986239 \cdot 10^{-6}$	$2.\underline{262}24080 \cdot 10^{-3}$
2048	$1.\underline{79780}283 \cdot 10^{-1}$	$1.10345247 \cdot 10^{-5}$	$2.\underline{2623}7412 \cdot 10^{-3}$
4096	$1.\underline{797803}85 \cdot 10^{-1}$	$1.19908439 \cdot 10^{-5}$	$2.\underline{2623}0003 \cdot 10^{-3}$
ϕ^*	$1.79780387 \cdot 10^{-1}$	$1.23569050 \cdot 10^{-5}$	$2.26238171 \cdot 10^{-3}$
$\bar{p}$	5.449	1.853	4.213 †

Table C.3: Results for the single block discretization of the domain, obtained with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 0$; far-field at 150 $\bar{c}$; modified pressure boundary condition used.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.68990591 \cdot 10^{-1}$	$7.05997590 \cdot 10^{-3}$	$3.09464283 \cdot 10^{-3}$
64	$1.\underline{7}4743791 \cdot 10^{-1}$	$1.24850520 \cdot 10^{-3}$	$2.05712685 \cdot 10^{-3}$
128	$1.\underline{7}8280146 \cdot 10^{-1}$	$1.77926448 \cdot 10^{-4}$	$2.12465754 \cdot 10^{-3}$
256	$1.\underline{79}425738 \cdot 10^{-1}$	$2.71727681 \cdot 10^{-5}$	$2.\underline{2}2246115 \cdot 10^{-3}$
512	$1.\underline{797}02249 \cdot 10^{-1}$	$1.\underline{2}0028986 \cdot 10^{-5}$	$2.\underline{2}5266869 \cdot 10^{-3}$
1024	$1.\underline{797}63492 \cdot 10^{-1}$	$1.16340609 \cdot 10^{-5}$	$2.\underline{26}003639 \cdot 10^{-3}$
2048	$1.\underline{7977}6578 \cdot 10^{-1}$	$1.\underline{2}0558493 \cdot 10^{-5}$	$2.\underline{26}171927 \cdot 10^{-3}$
4096	$1.\underline{79779}257 \cdot 10^{-1}$	$1.\underline{2}2470654 \cdot 10^{-5}$	$2.\underline{262}11023 \cdot 10^{-3}$
ϕ^*	$1.79779947 \cdot 10^{-1}$	$1.24056429 \cdot 10^{-5}$	$2.26222855 \cdot 10^{-3}$
$\bar{p}$	2.288	1.141	2.106

Table C.4: Results for the single block discretization of the domain, obtained with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, assuming uniform Cartesian grid; linear reconstruction weighting factor $\hat{\kappa} = 0$; far-field at 150 $\bar{c}$; modified pressure boundary condition used.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.65532737 \cdot 10^{-1}$	$-1.57766643 \cdot 10^{-3}$	$2.13255906 \cdot 10^{-3}$
64	$1.\underline{7}2232634 \cdot 10^{-1}$	$-6.49164204 \cdot 10^{-3}$	$1.73581819 \cdot 10^{-3}$
128	$1.\underline{7}6743642 \cdot 10^{-1}$	$-4.76257699 \cdot 10^{-3}$	$2.02229488 \cdot 10^{-3}$
256	$1.\underline{7}8558083 \cdot 10^{-1}$	$-2.74755812 \cdot 10^{-3}$	$2.17647191 \cdot 10^{-3}$
512	$1.\underline{79}248721 \cdot 10^{-1}$	$-1.45803866 \cdot 10^{-3}$	$2.\underline{2}3046829 \cdot 10^{-3}$
1024	$1.\underline{79}535005 \cdot 10^{-1}$	$-7.45009630 \cdot 10^{-4}$	$2.\underline{2}4953016 \cdot 10^{-3}$
2048	$1.\underline{79}663081 \cdot 10^{-1}$	$-3.71797592 \cdot 10^{-4}$	$2.\underline{2}5683221 \cdot 10^{-3}$
4096	$1.\underline{797}23088 \cdot 10^{-1}$	$-1.81079366 \cdot 10^{-4}$	$2.\underline{2}5984132 \cdot 10^{-3}$
ϕ^*	$1.79775988 \cdot 10^{-1}$	$1.82339234 \cdot 10^{-5}$	$2.26195053 \cdot 10^{-3}$
$\bar{p}$	1.094	0.969	1.279

Table C.5: Results for the single block discretization of the domain, obtained with following settings: JST scheme to compute convective flux; dissipation coefficients used: $k_2 = 0.25$, $k_4 = 0.015625$; far-field at 150 $\bar{c}$; modified pressure boundary condition at solid wall. No estimate could be obtained for the spatial convergence order for c_m.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.\underline{7}6309093 \cdot 10^{-1}$	$-2.74721388 \cdot 10^{-3}$	$3.87334592 \cdot 10^{-3}$
64	$1.\underline{7}8507728 \cdot 10^{-1}$	$-1.78304347 \cdot 10^{-4}$	$2.58382219 \cdot 10^{-3}$
128	$1.\underline{79}400636 \cdot 10^{-1}$	$-7.03071555 \cdot 10^{-4}$	$2.33722082 \cdot 10^{-3}$
256	$1.\underline{79}609601 \cdot 10^{-1}$	$-2.10428847 \cdot 10^{-4}$	$2.27502484 \cdot 10^{-3}$
512	$1.\underline{79}662119 \cdot 10^{-1}$	$-4.72503043 \cdot 10^{-5}$	$2.25935564 \cdot 10^{-3}$
1024	$1.\underline{79}685217 \cdot 10^{-1}$	$-1.39285941 \cdot 10^{-6}$	$2.25727058 \cdot 10^{-3}$
2048	$1.\underline{79}698923 \cdot 10^{-1}$	$1.02730972 \cdot 10^{-5}$	$2.25834029 \cdot 10^{-3}$
4096	$1.\underline{7970}7007 \cdot 10^{-1}$	$1.\underline{3}0144502 \cdot 10^{-5}$	$2.25959865 \cdot 10^{-3}$
ϕ^*	$1.79718633 \cdot 10^{-1}$	$1.38565062 \cdot 10^{-5}$	—
$\bar{p}$	0.762	2.089	—

Table C.6: Results for the single block discretization of the domain, obtained with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 1/3$; far-field at 80 $\mathrm{k}\bar{c}$.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.70289358 \cdot 10^{-1}$	$4.10728641 \cdot 10^{-3}$	$3.45350522 \cdot 10^{-3}$
64	$1.75823878 \cdot 10^{-1}$	$5.84468249 \cdot 10^{-4}$	$2.18424316 \cdot 10^{-3}$
128	$1.79093619 \cdot 10^{-1}$	$1.94497523 \cdot 10^{-5}$	$2.18291842 \cdot 10^{-3}$
256	$1.\underline{80}075911 \cdot 10^{-1}$	$-2.06930203 \cdot 10^{-5}$	$2.\underline{2}4799830 \cdot 10^{-3}$
512	$1.\underline{80}294049 \cdot 10^{-1}$	$-9.22571045 \cdot 10^{-6}$	$2.\underline{26}584719 \cdot 10^{-3}$
1024	$1.\underline{803}36340 \cdot 10^{-1}$	$-2.89396662 \cdot 10^{-6}$	$2.\underline{26}883841 \cdot 10^{-3}$
2048	$1.\underline{8034}3874 \cdot 10^{-1}$	$-\ 7.974599 \cdot 10^{-7}$	$2.\underline{269}08260 \cdot 10^{-3}$
4096	$1.\underline{80345}162 \cdot 10^{-1}$	$-\ 1.968521 \cdot 10^{-7}$	$2.\underline{269}03326 \cdot 10^{-3}$
ϕ^*	$1.80345428 \cdot 10^{-1}$	$4.42929 \cdot 10^{-8}$	$2.26910430 \cdot 10^{-3}$
$\bar{p}$	2.548	1.803	3.615 †

Table C.7: Results obtained for composite overset discretization of the domain with following settings: Roe's approximate Riemann solver to compute convective flux; linear reconstruction of the state at the interface, about the centre of mass of the control volume; linear reconstruction weighting factor $\hat{\kappa} = 1/3$; far-field at $150\,\bar{c}$.

grid resolution $\mathrm{N_c}$	c_l	c_d	c_m
32	$1.\underline{7}1594805 \cdot 10^{-1}$	$4.06551234 \cdot 10^{-3}$	$3.44918052 \cdot 10^{-3}$
64	$1.\underline{7}5363561 \cdot 10^{-1}$	$5.81904160 \cdot 10^{-4}$	$2.17323563 \cdot 10^{-3}$
128	$1.\underline{7}8552136 \cdot 10^{-1}$	$3.00330172 \cdot 10^{-5}$	$2.17580020 \cdot 10^{-3}$
256	$1.\underline{79}529489 \cdot 10^{-1}$	$-9.06903892 \cdot 10^{-6}$	$2.\underline{2}4157180 \cdot 10^{-3}$
512	$1.\underline{797}40459 \cdot 10^{-1}$	$2.75543775 \cdot 10^{-6}$	$2.\underline{2}5931637 \cdot 10^{-3}$
1024	$1.\underline{797}78184 \cdot 10^{-1}$	$9.25195908 \cdot 10^{-6}$	$2.\underline{262}19100 \cdot 10^{-3}$
2048	$1.\underline{7978}3864 \cdot 10^{-1}$	$1.13978406 \cdot 10^{-5}$	$2.\underline{2624}0533 \cdot 10^{-3}$
4096	$1.\underline{79784}361 \cdot 10^{-1}$	$1.\underline{2}0153169 \cdot 10^{-5}$	$2.\underline{262}34616 \cdot 10^{-3}$
ϕ^*	$1.79784408 \cdot 10^{-1}$	$1.22647777 \cdot 10^{-5}$	$2.26242260 \cdot 10^{-3}$
$\bar{p}$	3.516	1.797	3.745 †

Appendix $\mathcal{D}$

MEASURING THE EFFICIENCY OF THE DUAL NUMBER METHOD IN VECTOR MODE

To determine the efficiency of the dual number method for computing sensitivities, depending on the dimension of the non-real component of the dual number, a representative model problem has been considered. This appendix presents the details of the model problem considered, as well as the method used to determine the efficiency. Moreover, the approach taken to determine an estimate for the number of operations that is required to fully populate the Jacobian matrix, for a dual number with a certain dimension of the non-real part, is also discussed.

Model problem

The model problem considers the calculation of the residual vector of the 2D Euler equations for unsteady inviscid compressible flow. The partial differential equations have been discretized using a finite-difference method and the convective flux is computed using Roe's approximate Riemann solver. A rectangular flow domain has been used, for which the horizontal dimension is two times the vertical dimension. The domain has been discretized using a Cartesian grid that is uniform in both directions. The flow field is initialized with the superposition of an isentropic vortex [48, 138, 205] and a uniform flow field with a free-steam Mach number of $\mathrm{M}_\infty = 0.50$, as

$$
\begin{aligned}
u(\boldsymbol{x}) &= \sqrt{\gamma}\,\mathrm{M}_\infty \left(1 - \frac{\Gamma}{2\pi R_\mathsf{c}} \mathrm{e}^{\frac{1}{2}\left[1 - \frac{(x-x_0)^2+(y-y_0)^2}{R_\mathsf{c}^2}\right]} (y - y_0)\right), \\
v(\boldsymbol{x}) &= \sqrt{\gamma}\,\mathrm{M}_\infty \frac{\Gamma}{2\pi R_\mathsf{c}} \mathrm{e}^{\frac{1}{2}\left[1 - \frac{(x-x_0)^2+(y-y_0)^2}{R_\mathsf{c}^2}\right]} (x - x_0), \\
\rho(\boldsymbol{x}) &= \left(1 - \frac{[\gamma - 1]\,\Gamma^2\,\mathrm{M}_\infty^2}{8\pi^2} \mathrm{e}^{1 - \frac{(x-x_0)^2+(y-y_0)^2}{R_\mathsf{c}^2}}\right)^{\frac{1}{\gamma-1}}, \\
p(\boldsymbol{x}) &= \left(1 - \frac{[\gamma - 1]\,\Gamma^2\,\mathrm{M}_\infty^2}{8\pi^2} \mathrm{e}^{1 - \frac{(x-x_0)^2+(y-y_0)^2}{R_\mathsf{c}^2}}\right)^{\frac{\gamma}{\gamma-1}},
\end{aligned}
$$

with initial position $(x_0, y_0) = (-0.5, 0)$, radius of the vortex core $R_\mathsf{c} = 0.1$, strength of the vortex $\Gamma = 1.0$ and the ratio of specific heats is taken as $\gamma = 1.4$. Note, that non-dimensional variables are used for both the flow variables — non-dimensionalized using the free-stream conditions — and the coordinates — non-dimensionalized using the dimension of the domain in vertical direction. Boundary conditions are enforced by prescribing the exact solution for the vertices at the boundary.

Timing

As a measure for determining the relative efficiency of the method, an estimate for the time required to fully populate the Jacobian matrix is used. Note, that the possibility of using colouring to construct the Jacobian matrix more efficiently is not considered for this purpose. Considering that (i) the horizontal dimension of the domain is twice the vertical dimension; (ii) the grid is uniform in both directions; (iii) for the 2D Euler equations there are four conserved flow variables and (iv) assuming that an odd number of vertices is used for the discretization of the domain, the number of derivatives that must be computed to populate the Jacobian matrix is given by

$$\mathrm{n}_{\text{der}} = 2\mathrm{n}_{\text{i}}\left(\mathrm{n}_{\text{i}} + 1\right), \tag{D.1}$$

where n_{i} is the number of vertices used for the discretization of the domain in the x-direction. Using this result, the estimate for the time required to populate the Jacobian matrix is defined by

$$\bar{t}_{\text{jac}} := \left\lceil \frac{\mathrm{n}_{\text{der}}}{\mathrm{n}_{\text{d}}} \right\rceil \frac{\Delta t_{\text{res}}}{t_{\text{ref}}}, \tag{D.2}$$

with $\lceil\cdot\rceil$ the ceiling operation, defined for $x \in \mathbb{R}$ as: $\lceil x \rceil := \min\{\mathrm{m} \in \mathbb{Z} \,:\, \mathrm{m} \geq x\}$, n_{d} the dimension of the non-real component of the dual number. Moreover, Δt_{res} is the time required for a single computation of the complete residual vector and t_{ref} a reference time. For a reference time, the time required to fully populate the Jacobian matrix by means of a central-difference finite-difference approximation is used.

To determine Δt_{res}, the time required to perform n_{res} residual evaluations is recorded. This procedure is then repeated for n_{run} times and subsequently, the mean of this result is computed by averaging over the number of runs. The time for a single residual evaluation is finally obtained by dividing the previous result by n_{res}.

The value of t_{ref} is determined in the same manner, using real-valued floating-point numbers instead of the dual numbers. Note, that the method used to determine both t_{ref} and Δt_{res} means that both are actually stochastic variables. Therefore, the mean — or expected value — of $\bar{t}_{\text{jac}}$ is not necessarily strictly proportional to the quotient of the mean of t_{ref} and Δt_{res}. To get a more accurate estimate for the expected value of $\bar{t}_{\text{jac}}$, a Taylor series expansion of the expected value can be used [32], for which the second-order series expansion reads

$$\mathrm{E}\left[\bar{t}_{\text{jac}}\right] \approx \left\lceil \frac{\mathrm{n}_{\text{der}}}{\mathrm{n}_{\text{d}}} \right\rceil \left(\frac{\mathrm{E}\left[\Delta t_{\text{res}}\right]}{\mathrm{E}\left[t_{\text{ref}}\right]} - \frac{\mathrm{Cov}\left[\Delta t_{\text{res}}, t_{\text{ref}}\right]}{\mathrm{E}^2\left[t_{\text{ref}}\right]} + \frac{\mathrm{Var}\left[t_{\text{ref}}\right]\mathrm{E}\left[\Delta t_{\text{res}}\right]}{\mathrm{E}^3\left[t_{\text{ref}}\right]} \right), \tag{D.3}$$

where $\mathrm{E}[\cdot]$ denotes the expected value and $\mathrm{Var}[\cdot]$ the variance of a stochastic variable. Furthermore, $\mathrm{Cov}[\cdot,\cdot]$ denotes the covariance of two stochastic variables. Therefore, figure 6.1 on page 124 shows $\mathrm{E}\left[\bar{t}_{\text{jac}}\right]$ computed according to equation (D.3) instead of the result obtained by evaluating equation (D.2). The error-bars depicted in the graph

Table D.1: Sum of addition and multiplication floating-point operations that are required to carry out the mathematical operations listed. Variables μ_{add} and μ_{mul} denote the average value for the number of addition and multiplication operations, respectively. The corresponding variance of both quantities are denoted by σ^2_{add} and σ^2_{mul}. The results have been obtained using 1 million pseudo-random input values.

mathematical operation	μ_{add}	μ_{mul}	σ^2_{add}	σ^2_{mul}
x/y	9	9.955	0	$4.391 \cdot 10^{-2}$
$\sqrt{x}$	14.030	13.016	$4.442 \cdot 10^{-1}$	$1.263 \cdot 10^{-1}$
e^x	17	13.297	0	$9.257 \cdot 10^{-1}$
x^y	54.282	29.550	$2.631 \cdot 10^{-1}$	1.754

represent the standard deviation, which has been computed by taking the square root of the variance. The variance is also computed by means of a second-order Taylor series expansion, which reads

$$\mathrm{Var}\left[\bar{t}_{\mathsf{jac}}\right] \approx \left\lceil \frac{\mathrm{n}_{\mathsf{der}}}{\mathrm{n}_{\mathsf{d}}} \right\rceil^2 \frac{\mathrm{E}^2\left[\Delta t_{\mathsf{res}}\right]}{\mathrm{E}^2\left[t_{\mathsf{ref}}\right]} \left(\frac{\mathrm{Var}\left[\Delta t_{\mathsf{res}}\right]}{\mathrm{E}^2\left[\Delta t_{\mathsf{res}}\right]} - 2\frac{\mathrm{Cov}\left[\Delta t_{\mathsf{res}}, t_{\mathsf{ref}}\right]}{\mathrm{E}\left[\Delta t_{\mathsf{res}}\right]\mathrm{E}\left[t_{\mathsf{ref}}\right]} + \frac{\mathrm{Var}\left[t_{\mathsf{ref}}\right]}{\mathrm{E}^2\left[t_{\mathsf{ref}}\right]} \right). \tag{D.4}$$

Number of operations required

To determine an estimate for the number of operations that is required to fully populate the Jacobian matrix, for a dual number with a certain dimension of the non-real part, two assumptions have been made:

(i) all mathematical operations for floating-point numbers can be represented by a combination of floating-point addition and floating-point multiplication operations;

(ii) performing a floating-point multiplication is equally expensive as a floating-point addition, i.e. takes the same number of clock-cycles to be performed.

The first assumption is a reasonable assumption, because some processors have a floating-point unit — which takes care of handling the floating-point operations, in a modern processor core — with a hardware implementation for only the most trivial mathematical operations. Therefore, in such a situation, the result of a mathematical function not implemented in hardware must be obtained in an alternative way. For this purpose, mathematical libraries exist — which usually are implemented together with the operating system. The mathematical functions in the library are implemented as a combination of additions, multiplications and bit shifts.

The second assumption is based on numerical experiments, which indicated this one to one ratio. Moreover, the hardware documentation [88] of a leading microprocessor vendor shows that the latency and throughput for floating-point additions and multiplications do not differ much for modern hardware.

Using these assumptions, an estimate for the number of operations can be obtained. For this purpose, the mathematical operations that are used for the computation of the flow residual have been represented as a combination of additions and multiplications. All operations involved are listed in table D.1, together with the average number of additions and multiplications required to compute the result. These results have been computed using pseudo-random input variables. For variables that allow for using negative values, the input ranges from -1 to 1. Input variables that must be positive, range from 0 to 1. The results presented have been obtained using 1 million pseudo-random input values.

Subsequently, the mathematical operations that have been represented are used to replace the original mathematical operations in the computation of the residual of the flow equations. Then, the total number of addition and multiplication operations required for the evaluation of the residual of the flow equations can be determined. This task can also be performed using dual numbers, instead of real-valued floating-point numbers. The non-real part of the flow solution is initialized to 1.0, for a number of pseudo-randomly selected flow variables equal to the dimension of the non-real part. The total number of operations that is required to fully populate the Jacobian matrix is then computed using

$$\mathrm{n}_{\mathsf{fp}} := (\mathrm{n}_{\mathsf{add}} + \mathrm{n}_{\mathsf{mul}}) \left\lceil \frac{\mathrm{n}_{\mathsf{der}}}{\mathrm{n}_{\mathsf{d}}} \right\rceil , \tag{D.5}$$

where $\mathrm{n}_{\mathsf{add}}$ denotes the number of floating-point additions that were counted and $\mathrm{n}_{\mathsf{mul}}$ denotes the number of floating-point multiplications. Note, that the ceiling operation is again used to make sure that all entries of the Jacobian matrix can be computed. Taking the ratio of this result with the total number of operations required to compute the Jacobian matrix by means of a central-difference finite-difference approximation, provides a measure for the number of operations saved by using a dual number with a non-real part containing multiple entries.

ACKNOWLEDGEMENTS

> *"No one who achieves success does so without acknowledging the help of others. The wise and confident acknowledge this help with gratitude."*
>
> — ALFRED N. WHITEHEAD (1861 – 1947)

REACHING this part of the thesis either means that the reader has shown a great deal of perseverance by getting this far, or that reading the most interesting part — the main body of this work — was skipped. No matter which approach was taken in reaching this page, the reader must be interested in who — else than the person who's name is on the cover of this thesis — contributed to this work. Since the work presented in this dissertation could not have reached its current state without the unconditional help of a number of people.

First of all, I would like to express my sincere gratitude towards Edwin van der Weide. Apart from always being prepared to provide me with good advice, he also has been a great help in solving the many problems I encountered during the project. Moreover, I think I developed a great deal of object-oriented programming skills thanks to carefully examining the C++ source code he wrote. A profound thanks also goes to Harry Hoeijmakers, who tirelessly tried to keep me focused, during the short conversations we had on the start of his working day, on a nearly daily basis. I also thank him for providing inspiration and guidance during the course of the project and for accepting me to pursue a PhD degree.

The people who helped me proof-reading the manuscript of this thesis deserve my gratitude as well, their effort resulted in a significant reduction of the number of typographical errors, inconsistencies and lack of clarity in the text, the figures and the formulas. The people who were prepared to spend their spare time for improving my thesis are: Geert Campmans, Kay Jongsma, Tom Jongsma, Tineke Jongsma and especially Natalie Ameloot who was very thorough in performing this task. I am also grateful to Harry, Edwin and Arie Verhoeff for carefully reading of and providing comments on the manuscript of this thesis. Furthermore, Arie — who works for the Suzlon Blade Technology in Hengelo, the company that financially supported this project — deserves my thanks for the nice cooperation we had.

I am Kay and Tom also thankful for accepting a role as paranymphs, at my thesis defence. Moreover, Kay deserves the credits for designing the cover of this thesis, which would not have been as nice as it is now if I had tried to do it myself.

For those readers who skipped to this page to see if their name was mentioned here and who turn out to be greatly disappointed because it is not, those people I would also like to thank. Although, your contribution to this result apparently did not qualify you for being mentioned explicitly by name, or unfortunately just slipped my attention —

which would be a terrible shame on me — I really am grateful for your contribution too, whatever it is you considered being worth acknowledging.

I like to think that I could have finished this thesis successfully without the moral support of my family and all the other people around me. This idea of mine might, however, not be particularly true. I am therefore very grateful that I did not need to find out how working on my PhD research would have been without receiving these unprompted words of encouragement.

Finally, I would like to thank the students and PhD students that have been practising judo with me at Arashi for all those years. Of those people I would particularly like to thank Andries Aarden, Bram Dil, Stefan Hartman, Ward Hendriks, Wilco Tax and Wouter Klein Wolterink. Although, they did not strictly contribute to the final result itself, they did help keeping me motivated. Especially, in the occasions that things did not work out in the way I did suppose they would. Them allowing me to throw them around, proved to be a very effective way to get rid of all the accumulated frustrations.

Enschede, June 2014,

Sietse Jongsma

CURRICULUM VITAE

On the rainy Monday afternoon of the 15th of April 1985, Sietse Jongsma was born, as the first of a twin, in a hospital in Doetinchem — a city in *de Achterhoek*, close to the border with Germany. In less than a year after he was born, his parents moved to *Friesland*, where they both originally came from. Sietse grew up in the small village of Baard, where he received his primary education.

For receiving his secondary education, which he started in 1997, he had to cycle to Leeuwarden — the capital of *Friesland*. There he first went to Slauerhoff college — which was later renamed to Piter Jelles Haydenstraat — for three years. The final three years of his secondary education he did at Piter Jelles Montessori.

After finishing his secondary education in 2003, Sietse chose to continue his educational career at the University of Twente. With his habit of destroying things, doing a Bachelor in Mechanical Engineering was the obvious choice. As part of his Bachelor education Sietse did a minor in Applied Physics, studying the subjects of Linear analysis, Electrodynamics and Quantum mechanics.

After successfully obtaining his Bachelor degree in 2006, he stayed in Twente[19] and continued his university education by doing his Master in Mechanical Engineering. Because of his fascination for flow phenomena — which occur everywhere in everyday life — he chose to specialize in the subject of fluid dynamics in the Engineering Fluid Dynamics group of prof. dr. ir. Harry Hoeijmakers. As a part of his Master education he did an internship, from July to September of the year 2007, at the National Research Council in Ottawa, Canada, under the supervision of dr. Hongyi Xu. Upon returning to The Netherlands, Sietse started his Master graduation assignment on the implementation of a turbulence model in an unstructured finite volume method, that was being developed by ir. Hein de Vries as a part of his PhD research. Hein was also his daily supervisor for that project. After successfully completing his Master assignment at the end of August 2008, Sietse decided to stay in Twente, to pursue a PhD degree.

In September of that same year he started on the PhD project that was focussed on the development of a gradient-based optimization method for the aerodynamic design of wind turbine rotor blades. Dr. ir. Edwin van der Weide became his daily supervisor for this project. During the research project Sietse took a number of courses to further develop his skills, to obtain more in-depth knowledge and to broaden his horizon. The most notable courses were “Introduction to Optimization Methods and Tools for Multidisciplinary Design in Aeronautics and Turbomachinery” and “Uncertainty Quantification in Computational Fluid Dynamics” both at the Von Karman Institute for Fluid Dynamics.

---

[19] Twente is the name of the university but also the name of a region.

He also took a course on “Turbulence and Turbulence Modeling” at the J. M. Burgerscentrum. Furthermore, Sietse assisted in the education provided by the Engineering Fluid Dynamics group, helping students doing their classroom assignments for the Bachelor course on Fluid Dynamics and Heat Transfer. The result of his PhD research has been presented in this thesis.

STELLINGEN

behorend bij het proefschrift:

"ON A METHOD FOR SIMULATION-BASED WIND TURBINE BLADE DESIGN"

S. H. Jongsma

11 juli 2014

1. Newton iteratie is een effectieve methode voor het oplossen van niet-lineaire vergelijkingen, mits een goede beginschatting kan worden verkregen en mits de afgeleiden nauwkeurig en efficiënt kunnen worden berekend.

   (*hoofdstuk 6 en 7 van dit proefschrift*)

2. De ontwikkeling van een robuust algoritme voor het verkrijgen van de verbindingsinformatie voor *overset* rekenroosters is geen sinecure.

   (*hoofdstuk 4 van dit proefschrift*)

3. Valideren van een rekenmethode voor het numeriek simuleren van stromingen vereist ook dat uiterste zorg wordt besteed aan het overeen laten komen van de geometrie die wordt gebruikt in de numerieke simulatie met de geometrie die werd gebruikt in het overeenkomstige experiment.

   (*hoofdstuk 5 van dit proefschrift*)

4. Onjuist gebruik van het woord gewicht voor de natuurkundige grootheid massa is hardnekkig.

   (*Comptes Rendus de la 3^e CGPM (1901), 1901, 70*)

5. Een globaal optimum is een mooi wiskundig concept, maar in een ontwerpproces zou voor een ingenieur het vinden ervan geen primair doel moeten zijn.

   (*A. Jameson en J. C. Vassberg, AIAA Paper 2001-0538 (2001)*)

6. Vooruitgang in de numerieke stromingsleer is sterk gekoppeld aan ontwikkelingen op het gebied van de computer hardware.

   (*AGARD AR-303 (1994)*)

7. Het baseren van een optimalisatiemethode op het principe van de evolutie is — indien middelen en tijd beperkt zijn — geen goed idee voor het oplossen van een optimalisatievraagstuk van enige complexiteit.

8. De publieke weerzin tegen het toepassen van kernenergie voor het opwekken van elektrische energie lijkt in belangrijke mate gestoeld op gebrek aan kennis van technische ontwikkelingen op dit gebied en van de mogelijkheden van kernenergie.

9. Meer dan eens is een zogenaamde klantenservice niet gericht op het verlenen van een dienst aan de klant, zoals het woord impliceert, maar op het zo efficiënt mogelijk afschepen van de klant.

10. Het privégebruik van een tablet-PC is interessant voor bijvoorbeeld treinforensen en mensen die een ‘bankhangend’ bestaan leiden.