GPU-Accelerated High-Order Scale-Resolving Simulations using the Flux Reconstruction Approach

by

George Ntemos

This thesis is submitted for the degree of
Doctor of Philosophy of Imperial College London

Department of Aeronautics
Imperial College London
London SW7 2AZ
United Kingdom

March 2018

Abstract

High-order methods in Computational Fluid Dynamics (CFD) offer a potential route towards the resolution of hitherto intractable fluid-dynamics problems in industry. The Flux Reconstruction (FR) approach provides a unifying framework for a number of popular high-order methods such as the Discontinuous Galerkin (DG). Its suitability for use on unstructured grids along with its ability to facilitate massively parallelised implementation on architectures such as GPUs provide a means to tackle computationally challenging flows around complex geometries. Such a flow can be found in the rod-aerofoil tandem configuration: Complex, unsteady flow structures generated by and interacting with more than a single solid body are central to a number of applications in the aerospace industry.

The current thesis attempts to demonstrate the suitability of the FR approach in successfully simulating the flow around a rod-aerofoil configuration. The in-house CFD solver employed in the research is presented and the FR implementation analysed. Computational grid resolution issues arising from the rod-aerofoil problem are studied and a novel strategy for the stabilisation of the computation is implemented in the form of local entropy stability. The results obtained are analysed and conclusions are drawn on the utility of the FR approach in the absence of a sub-grid scale model (Implicit LES - under-resolved DNS).

The present work confirms the utility of local entropy stability for the stabilisation of the rod-aerofoil simulation of aerofoil-chord based Reynolds number of $Re_a = 480,000$. It will also demonstrate that the under-resolved DNS setup that resulted in a computational cost of approximately six hours for a single flow pass over the aerofoil chord on 200 Nvidia P100 GPUs resulted in moderate success for a significant portion of the flow dynamics, which not adequately predicted when compared with experiment. The latter led to a series of useful conclusions. The core

of the conclusions involved the apparent over-prediction of time-averaged velocity and momentum deficits across wakes and as well as over-prediction of turbulent intensities. An identification of the problematic areas is therefore given and potential alleviation techniques outlined.

Acknowledgements

Here I would like to, first and foremost, thank my primary supervisor Dr Peter Vincent for offering me the opportunity to undertake my PhD in his group. His skill, fortitude and wisdom have guided me through a challenging but rewarding four years. I would also like to thank my co-supervisor Prof Michael Leschziner whose vast experience proved invaluable for the completion of this thesis. I thank my research group comprised of very bright and capable individuals and especially my office mate Dr Arvind Iyer whose great understanding of all aspects of our research has gotten me out of tricky situations countless times. I would like to end by thanking my parents and sister for their unconditional support throughout my studies; I would not have made it without them.

Declaration of Originality

I certify that the work presented in this thesis has been carried out by the author at the Aeronautics Department of Imperial College London and has not been submitted to any other institution. Furthermore, all work not being part of the author's research has been properly acknowledged.

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Nomenclature

ϵ	Limiting parameter of an element in the entropy-stability implementation.
γ	Ratio of the specific heats of the fluid.
Γ_n	One-dimensional mapping from reference to physical space.
$\hat{\Omega}_n$	n^{th} element of domain in reference space.
$\hat{\mathbf{F}}$	Transformed flux in multi-dimensional, advection-diffusion equation.
$\hat{\mathbf{U}}$	Transformed conserved variable in multi-dimensional, advection-diffusion equation.
$\hat{f}^{\delta}$	Transformed flux.
$\hat{u}^{\delta}$	Transformed solution.
λ	Characteristic velocity of an element in the entropy-stability implementation.
Ω	Domain of 1D advection equation.
Ω_n	n^{th} element of domain of 1D advection equation.
ϕ_i	Energy associated with Legendre mode L_i.
τ	Limiting parameter of a solution point in the entropy-stability implementation.
$\mathbf{F}$	Flux in multi-dimensional, advection equation.
$\mathbf{f}$	Flux tensor for the two-dimensional, compressible Navier-Stokes.
$\mathbf{g}(x)$	Solution correction function for advection-diffusion type problem.
$\mathbf{h}(x)$	Flux correction function for advection-diffusion type problem.
$\mathbf{J}$	Jacobian of spatial transformation matrix.

$\mathbf{J}$	Jacobian of the transformation matrix in multiple dimensions.
$\mathbf{U}$	Conserved variable in multi-dimensional, advection equation.
$\mathbf{u}$	Conserved variables vector for the two-dimensional, compressible Navier-Stokes.
$\mathbf{u}^I$	Common interface solution of the LDG approach.
C	Chord length of the aerofoil.
c	Local speed of sound.
C_d	Drag coefficient of the cylinder.
C_p	Pressure coefficient of the cylinder. Time averaged field.
f	Generic flux of u.
f_n^δ	Numerical flux inside element Ω_n.
$g(x)$	Flux correction function for the FR procedure.
L_e	Characteristic length of an element in the entropy-stability implementation.
L_i	i^{th} Legendre polynomial.
l_i	Lagrange polynomial basis.
q_p	Number of Gauss-Legendre quadrature points in 1D.
r	Standard space spatial coordinate.
R_∞	In-coming Riemann characteristic.
r_c	Radius of the cylinder.
R_i	Out-going Riemann characteristic.
s	Thermodynamic entropy.
t	Generic time coordinate.
t_{nd}	Non-dimensional time in the rod-aerofoil simulation.
u	Generic scalar solution.

u_n^δ Numerical solution inside element Ω_n.

U_∞ Free-stream velocity of the rod-aerofoil simulation.

u_{nd} Non-dimensional velocity in the rod-aerofoil simulation.

V_e Physical volume of an element in the entropy-stability implementation.

x Generic spatial coordinate.

Contents

Abstract **1**

Acknowledgements **3**

Declaration of Originality **4**

Copyright Declaration **5**

Nomenclature **6**

Table of Contents **9**

List of Figures **12**

List of Tables **23**

1 Introduction **24**

1.1 Flow Simulation and the Aerospace Industry 24

1.2 Methods in CFD 26

1.3 Research Objectives 28

1.3.1 Introduction to the Problem 28

1.3.2 Thesis Outline 29

2 Flux Reconstruction **31**

2.1 One-Dimensional Analysis 31

2.1.1 Reconstructing the Flux 35

2.2 Extension to Multiple Dimensions 38

2.2.1 Treatment of Viscous Flux 43

3 Computational Implementation 47

3.1 Introduction . . . 47

3.2 Implementation . . . 48

3.2.1 Functionality . . . 48

3.2.2 Main Code Blocks . . . 50

3.3 Boundary Conditions . . . 52

3.3.1 Solid Boundaries . . . 52

3.3.2 Far-Field Boundary . . . 54

3.4 Summary . . . 60

4 The Rod-Aerofoil Test Case 61

4.1 Geometry . . . 61

4.2 Previous Studies . . . 63

4.2.1 Experimental Study . . . 63

4.2.2 CFD Studies . . . 64

4.2.3 Summary . . . 67

4.3 Introduction to the Computational Grid . . . 69

5 Computational Grid and Resolution 72

5.1 Two-Dimensional Cylinder Studies . . . 72

5.2 Three-Dimensional Studies . . . 82

5.2.1 Rod-Aerofoil . . . 82

5.2.2 Effect of Spanwise Resolution . . . 90

5.2.3 The Three-Dimensional Rod-Aerofoil Domain . . . 94

6 Numerical Stability 97

6.1 Aliasing-Induced Instabilities . . . 97

6.2 Entropy Stability . . . 103

6.2.1 Validation of the Implementation . . . 107

7 Simulation of The Rod-Aerofoil Configuration 112

7.1 Computational Setup . . . 112

7.2 Results and Discussion . . . 115

7.2.1 Overall Flow Features . . . 115

7.2.2 Flow Statistics 122
7.3 Conclusions 145

8 Conclusions And Future Research **148**
8.1 Conclusions 148
8.2 Future Research 151

Bibliography **153**

Appendix A **160**

Appendix B **161**

List of Figures

1.1.1 Depiction of the current CFD capability for aircraft aerodynamics. Estimation by Airbus, image reproduced from [6]. The vertical axis corresponds to the load factor and the horizontal axis corresponds to Equivalent Air Speed (EAS) at a given altitude. 25

2.1.1 Arbitrary 1D domain Ω of equation 2.1.1 with physical solution u and flux f curves at arbitrary time $t = t_0$. Ω is defined in the interval $[x_0, x_N]$. 32

2.1.2 Polynomial representation of piecewise discontinuous solution u^δ and piecewise continuous flux f^δ inside Ω. 32

2.1.3 Two adjacent elements in transformed space $\hat{\Omega}_s$, each with a set of five arbitrary solution points $s_i, i = 0, 1, ..., 4$ defined. The dashed lines define the spatial extent of each element which in transformed space r is $[-1.0, 1.0]$. In accordance with figure 2.1.2, subscript $(_n)$ is defined as $n = 1, 2, ..., m-1, m, ..., N$ and refers to element $\hat{\Omega}_n$. For each element, the blue curve is the element-wise solution $\hat{u}_n^\delta$ in polynomial representation (equation 2.1.7), computed via the known values $\hat{u}_{n,i}^\delta(s_i)$ at the solution points s_i, depicted here as the intersections of the dotted lines with the blue curve. The red curve is the element-wise flux $\hat{f}_n^{\delta D}$ in polynomial representation (equation 2.1.9), computed via the known values $\hat{f}_{n,i}^{\delta D}(s_i) = \hat{f}_{n,i}^{\delta D}(\hat{u}_{n,i}^\delta(s_i))$. At this stage of the FR procedure and as the horizontal dotted line denotes, the flux is piecewise discontinuous between the two elements $\hat{\Omega}_{m-1}$ and $\hat{\Omega}_m$. 35

2.1.4 Elements Ω_{m-1} and Ω_m following the flux reconstruction procedure. The dotted red lines represent the previous states of the discontinuous fluxes $\hat{f}^{\delta D}_{m-1}$ and $\hat{f}^{\delta D}_{m}$ of figure 2.1.3. The horizontal dotted line depicts the common interface flux as a result of the correction procedure. 38

2.2.1 Arbitrary quadrilateral element Ω_m in physical space (x, y). Hollow circles represent solution points and solid circles represent flux points. 39

2.2.2 Element of figure 2.2.1 transformed into reference space (r, s), $\hat{\Omega}_m$. The (r, s) coordinates of the corners of the square are marked. . . . 40

3.3.1 Imposition of wall boundary conditions via "ghost" flux points. Solid circles represent the flux points of an element interface adjacent to the boundary. 'Ghost' points are represented with hollow circles. A Riemann-solve is performed between the two sets of points in order for the boundary condition to be enforced. 53

3.3.2 PyFR pointwise kernel for the Riemann-Invariant characteristic BCs. Lines 12 to 31 compute the relevant quantities defined in equations 3.3.14 to 3.3.22. Lines 33 to 40 then impose the boundary condition. 58

3.3.3 Acoustic waves generated and reflected by the domain boundaries at the start of the simulation, coloured by pressure magnitude p. Two-dimensional simulation of Rod-Aerofoil configuration with cold start (initial conditions equal to the freestream) and boundary conditions imposed as per section 3.3.2.1. Reynolds number $Re = 480,000$, Mach number $M = 0.21$. PyFR version 0.1.0 [23]. Far-field boundary conditions: $[rho_{fs} \quad u_{fs} \quad v_{fs}]$ imposed at the inlet (left-hand side edge, upper and lower edges), $[p_{fs}]$ imposed at the outlet (right-hand side edge). Acoustic wave reflection from the domain boundary is evident. 59

3.3.4 Physically realistic acoustic waves generated at the start the simulation, coloured by pressure magnitude p. Two-dimensional simulation of Rod-Aerofoil configuration with cold start (initial conditions equal to the freestream) and boundary conditions imposed as per section 3.3.2.2. Reynolds number $Re = 480,000$, Mach number $M = 0.21$. PyFR version 0.1.0 [23]. Colour map rendered in Paraview [32]. Riemann-Invariant boundary conditions imposed on all edges of the domain. Colouring scale of figure 3.3.3, acoustic wave reflection is dampened. 59

4.1.1 Cross-section of the rod-aerofoil configuration. U_∞ denotes the freestream, incoming velocity. 62

4.3.1 Cross-section of a computational grid across the $x - y$ plane (see figure 4.1.1). This grid is comprised of approximately $120,000$ quadrilateral elements on the $x - y$ plane. Meshing was done via Gmsh [44]. 69

4.3.2 Magnified version of the mesh around the aerofoil revealing the structured boundary layer approach. 70

4.3.3 Magnified view around the aerofoil leading edge. 71

5.1.1 An example that depicts a two-dimensional mesh around the cylinder with $\sim 12,000$. quadrilaterals. 73

5.1.2 Structured boundary layer region of figure 5.1.1. This particular example features a division of 100 elements in the circumferential direction and 21 elements in the radial direction. Expansion ratio along the radial direction was 1.05. This example yielded unsuccessful simulations (blow-ups). All meshes that resulted in successful computations are illustrated later and the reader can appreciate the visual differences in resolution. 74

5.1.3 Flow around 2D cylinder of figure 5.1.3, boundary layer region with 300 circumferential and 51 radial divisions. Instantaneous vorticity magnitude at $t_{nd} = 16.0$, with the simulation having started at $t_{nd} = 0.0$ (equation 4.1.1). Strouhal number $St = 0.199$ 76

5.1.4 Flow around 2D cylinder, boundary layer region with 300 circumferential and 51 radial divisions. Zoom-in around the boundary layer region with the computational grid superimposed. Instantaneous vorticity magnitude at $t_{nd} = 16.0$, with the simulation having started at $t_{nd} = 0.0$ (equation 4.1.1). Strouhal number $St = 0.199$. 76

5.1.5 Position of point P_1. The centre of the cylinder is at $0, 0$ of the x, y plane and d denotes the cylinder diameter. 77

5.1.6 Boundary layer regions with 61 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 158,000$ elements, 300 circumferential subdivisions and $\sim 113,000$ elements, 280 circumferential subdivisions and $\sim 98,000$ elements, 260 circumferential subdivisions and $\sim 85,000$ elements. 77

5.1.7 Boundary layer regions with 51 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 150,000$ elements, 300 circumferential subdivisions and $\sim 100,000$ elements, 280 circumferential subdivisions and $\sim 90,000$ elements, 260 circumferential subdivisions and $\sim 80,000$ elements. 78

5.1.8 Boundary layer regions with 41 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 145,000$ elements, 300 circumferential subdivisions and $\sim 95,000$ elements, 280 circumferential subdivisions and $\sim 86,000$ elements, 260 circumferential subdivisions and $\sim 77,000$ elements. 79

5.1.9 Boundary layer regions with 31 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 141,000$ elements, 300 circumferential subdivisions and $\sim 93,000$ elements, 280 circumferential subdivisions and $\sim 83,000$ elements, 260 circumferential subdivisions and $\sim 75,000$ elements. 80

5.1.10 Power spectral density (PSD) of velocity u (S_{uu}) at point P_1. Grid with boundary layer region of 300 circumferential and 51 radial elements. Strouhal number $St = 0.199$. 81

5.2.1 Computational grid of the preliminary Rod-Aerofoil simulation, $x-y$ plane. The number of hexahedral elements was approximately $270,000$. 83
5.2.2 Zoom-in of figure 5.2.1 around the Rod-Aerofoil area. 83
5.2.3 Zoom-in of figure 5.2.1 around the aerofoil leading edge. 84
5.2.4 Three-dimensional depiction of the grid of figure 5.2.1. The sliced-span approach of the meshing is evident across the circular boundary. . . 84
5.2.5 A $z-y$ cross-section of the mesh of figure 5.2.1 demonstrating the slices of the spanwise direction. Here, spanwise axis z is normal to the $x-y$ plane of figure 4.1.1 and is parallel to the plane of the page pointing downwards. 85
5.2.6 Instantaneous vorticity magnitude of the preliminary Rod-Aerofoil simulation. If at the start of the simulation we define non-dimensional time as $t_{nd} = 0.0$, the image corresponds to $t_{nd} = 16.5$. 86
5.2.7 Definition of data extraction regions. Point P_1 has the same cylinder-relative position in the $x-y$ plane as in figure 5.1.5. Its spanwise location is the middle of the domain. Line L_1 is parallel to the y axis and its distance from the aerofoil leading edge is $0.255C$. Rg_1, Rg_2 and Rg_3 denote regions of differing three-dimensional aspect ratios of hexahedral elements. 87
5.2.8 Convergence of three averaging windows. Window 1 corresponds to averaged u ($\bar{u}$), from $t_{nd} = 12.0$ to $t_{nd} = 16.0$. Window 2 corresponds to $t_{nd} = 14.0$ to $t_{nd} = 18.0$. Window 3 corresponds to $t_{nd} = 16.0$ to $t_{nd} = 20.0$. 88
5.2.9 Time-averaged $x-$*component* of velocity u ($\bar{u}$) along line L_1 of figure 5.2.7. Averaging period equal to 25.0 time units ($t_{nd} = 20.0$ to $t_{nd} = 45.0$). Comparison with experiment. 89
5.2.10 Power spectral density (PSD) of velocity u (S_{uu}) of preliminary Rod-Aerofoil simulation at point P_1 of figure 5.2.7. 90
5.2.11 The $x-y$ plane of the computational grid of the 3D cylinder spanwise resolution study. 91
5.2.12 Zoom-in around the cylinder boundary layer of figure 5.2.10. 92

5.2.13 Instantaneous vorticity magnitude of the three-dimensional cylinder simulation at $t_{nd} = 20.0$. 93
5.2.14 Power spectral density (PSD) of velocity u (S_{uu}) of three-dimensional cylinder simulation at point P_1 of figure 5.1.5. Data analysed from $t_{nd} = 20.0$ to $t_{nd} = 45.0$ 93
5.2.15 Computational grid of the large-scale Rod-Aerofoil simulation of Chapter 7, $x - y$ plane. The number of hexahedral elements was approximately $435,000$. 94
5.2.16 Zoom-in around the Rod-Aerofoil area of figure 5.2.15. 95
5.2.17 Zoom-in around the aerofoil leading edge of figure 5.2.15. 95
5.2.18 A $z - y$ cross-section of the mesh of figure 5.2.15 demonstrating the slices and thickness of the spanwise direction. Here, axis z is normal to the $x - y$ plane of figure 4.1.1. 96

6.1.1 Arbitrary one-dimensional element in physical space x, $[0, x_0]$, with solution u^δ and flux f^δ marked. The element contains five solution points s_n, $n = 0, 1, ..., 4$. 97
6.1.2 Instantaneous vorticity magnitude of the preliminary Rod-Aerofoil simulation. Snapshot at non-dimensional time $t_{nd} \approx 17.335$, approximately 0.002 time-units prior to simulation blowing-up. Circled is the vortical region that resulted in instability. Flow quantities such as velocity are at this stage rapidly growing towards infinity ($NaNs$). A desired consequence of using anti-aliasing would be the alleviation of the aforementioned phenomenon inside the circled region. 100
6.1.3 Instantaneous vorticity magnitude of the preliminary Rod-Aerofoil simulation with use of anti-aliasing. Snapshot at non-dimensional time $t_{nd} \approx 17.335$, approximately identical to figure 6.1.2. Note the suppression of the intensity of structures on the same colour scale, implying no immediate onset of instability. 101

6.2.1 PyFR pointwise mako kernel for the computation of the permissible time-step for each iteration. Lines 13-21 compute the primitive variables needed for expression 6.2.5. Lines 22 and 23 compute the needed characteristics speeds. Quantity λ is then computed in line 24 before line 26 returns the time step. As mentioned in chapter 3, this kernel is platform-independent and the low-level code that implements it generated at runtime. 105

6.2.2 PyFR pointwise mako kernel for the computation of $\tau_{e,n}^k$ for each iteration and for each element. After suitable determination of the flow variables by lines 13-17, $\tau_{e,n}^k$ is returned by line 18. 106

6.2.3 PyFR pointwise mako kernel for the computation of ϵ for each iteration and for each element. This kernel accepts as input the output of the kernel depicted in 6.2.2, τ. Line 12 will compute the minimum of all τ as required by the procedure. Lines 18 to 22 will then compute the needed flow variables that will then result in ϵ through line 23. 107

6.2.4 Non-dimensional pressure distribution at $t_{nd} = 5.0$ of the two-dimensional supersonic test case. Artificial-viscosity limiter used. Steady state reached, with oblique shock waves evident. 108

6.2.5 Pressure distribution at $t_{nd} = 5.0$ of the two-dimensional supersonic test case. Entropy limiter used. Steady state reached, with oblique shock waves evident. 109

6.2.6 Instantaneous vorticity magnitude of the large-scale simulation of Chapter 7. Note the highly vortical structures that impinge into the aerofoil leading edge, often resulting in instabilities that anti-aliasing of quadrature degree $Q_d = 11$ is not able to alleviate. 110

6.2.7 Distribution of quantity τ of the entropy limiter of instantaneous condition depicted in figure 6.2.6. 110

7.1.1 Computational grid of the large-scale Rod-Aerofoil simulation, $x-y$ plane. The number of tetrahedral elements (equal to the number of hexahedral elements per spanwise slice) is approximately $22,000$. . . 113

7.1.2 Strong scaling of PyFR on Piz Daint for the Rod-Aerofoil large-scale simulation. The speed-up is normalised by the performance on 16 GPUs, i.e. with wall-clock time equal to 1.0 for a constant number of iterations on 16 GPUs, the wall-clock time on 32 GPUs was reduced by a factor of ~ 1.84 (speed-up of ~ 1.84) for the same problem size and number of iterations. The red line was obtained via tests on 16, 32, 64, 128 and 256 GPUs. The time step size was kept constant at $\delta t = 2 \times 10^{-6}$. 114

7.2.1 Instantaneous Q-criterion iso-surfaces with magnitude of 1.0. The latter, non-dimensional magnitude was chosen so as to allow the visualisation of flow structures of length scales comparable to the scales of the solid bodies. Colouring via velocity magnitude. Simulation time $t_{nd} = 100.0$. 116

7.2.2 Instantaneous vorticity magnitude on plane normal to the spanwise axis and located in the middle of the spanwise thickness ($z = 0.25C$). Colour scale 1.0 to 100.0 non-dimensional units of vorticity (as velocity and length scales are non-dimensionalised via U_∞ and C.). Simulation time $t_{nd} = 100.0$. 116

7.2.3 Zoom-in of image 7.2.3 around the cylinder boundary layer and separation point. 117

7.2.4 Vortices generated in the wake of the cylinder. Instantaneous vorticity magnitude, colour scale 1.0 to 100.0 non-dimensional units of vorticity. Simulation time $t_{nd} = 100.0$. 118

7.2.5 A vortex impinging on the aerofoil leading edge. Instantaneous vorticity magnitude, colour scale 1.0 to 100.0 non-dimensional units of vorticity. Left snapshot at simulation time $t_{nd} = 100.0$. Right snapshot at simulation time $t_{nd} = 100.165$. 118

7.2.6 Time history of time step size δt using the adaptive controller of PyFR. The 500 iterations of the plot cover a non-dimensional simulation time of $t_{nd} \approx 0.0019$. 119

7.2.7 Instantaneous vorticity magnitude prior to local instability at $t_{nd} \approx 41.3$. Colour scale 1.0 to 300.0 non-dimensional units of vorticity. The circle denotes the problematic area vorticity magnitude above the 300.0 scale. 120

7.2.8 Entropy stability limiter parameter τ (defined in section 6.2) distribution and intensity in the vicinity of the circled area of figure 7.2.7. 121

7.2.9 Definition of data extraction regions. Points P_1, P_2, P_3 and P_4 have coordinates $[-0.87C, -0.05C]$, $[0.25C, -0.16C]$, $[0.25C, -0.30C]$ and $[0.25C, -0.90C]$ respectively. Their spanwise location is the middle of the domain. Lines L_1, L_2 and L_3 are parallel to the y axis and their x coordinates are $-0.255C$, $0.25C$ and $1.1C$ respectively. Their spanwise location is the middle of the domain. 122

7.2.10 Time-averaged streamwise velocity u profile predicted via simulation with PyFR, along line L_1 of figure 7.2.9. Velocity u_∞ is the freastream velocity. Averaging period $t_{nd} \approx 30.0$ to $t_{nd} \approx 100.0$. 123

7.2.11 Time-averaged streamwise velocity u profile along line L_1 of figure 7.2.9. Comparison with experiment. 124

7.2.12 Time-averaged streamwise velocity u profile along line L_1 of figure 7.2.9. Comparison with experiment and CFD. 125

7.2.13 Pressure coefficient C_p of the time-averaged field around the cylinder. Quantity d_c denotes the diameter of the cylinder. Coordinate x is the streamwise coordinate as per figure 7.2.9. $x = 0$ corresponds to the leading edge of the cylinder 126

7.2.14 Time-averaged streamwise velocity u profile along line L_1 of figure 7.2.9. Comparison with experiment and the LES work of Agrawal *et al.* [41]. 128

7.2.15 Pressure coefficient C_p of the time-averaged field around the cylinder. Quantity d_c denotes the diameter of the cylinder. Coordinate x is the streamwise coordinate as per figure 7.2.9. $x = 0$ corresponds to the leading edge of the cylinder 129

7.2.16 Profile of turbulence intensity $\sqrt{< uu >}$ along line L_1 of figure 7.2.9. Comparison with experiment and other CFD studies. 130

7.2.17 Time-averaged Reynolds stress $< uv >$ profile along line L_1 of figure 7.2.9 via simulation with PyFR. 131
7.2.18 Time-averaged streamwise velocity u profile along line L_2 of figure 7.2.9. Comparison with experiment and CFD. 132
7.2.19 Turbulence intensity profile $\sqrt{< uu >}$ along line L_2 of figure 7.2.9. Comparison with experiment and CFD. 133
7.2.20 Time-averaged Reynolds stress $< uv >$ profile along line L_2 of figure 7.2.9 via simulation with PyFR. 134
7.2.21 Time-averaged streamwise velocity u profile along line L_3 of figure 7.2.9. Comparison with experiment and CFD. 135
7.2.22 Turbulence intensity profile $\sqrt{< uu >}$ along line L_3 of figure 7.2.9. Comparison with experiment and CFD. 136
7.2.23 Time-averaged Reynolds stress $< uv >$ profile along line L_3 of figure 7.2.9 via simulation with PyFR. 137
7.2.24 Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9 via simulation with PyFR. 138
7.2.25 Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9 via simulation with PyFR. Estimation via Welch method of order 64. 139
7.2.26 Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9 via simulation with PyFR. Comparison of one-point spectrum with average spectrum of nine equidistant points distributed along the spanwise direction. 140
7.2.27 Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9. Comparison with experiment and other CFD approaches. 141
7.2.28 Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9. Comparison with experiment and the compressible LES Charles code of Agrawal *et al.* [41]. 141
7.2.29 Power spectral density S_{uu} of velocity u at point P_2 of figure 7.2.9. Comparison with experiment and CFD. 142
7.2.30 Power spectral density S_{uu} of velocity u at point P_2 of figure 7.2.9. Comparison with experiment and the compressible LES Charles code of Agrawal *et al.* [41]. 143

7.2.31 Power spectral density S_{uu} of velocity u at point P_3 of figure 7.2.9 via simulation with PyFR. 144

7.2.32 Power spectral density S_{uu} of velocity u at point P_4 of figure 7.2.9 via simulation with PyFR. 144

List of Tables

3.2.1 Tabulation of main functionality of PyFR, version 1.6.0. The reader can also refer to [23]. 50

4.2.1 Tabulation of main CFD characteristics of previous studies of the rod-aerofoil problem. Predicted Strouhal numbers are included. . . . 68

5.1.1 Strouhal number prediction as a function of boundary layer resolution. 81

Chapter 1

Introduction

1.1 Flow Simulation and the Aerospace Industry

One of the primary factors that influence aerospace vehicle design and performance is the ability to predict and assess fluid flows with sufficient accuracy. Whilst experiments have proved an effective tool towards achieving this, and are still in extensive use, Computational Fluid Dynamics (CFD) was accurately predicted by Kuechemann in 1978 to produce an increasingly powerful impact on flow modelling and simulation in the aerospace industry [1]. CFD is currently in widespread use, steadily complementing, or even replacing, the wind tunnel in an increasing number of areas during the research, design and development process [3, 4].

In spite of the proven utility of CFD in industry, there remain a large portion of the aerospace sector which cannot effectively rely on simulation alone. As summarised by NASA in 2014, a number of problems and challenges that are fundamental to the aerospace industry remain open [5]. These challenges have so far prevented the attainment of a general predictive capability of CFD for the simulation of flow around the entire geometry of a modern aircraft under all flow conditions encountered inside the aircraft's flight envelope.

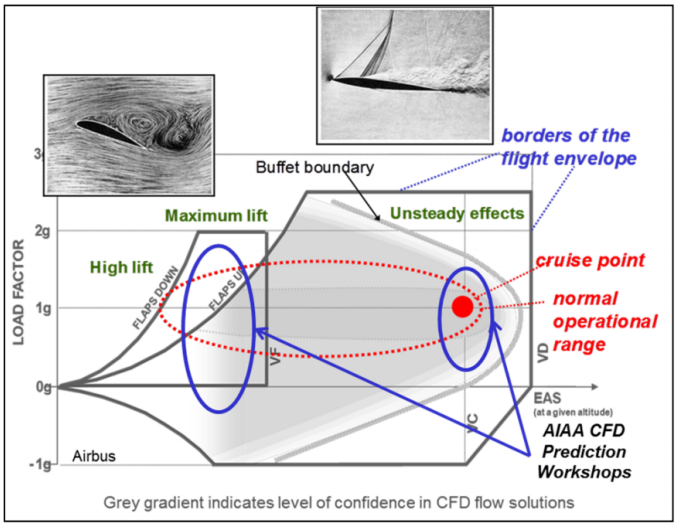


Figure 1.1.1: Depiction of the current CFD capability for aircraft aerodynamics. Estimation by Airbus, image reproduced from [6]. The vertical axis corresponds to the load factor and the horizontal axis corresponds to Equivalent Air Speed (EAS) at a given altitude.

Figure 1.1.1 summarises this challenge. The grey scale inside the aircraft's envelope corresponds to the current level of confidence aircraft manufacturers have in industrial CFD tools. One can clearly appreciate the lack of prediction ability even in areas of the envelope in which aircraft routinely operate. One of the primary reasons for this lack of confidence is that the current ability to accurately simulate highly unsteady turbulent flows, particularly when phenomena such as boundary layer transition and separation are present, is very limited for a large number of applications [5]. Furthermore, it is predicted that an increase in computational capability alone is not necessarily sufficient. An increase in accuracy would be facilitated by an increase in computational capability, however stability is not necessarily guaranteed [5]. Therefore, progress in CFD methods is also required if one is to achieve the capability that would allow simulations of sufficient quality for design. In addition to being an alternative to costly and time-consuming experiments, accurate simulations allow the prediction of phenomena that cannot be reproduced in experiment such as transient flows resulting from accelerating frames of reference, facilitating cost and time reduction for optimum design.

Problems that are considered hitherto intractable include the aerodynamic simulation of current and next-generation unmanned aerial vehicles (UAV), that are characterised

by highly unsteady and separated flows over complex geometries such as spoilers and other control surfaces. Such flows are in the Reynolds number range of $10^4 - 10^7$, and their efficient simulation requires the accurate solution of the compressible set of the Navier-Stokes equations for highly turbulent conditions whilst not requiring untenable computational resource. Therefore, there is strong motivation to develop and assess novel computational methods that offer extended capabilities for the most challenging conditions in aerospace design.

1.2 Methods in CFD

Numerical methods for CFD can be divided into major families depending upon the type of spatial discretisation used for the solution of the Navier-Stokes equations. The most widely used methods [7] are the finite difference method (FD), the finite volume method (FV) and the finite element method (FE). Broadly, the FD method discretises the differential form of the equations into stencils of points, FV discretises the computational domain into cells within which the integral form of the equations is solved and the FE method discretises the computational domain into elements along which the solution is represented by polynomials that satisfy the governing equations. Most current commercial CFD simulation codes are based on the FV method. These software packages include CD-adapco's (now Siemens) Star CCM+ and Star-CD [60] and Ansys' Fluent and CFX [61], [62]. The OpenFOAM [63] numerics platform has also been extensively used in performing FV simulations. An recent example of the utilisation of the OpenFOAM platform in the simulation of the 3D flow around around a complete aircraft is the work of Kosik *et al.* [64].

Turbulent flows and their chaotic nature present a major challenge in terms of the spatial and temporal scales that need to be simulated. CFD simulations have, historically, been dependant mainly upon statistical modelling of the turbulent motions, instead of directly solving the Navier-Stokes equations across all flow scales (direct numerical simulation - DNS) [4]. These methods include the Reynolds-averaged Navier-Stokes (RANS) approach and, to a lesser extent, large eddy simulation (LES) [4]. Both approaches rely upon explicit modelling of all or part of the turbulent motions in order to reduce the computational demand relative to the fully resolved

simulation of turbulent flow.

In addition to the type of spatial discretisation and the treatment of turbulent scales, the numerical order of the spatial scheme has become a further, *de facto*, criterion by which CFD codes are grouped. Industrial and commercial applications have predominately relied upon low-order FD and FV solvers performing RANS simulations, due to low computational cost and proven reliability over carefully defined flow regimes [5]. Here, low order is taken to mean that the spatial discretisation scheme offers a spatial order of accuracy of 2 or below. As stated in the previous section, these established, low-order approaches have not yet been able to render a large number of design-relevant simulations for the most challenging flow problems tractable. Whilst this is not necessarily a result of the low-order nature of the approach, *per se*, the scientific community understands that high-order schemes, potentially, offer a wider range of opportunity in the search of increased accuracy for a given computational cost [3], [6], [7]. Some of the first to develop and apply FE methods for CFD were Franca *et al.* in their work which developed a stabilised version of the FE method [65]. Stabilised FE methods are now used in commercially available CFD packages, such as the COMSOL platform [66].

It is often convenient to categorise high-order methods into those developed for structured grids and those developed for unstructured grids. Examples of structured implementations include the most popular forms of high-order FV methods, for example the essentially non-oscillatory schemes (ENO) of Harten *et al.* [8] and the weighted essentially non-oscillatory schemes (WENO) of Liu *et al.* [10]. In the FD family, of notable contribution is the work of Lele which resulted in a FV scheme with spectral-like resolution [9]. The aforementioned methods constitute some of the first robust applications of high-order spectral schemes suited to relatively simple geometries. Research on schemes designed for use on unstructured grids includes the contributions of Barth *et al.* [11], Bassi *et al.* [12] and Cockburn *et al.* [13]. However, the two predominantly used methods for unstructured grids are the discontinuous Galerkin (DG) and the Spectral Difference (SD) method. The former is, arguably, the most popular. The DG method was originally developed during the 1970s by Reed and Hill [14], and it utilises the piecewise-polynomial approach combined with a technique closely associated with the classic FV approach in that information

between elements is allowed to be "shared" via Riemann solvers across the element faces. The polynomial representation of the solution is such that it satisfies the variational form of the governing equations. Originally developed by Kopriva *et al.* [16] and Liu *et al.* [15], the SD method is a relatively new scheme and attempts to add aspects of the FD approach to the spectral FE framework.

The DG method is of particular importance for the present thesis, since, fundamentally, it is the scheme the is used in order to perform the research. The DG scheme can be recovered from a particular form of the flux reconstruction approach (FR) that will be presented in detail in chapter 2.

1.3 Research Objectives

1.3.1 Introduction to the Problem

As mentioned previously, there exist flow conditions that are of major importance to industry, whilst simulation of them of sufficient accuracy is an open challenge. These areas include:

- The accurate resolution of turbulent, multi-scale, highly unsteady flow.
- The stability of the simulation.
- The computational cost.

A successful, industry-relevant technology must be able to address all of the above. The present work in particular, attempts to alleviate stability problems that arise form the simulations at hand. These stability issues centre on the accumulation of excess energy in the resolved modes of the flow variables, a phenomenon which, for instance, a computationally realistic resolution of the computational grid and well-established techniques such as anti-aliasing cannot adequately address if one does not opt for computationally highly demanding quadrature degrees.

The broad objective of the present thesis is to explore the performance and utility of an in-house computational implementation of a high-order FE approach, PyFR [17], as a means towards addressing the above challenges. Of particular interest were the accuracy with which complex flow structures could be resolved without

requiring excessive computational resource and the stability of this approach. The problem of numerical stability is a fundamental issue of high-order FE schemes and, specifically, the FR approach [18]. In consequence, a significant portion of the present work focused on the exploration and implementation of strategies that allow the stabilisation of the simulation. The latter was achieved through a combination of algorithmic methods, an approach which, to the knowledge of the author, is not widely used.

The core of the present research involved the study of a particular test case that provided a means of evaluating the performance of the aforementioned approaches on industry-relevant problems. Such a problem can be found in the form of high Reynolds number (Re) flow that exhibits a turbulent wake from a blunt body impinging on a solid downstream. The Rod-Aerofoil test case was chosen as such a problem. The geometry has been studied experimentally by the work of Jacob *et al.* [19]. The experiment was undertaken in order to provide data aimed at testing of the performance of various CFD approaches. The flow dynamics that are present are similar to those in flows associated with industrial applications, such as aircraft flaps in the vicinity of the main wing and landing gear aerodynamics. Because of the existence of intense turbulence, a wide spectrum of small-scale phenomena are present in the flow and allow the evaluation of the time-resolving capabilities of a CFD simulation. A series of previous studies have been attempted and are summarised in chapter 4 of the present thesis. Studying the problem through a compressible Navier-Stokes solver using the (FR) high-order method on unstructured domains is a novel approach and therefore one that can potentially lead to observations that will facilitate further development.

1.3.2 Thesis Outline

The remainder of the thesis is organised as follows:

- Chapter 2: The mathematical framework behind the FR approach is presented in detail. The core of the method is described in detail via an one-dimensional analysis and its extension to multiple dimensions is outlined.

- Chapter 3: The computational implementation of the FR approach, PyFR, is

presented in detail. The blocks that constitute the compute-intensive operations of the code are described. The code characteristics and the computational platforms (for example, the NVidia Cuda platform) supported are outlined.

- Chapter 4: The Rod-Aerofoil test case is presented in detail and previous computational work reviewed. Part of the work presented in this chapter will form references against which the results of chapter 7 will be compared and analysed.

- Chapter 5: The strategy for the identification of a suitable range of meshes for the Rod-Aerofoil problem is presented. Simulations and results outlining the effect of the mesh type on the quality of prediction and computational cost are analysed. The two main factors studied influencing the performance of the mesh are boundary layer and span-wise resolution.

- Chapter 6: The treatment of stability issues and the solutions developed and employed are analysed. This chapter will introduce the concept of aliasing-induced instabilities and the strategy towards its alleviation. It will also present and analyse a novel approach into stabilising the rod-aerofoil simulation, namely the entropy stability scheme.

- Chapter 7: The configuration and results of the large-scale simulation of the rod-aerofoil problem are presented. An analysis of the results is made and conclusions are drawn on the utility of the methods and strategies employed.

- Chapter 9: Ultimate conclusions are drawn and the identification of areas for future work is made.

Chapter 2

Flux Reconstruction

The Flux Reconstruction (FR) approach was originally proposed by Huynh in 2007 [2]. One of the primary characteristics of the method is that it presents a unifying framework for popular high-order methods such as DG and SD [20]. It has also been primarily designed for implementation on unstructured grids.

2.1 One-Dimensional Analysis

The core of the method involves expressing the solution and flux of an advection-diffusion type problem in terms of a polynomial basis. The polynomials are defined from the known quantities on specified solution and flux points inside each element of a, generally unstructured and three-dimensional, computational grid.

The principles of the procedure can be demonstrated by considering the following one-dimensional (1D) advection problem.

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0. \tag{2.1.1}$$

In equation 2.1.1, if x and t are the spatial and temporal coordinates respectively, $u = u(x,t)$ is the scalar solution and $f = f(u)$ is the flux of solution u in direction x. A 1D domain inside which equation 2.1.1 holds can be defined.

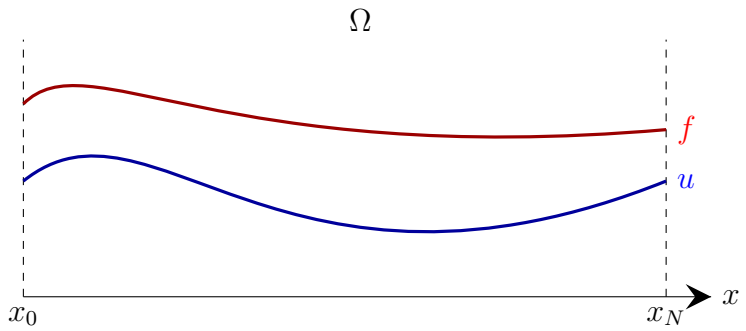


Figure 2.1.1: Arbitrary 1D domain Ω of equation 2.1.1 with physical solution u and flux f curves at arbitrary time $t = t_0$. Ω is defined in the interval $[x_0, x_N]$.

Figure 2.1.1 depicts the physical solution u and flux f of equation 2.1.1 at some arbitrary time $t = t_0$. In order for the problem to be solved numerically using the FR approach, we divide domain Ω into N 1D elements such that $\Omega_n = [x_{n-1}, x_n] \in \Omega$. Now let u_n^δ be the (generally discontinuous between elements) numerical solution and f_n^δ be the (continuous between elements) numerical flux inside Ω_n. The FR approach represents u_n^δ with a polynomial of degree k and f_n^δ with a polynomial of degree $k+1$. Although there is no mathematical requirement for the polynomial degree of the flux representation to be $k+1$ in order for the resultant discretised equations to be solved, the FR method assumes that the derivatives of the flux representation sits in the same polynomial space as the representation of the solution. Hence, in FR, the polynomial representation of the flux is of one degree higher than that of the solution.

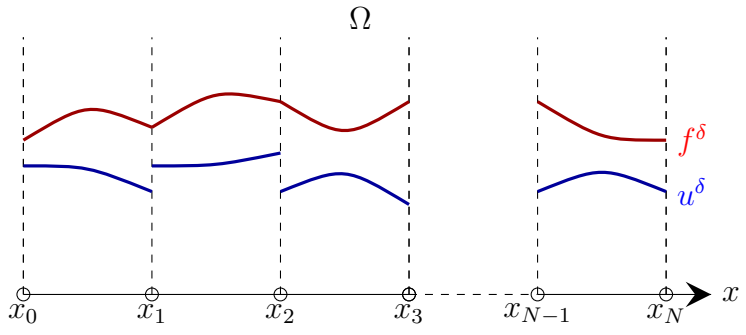


Figure 2.1.2: Polynomial representation of piecewise discontinuous solution u^δ and piecewise continuous flux f^δ inside Ω.

It is advantageous to transform each element Ω_n of the physical domain Ω into a space that is more convenient to work with. The reasons for this include the

fact that it facilitates computational implementation, it allows for clearer and easier mathematical description. It also allows for the matrix-matrix multiplications that arise in the implementation to be performed via the use of single-operator matrices. Transforming each reference element $\hat{\Omega}_n(r) = [-1, 1]$ into the physical element $\Omega_n(x) = [x_{n-1}, x_n]$ is done using the following mapping

$$\Gamma_n(x) = x(r) = \left(\frac{1-r}{2}\right) x_{n-1} + \left(\frac{1+r}{2}\right) x_n. \tag{2.1.2}$$

The inverse procedure Γ_n^{-1} would recover x.

$$\Gamma_n^{-1}(r) = r(x) = 2\left(\frac{x - x_{n-1}}{x_n - x_{n-1}}\right) - 1. \tag{2.1.3}$$

It is noted here that the linear transformation described by equation 2.1.2 is generally invertible if the physical and reference spaces are of equal dimensions.

Applying mapping 2.1.3 onto equation 2.1.1 will, for the n^{th} element, yield the following expression

$$\frac{\partial \hat{u}_n^\delta}{\partial t} + \frac{\partial \hat{f}_n^\delta}{\partial r} = 0. \tag{2.1.4}$$

In equation 2.1.4, $\hat{u}_n^\delta$ and $\hat{f}_n^\delta$ are, respectively, the transformed, element-wise solution and flux in the reference element $\hat{\Omega}_s$. As a result we can arrive at the following expression

$$\hat{u}_n^\delta(r, t) = u_n^\delta(\Gamma_n^{-1}(r), t), \tag{2.1.5}$$

which is a polynomial of degree k and

$$\hat{f}_n^\delta(r, t) = \frac{f_n^\delta(\Gamma_n^{-1}(r), t)}{J_n}, \tag{2.1.6}$$

which is a polynomial of degree k+1. J_n is the Jacobian of the transformation matrix which, in this linear case of constant entries, is $J_n = (x_n - x_{n-1})/2$. In the case of non-linear transformations or transformations in multiple spatial dimensions, the Jacobian is generally not constant.

Assuming a Lagrange polynomial basis l_i and an associated set of $k+1$ solution points inside element $\hat{\Omega}_n$, the following projection of the solution can be made

$$\hat{u}_n^\delta = \sum_{i=0}^{k} \hat{u}_{n,i}^\delta l_i, \tag{2.1.7}$$

with

$$l_i = \prod_{j=0, j\neq i}^{k} \left(\frac{r - s_j}{s_i - s_j} \right). \tag{2.1.8}$$

In equation 2.1.8, s_i is the location of the solution points in reference space along axis r. As already mentioned, the number of the solution points determines the order of the scheme whilst the type of distribution is subject to significant theoretical research, with popular examples including the Gauss-Legendre and the Gauss-Legendre-Lobatto types of points.

In equation 2.1.7, $\hat{u}_{n,i}^\delta$ is the value of the transformed solution at solution point i of element n. Flux $\hat{f}_n^\delta$ would now need to be projected in similar fashion, arising into $\hat{f}_n^\delta$

$$\hat{f}_n^{\delta D} = \sum_{i=0}^{k} \hat{f}_{n,i}^{\delta D} l_i. \tag{2.1.9}$$

In equation 2.1.9, $\hat{f}_{n,i}^{\delta D}$ is the transformed flux at solution point i, and l_i is the associated Lagrange polynomial.

As the superscript $(^D)$ denotes in $\hat{f}_n^{\delta D}$, equation 2.1.9 results in a flux representation which is generally piecewise discontinuous across elements. This is a result of the flux being initially computed via the known solution $\hat{u}_n^\delta$ of equation 2.1.7 (since $f = f(u)$) which is itself discontinuous as depicted in figure 2.1.3.

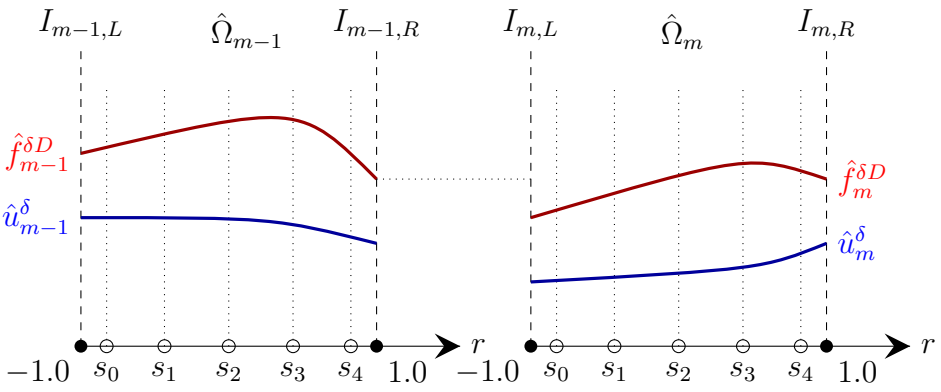


Figure 2.1.3: Two adjacent elements in transformed space $\hat{\Omega}_s$, each with a set of five arbitrary solution points $s_i, i = 0, 1, ..., 4$ defined. The dashed lines define the spatial extent of each element which in transformed space r is $[-1.0, 1.0]$. In accordance with figure 2.1.2, subscript $(_n)$ is defined as $n = 1, 2, ..., m-1, m, ..., N$ and refers to element $\hat{\Omega}_n$. For each element, the blue curve is the element-wise solution $\hat{u}^{\delta}_n$ in polynomial representation (equation 2.1.7), computed via the known values $\hat{u}^{\delta}_{n,i}(s_i)$ at the solution points s_i, depicted here as the intersections of the dotted lines with the blue curve. The red curve is the element-wise flux $\hat{f}^{\delta D}_n$ in polynomial representation (equation 2.1.9), computed via the known values $\hat{f}^{\delta D}_{n,i}(s_i) = \hat{f}^{\delta D}_{n,i}(\hat{u}^{\delta}_{n,i}(s_i))$. At this stage of the FR procedure and as the horizontal dotted line denotes, the flux is piecewise discontinuous between the two elements $\hat{\Omega}_{m-1}$ and $\hat{\Omega}_m$.

2.1.1 Reconstructing the Flux

Following the calculation of the discontinuous flux, in order for the scheme to be conservative the flux needs to become continuous across element interfaces, thus recovering the condition depicted in figure 2.1.2. In order for this to be achieved, one must first calculate the fluxes at either side of a reference element. In figure 2.1.3, the dashed lines $I_{m-1,L}$, $I_{m-1,R}$, $I_{m,L}$ and $I_{m,R}$ represent the left interface of element Ω_{m-1}, the right interface of element Ω_{m-1}, the left interface of element Ω_m and the right interface of element Ω_m respectively. One can also recognise that lines $I_{m-1,R}$ and $I_{m,L}$ represent (in physical space) the common interface of elements Ω_{m-1} and Ω_m. Without loss of generality, we will focus on element Ω_{m-1}. One can calculate the discontinuous flux for each of the two interfaces of the element via equations 2.1.7 and 2.1.9:

$$\hat{u}^{\delta}_{m-1,L} = \sum_{i=0}^{k} \hat{u}^{\delta}_{m-1,i} l_i(r_L) = \sum_{i=0}^{k} \hat{u}^{\delta}_{m-1,i} l_i(-1) \tag{2.1.10}$$

$$\hat{f}^{\delta D}_{m-1,L} = \sum_{i=0}^{k} \hat{f}^{\delta D}_{m-1,i} l_i(r_L) = \sum_{i=0}^{k} \hat{f}^{\delta D}_{m-1,i}(\hat{u}^{\delta}_{m-1,L}) l_i(-1) \tag{2.1.11}$$

$$\hat{u}^{\delta}_{m-1,R} = \sum_{i=0}^{k} \hat{u}^{\delta}_{m-1,i} l_i(r_R) = \sum_{i=0}^{k} \hat{u}^{\delta}_{m-1,i} l_i(1) \tag{2.1.12}$$

$$\hat{f}^{\delta D}_{m-1,R} = \sum_{i=0}^{k} \hat{f}^{\delta D}_{m-1,i} l_i(r_R) = \sum_{i=0}^{k} \hat{f}^{\delta D}_{m-1,i}(\hat{u}^{\delta}_{m,R}) l_i(1) \tag{2.1.13}$$

In equations 2.1.10 to 2.1.13, r_L and r_R are the transformed coordinates at the right and the left side of the element respectively with (r_L,r_R)=(-1,1). Also, in the particular example depicted in figure 2.1.3, $k = 4$. In the interests of simplicity, we define $\hat{f}^{\delta D}_{m-1,L} = \hat{f}^{\delta D}_{L}$ and $\hat{f}^{\delta D}_{m-1,R} = \hat{f}^{\delta D}_{R}$ with $\hat{f}^{\delta D}_{L}$ corresponding to $I_{m-1,L}$ and $\hat{f}^{\delta D}_{R}$ corresponding to $I_{m-1,R}$ on figure 2.1.3. Once the discontinuous fluxes at either side the element are calculated, one needs to calculate the contribution of the flux of two adjacent elements at either side of the domain. In figure 2.1.3, element Ω_{m-1} is sharing interface $I_{m-1,R}$ with Ω_m (depicted here) and interface $I_{m-1,L}$ with Ω_{m-2} (not depicted here). The contribution that elements Ω_m and Ω_{m-2} have on the flux of Ω_{m-1} can be calculated by means of a suitable method that is dependant on the nature of the governing equations being solved. In the case of the Euler system of equations, a widely used method is a Riemann-type solver [21]. The resultant flux of the aforementioned procedure on interface $I_{m-1,R} \equiv I_{m,L}$ is denoted as $\hat{f}^{\delta I}_{R}$ and the resultant flux on interface $I_{m-2,R} \equiv I_{m-1,L}$ is denoted as $\hat{f}^{\delta I}_{L}$.

The stage in the FR procedure that guaranties a common interface flux between the elements, involves a suitable correction of the resultant total flux along each boundary such that the flux on the boundary is equal to the corresponding flux of the adjacent element on the same location. Denoting the correction as $\hat{f}^{\delta C}$, the following expression can be written

$$\hat{f}^{\delta C} = (\hat{f}^{\delta I}_{L} - \hat{f}^{\delta D}_{L}) g_L + (\hat{f}^{\delta I}_{R} - \hat{f}^{\delta D}_{R}) g_R. \tag{2.1.14}$$

In equation 2.1.14, g_L and g_R are the correction functions that constitute the core of the flux reconstruction procedure. These functions are suitable polynomials

of degree $(k+1)$, such that the common interface flux condition is satisfied. In the particular example depicted in figure 2.1.3, the polynomial degree of g_L and g_R would be 5. In order for the interface fluxes to be common across elements the following conditions should hold

$$g_L(r_L) = g_L(-1) = 1, \tag{2.1.15}$$

$$g_R(r_L) = g_L(-1) = 0, \tag{2.1.16}$$

$$g_L(r_R) = g_L(1) = 0, \tag{2.1.17}$$

$$g_R(r_R) = g_L(1) = 1. \tag{2.1.18}$$

Also, symmetry demands that

$$g_L(r) = g_R(-r). \tag{2.1.19}$$

Following the correction procedure, the total, transformed flux of the element is given in the expression below

$$\hat{f}^{\delta} = \hat{f}^{\delta D} + \hat{f}^{\delta C}. \tag{2.1.20}$$

The result of equation 2.1.20 is graphically illustrated in figure 2.1.4.

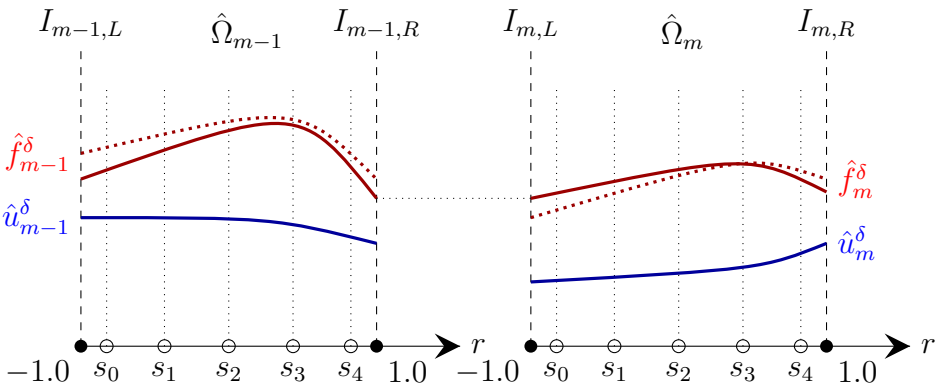


Figure 2.1.4: Elements Ω_{m-1} and Ω_m following the flux reconstruction procedure. The dotted red lines represent the previous states of the discontinuous fluxes $\hat{f}^{\delta D}_{m-1}$ and $\hat{f}^{\delta D}_{m}$ of figure 2.1.3. The horizontal dotted line depicts the common interface flux as a result of the correction procedure.

Having computed the expression of the total, corrected flux, one needs to compute its divergence. This is achieved by differentiating the total expression of the flux at each solution point

$$\frac{\partial \hat{f}^{\delta}}{\partial r}(r_i) = \sum_{i=0}^{k} \hat{f}^{\delta D}_{j} \frac{dl_j}{dr}(r_i) + (\hat{f}^{\delta I}_{L} - \hat{f}^{\delta D}_{L}) \frac{dg_L}{dr}(r_i) + (\hat{f}^{\delta I}_{R} - \hat{f}^{\delta D}_{R}) \frac{dg_R}{dr}(r_i). \tag{2.1.21}$$

Equation 2.1.22 can now be advanced in time via a suitable time-stepping method.

$$\frac{\partial \hat{u}^{\delta}_{i}}{\partial t} = -\frac{\partial \hat{f}^{\delta}}{\partial r}(r_i) \tag{2.1.22}$$

2.2 Extension to Multiple Dimensions

Having introduced the basic methodology of the FR approach for advection systems, this section will summarise its extension to multiple dimensions. The summary will be based on the work of [17], [18] and [22]. It is directly relevant to the computational implementation of Chapter 3.

We consider an arbitrary, quadrilateral element of a two-dimensional, unstructured computational grid. It is noted here that, all simulations presented in the current work utilised either quadrilateral in 2D or hexahedral in 3D elements. While is a variety of different element types such as triangles and tetrahedra have been widely used and studied in the past and are indeed supported by PyFR, the choise of

quadrilateral and hexahedral elements was made without an exhaustive suitability study. It is however true that limitations in their exclusive use exist, with an obvious example being the representation of more complex geometries than the rod-aerofoil configuration. The use of hybrid meshes would be more suitable in such a situation.

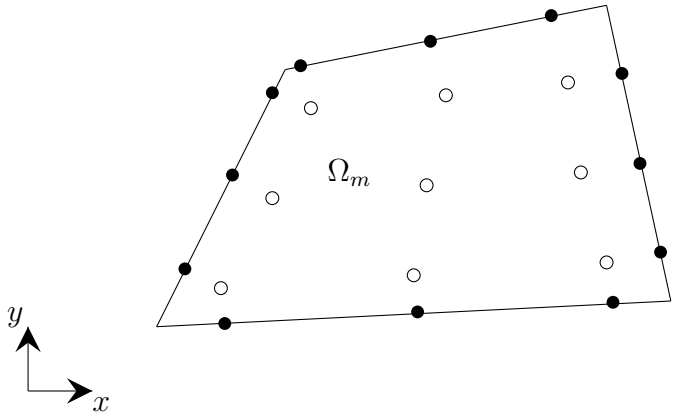


Figure 2.2.1: Arbitrary quadrilateral element Ω_m in physical space (x, y). Hollow circles represent solution points and solid circles represent flux points.

We assume that the element of figure 2.2.1 is part of a domain on which we consider the following advection type problem

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0. \tag{2.2.1}$$

In equation 2.2.1, $\mathbf{U}$ is the vector of the conserved quantity, $\mathbf{U} = (u_n, n = 1, ..., N)$ with N its number of components. $\mathbf{F} = \mathbf{F}(\mathbf{U})$ is the flux of $\mathbf{U}$. As mentioned in section 2.1, it is advantageous to transform 2.2.1 into a space that would allow more efficient computational implementation. Transforming element Ω_m of figure 2.2.1 into reference space results in the element $\hat{\Omega}_m$ of figure 2.2.2. The reference quadrilateral element in two dimensions is a square of dimensions 2×2.

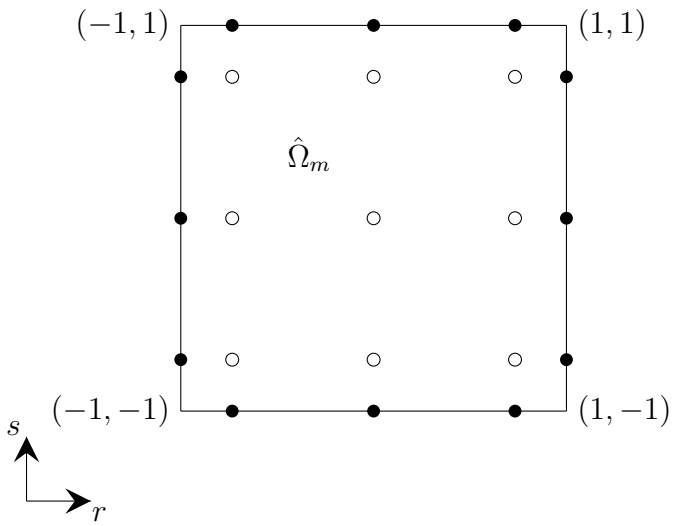


Figure 2.2.2: Element of figure 2.2.1 transformed into reference space (r, s), $\hat{\Omega}_m$. The (r, s) coordinates of the corners of the square are marked.

The transformation $(x, y) \rightarrow (r, s)$ is achieved via the suitable mapping $\mathbf{G}$ below

$$\mathbf{x} = \mathbf{G}(\hat{\mathbf{x}}), \tag{2.2.2}$$

with $\mathbf{x} = (x, y)$ and $\hat{\mathbf{x}} = (r, s)$.

The reverse procedure would naturally be defined as

$$\hat{\mathbf{x}} = \mathbf{G}^{-1}(\mathbf{x}) \tag{2.2.3}$$

Mapping equation 2.2.1 into transformed reference space (r, s) results in the following expression for a computational grid that does not alter in time (i.e. static)

$$\frac{\partial \mathbf{U}}{\partial t} + (\det \mathbf{J}^{-1})\hat{\nabla} \cdot \hat{\mathbf{F}} = 0. \tag{2.2.4}$$

In 2.2.4, $\hat{\nabla} = \partial/\partial\hat{x}_i$. $\hat{\mathbf{F}}$ is the flux in transformed space. Furthermore, $\mathbf{J}$ is the Jacobian of the transformation matrix.

$$\mathbf{J} = \frac{\partial G_i}{\partial \hat{x}_j}. \tag{2.2.5}$$

with $i = (1, 2)$, $j = (1, 2)$.

The reader should note here that, in order to allow greater efficiency in terms of computational implementation, physical $\mathbf{U}$ instead of transformed $\hat{\mathbf{U}}$ is retained in 2.2.5.

In analogous fashion to section 2.1, once the mapping function and the position of the solution and flux points inside the element have been defined, one needs to define a suitable nodal polynomial basis set that will be used to project the physical solution $\mathbf{U}$ into a polynomial space. Such a basis set has the form $[l_{\mathbf{U}}(\hat{\mathbf{x}})]$ with the properties

$$l_{u_i}(\hat{\mathbf{x}}_i) = 1, \tag{2.2.6}$$

and

$$l_{u_i}(\hat{\mathbf{x}}_j) = 0, i \neq j. \tag{2.2.7}$$

if l_{u_i} is the nodal polynomial associated with solution u_i at solution point i with transformed coordinates $\hat{\mathbf{x}}_i$.

Calculating the, generally, element-wise discontinuous physical solution at each of the solution points depicted in figure 2.2.2, one arrives at the following expression, with $n = 1, 2, ..., N$ and N being the number of conserved variables (which, in the case of the 2D Euler equations, are density, x-momentum, y-momentum and energy $[\rho, \rho u, \rho v, \rho E]$ respectively)

$$u_n(\hat{\mathbf{x}}) = \sum_{j=0}^{K} u_{n,j} l_{u_j}(\hat{\mathbf{x}}_i). \tag{2.2.8}$$

The discontinuous transformed flux $\hat{f_n}^D$ can be expressed in analogous fashion.

$$\hat{f}_n^D(\hat{\mathbf{x}}) = \sum_{j=0}^{K} \hat{f}_{n,j}^D l_{u_j}(\hat{\mathbf{x}}_i). \tag{2.2.9}$$

In 2.2.9 and 2.2.10, $K+1$ is the number of solution points inside the element. In direct analogy to the one-dimensional formulation, one needs to compute a common flux between element interfaces in order to arrive at a continuous representation of the flux. The first stage is to "correct" the discontinuous flux via a suitable set of correction functions $\hat{\mathbf{g}}_i(\hat{\mathbf{x}})$ such that

$$\hat{\mathbf{n}}_{f_i} \cdot \hat{\mathbf{g}}_f(\hat{\mathbf{x}}_{f_i}) = 1. \tag{2.2.10}$$

and

$$\hat{\mathbf{n}}_{f_i} \cdot \hat{\mathbf{g}}_f(\hat{\mathbf{x}}_{f_j}) = 0, i \neq j. \tag{2.2.11}$$

if $\hat{\mathbf{n}}_{f_i}$ is the unit vector normal to the interface at flux point f_i and pointing outwards with respect to the element. The divergence of the correction function $\hat{\mathbf{g}}_i(\hat{\mathbf{x}}_i)$ must sit in the same polynomial space as solution $u(\hat{\mathbf{x}})$. It should also be underlined here that several well-establish schemes such as DG and SD can be recovered by suitable specification of $\hat{\mathbf{g}}(\hat{\mathbf{x}})$ [17], [18], [22]. In the present work, DG was chosen as a "baseline" method. Whilst it is true that the experience the author's research team has gained in DG provided a reliable choice, it is also true that an exhaustive study regarding its suitability for the rod-aerofoil problem was not undertaken. Furthermore, it has been found that DG exhibits greater stability for certain problems than SD [67]. Although the latter observation may not necessarily be extendable to the specifics of the rod-aerofoil problem simulated using hexahedra, it was taken as an indication of the stability characteristics of DG.

In the case of the two-dimensional example, the correction component of the flux $\hat{f}_n^C(\hat{\mathbf{x}})$ for $n = (1, 2)$ is given below.

$$\hat{\mathbf{f}}^C(\hat{\mathbf{x}}) = \sum_{n_{in}=0}^{N_{in}} \sum_{n_f=0}^{N_f} [(\hat{\mathbf{f}} \cdot \hat{\mathbf{n}})^I_{n_{in},n_f} - (\hat{\mathbf{f}}^D \cdot \hat{\mathbf{n}})_{n_{in},n_f})] \hat{\mathbf{g}}_{n_{in},n_f}(\hat{\mathbf{x}}). \tag{2.2.12}$$

In 2.2.13 N_{in} is the number of interfaces of the two-dimensional element and N_f is the number of flux points on the current interface. The quantity $(\hat{\mathbf{f}}^D \cdot \hat{\mathbf{n}})_{n_{in},n_f})$ is the discontinuous flux at flux point (n_{in}, n_f).

Inspecting 2.2.12 further, $(\hat{\mathbf{f}} \cdot \hat{\mathbf{n}})^I_{n_{in},n_f}$ is the transformed flux at flux point (n_{in}, n_f), normal to the interface. This flux is computed in an analogous fashion to the one-dimensional approach: A method suitable for resolving the interface problem between two adjacent elements for the particular governing system of equations is employed. In the case of the Euler equations, a widely used method is a suitable Riemann solver such as [21]. A common interface flux is returned by such a procedure, which is $(\hat{\mathbf{f}} \cdot \hat{\mathbf{n}})^I$. The common flux is the result of information carried by both the current element $(\mathbf{u}_L)$ and its adjacent $(\mathbf{u}_R)$ at the interface in question.

The total, "corrected", transformed flux would now be of the form

$$\hat{\mathbf{f}} = \hat{\mathbf{f}}^D + \hat{\mathbf{f}}^C. \tag{2.2.13}$$

Equations 2.2.14 would result in the final expression for the time derivative of the solution at solution point u_i of location $\hat{\mathbf{x}}_i$ via 2.2.4.

$$\frac{\partial u_i}{\partial t} = -(\det \mathbf{J}^{-1})\nabla \cdot \hat{\mathbf{f}}_{\hat{\mathbf{x}}_i} \tag{2.2.14}$$

Via 2.2.13 and with $K+1$ solution points inside an element, the above expression can be expanded into

$$\begin{aligned}\frac{\partial u_i}{\partial t} = &- (\det \mathbf{J}^{-1}) \left[\sum_{n_{in}=0}^{N_{in}} \sum_{n_f=0}^{N_f} [(\hat{\mathbf{f}} \cdot \hat{\mathbf{n}})^I_{n_{in},n_f} - (\hat{\mathbf{f}}^D \cdot \hat{\mathbf{n}})_{n_{in},n_f})] \nabla_{\hat{\mathbf{x}}_i} \cdot \hat{\mathbf{g}}_{n_{in},n_f}(\hat{\mathbf{x}}_i) \right] - \\ &(\det \mathbf{J}^{-1}) \left[\sum_{n_i=0}^{K} \hat{f}_{\hat{\mathbf{x}}_i} \left[\frac{\partial l_{u_i}}{\partial \hat{\mathbf{x}}} \right]_{\hat{\mathbf{x}}_i} \right]\end{aligned} \tag{2.2.15}$$

Expression 2.2.16 can be marched forward in time via a suitable time-integration method such as high-order, multi-step, explicit RK scheme. The details of such a scheme will be introduced in Chapter 3.

2.2.1 Treatment of Viscous Flux

In the case of the Navier-Stokes equations, the flux should incorporate a viscous part. If the two-dimensional case is considered, the governing system is represented by the following equation.

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f} = 0. \tag{2.2.16}$$

In 2.2.16, $\mathbf{u}$ is the vector of the conserved variables of the compressible Navier-Stokes,

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}. \tag{2.2.17}$$

In 2.2.16, $\mathbf{f}$ is the total flux of the compressible Navier-Stokes, comprising both the inviscid and the viscous part of the flux. In 2.2.17, total energy E is defined as $E = C_p T + 1/2(u^2 + v^2)$, with T being the temperature and C_p the heat capacity of the fluid under constant pressure.

$$\mathbf{f} = \mathbf{f}^{inv} - \mathbf{f}^{visc}. \tag{2.2.18}$$

In equation 2.2.18, inviscid flux $\mathbf{f}^{inv}$ is expanded as

$$\mathbf{f}^{inv} = \hat{x} \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix} + \hat{y} \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}. \tag{2.2.19}$$

Viscous flux $\mathbf{f}^{visc}$ is expanded as

$$\mathbf{f}^{inv} = \hat{x} \begin{bmatrix} 0 \\ T_{xx} \\ T_{xy} \\ u_y T_{yx} + C_\mu \frac{\partial T}{\partial x} \end{bmatrix} + \hat{y} \begin{bmatrix} 0 \\ T_{yx} \\ T_{yy} \\ u_x T_{xy} + C_\mu \frac{\partial T}{\partial y} \end{bmatrix}. \tag{2.2.20}$$

In equations 2.2.19 and 2.2.20, $\hat{x}$ and $\hat{y}$ represent the unit vectors along spatial dimensions x and y respectively. Also, $C_\mu = \frac{\mu C_p}{P_r}$, with μ and P_r being the dynamic viscosity and the Prandtl number respectively. Furthermore, T_{ij} is the stress tensor of the viscous fluid.

The viscous part of the flux should be handled in a manner analogous to the inviscid part, i.e., a common total flux should be computed at the element boundaries. Therefore, the inviscid Riemann solver employed at element interfaces should be adjusted appropriately in order to accommodate the viscous part of the flux, $\mathbf{f}^{visc}$. One approach, and the one employed by the solver to be presented in Chapter 3, is the Local Discontinuous Galerkin approach (LDG) [52], [53].

Following the analysis of [53], in the LDG approach, the common interface solution $\mathbf{u}^I$ is defined as

$$\mathbf{u}^I = \frac{\mathbf{u}_L^D + \mathbf{u}_R^D}{2} - \beta(\mathbf{u}_L^D - \mathbf{u}_R^D). \tag{2.2.21}$$

The common interface solution from equation 2.2.21 is then used to construct a solution correction term $\hat{\mathbf{u}}^C$ inside the interior of the element.

$$\hat{\mathbf{u}}^C = (\hat{\mathbf{u}}_L^I - \hat{\mathbf{u}}_L^D)\mathbf{g}_L + (\hat{\mathbf{u}}_R^I - \hat{\mathbf{u}}_R^D)\mathbf{g}_R. \tag{2.2.22}$$

In equations 2.2.21 and 2.2.22, $\mathbf{u}_L^D$ and $\mathbf{u}_R^D$ are the discontinuous solutions at the left and right sides of the element interface respectively. Quantity β is a directional parameter and will typically have the values of ± 0.5 in order to ensure compact stencil of the scheme. Functions $\mathbf{g}_L$ and $\mathbf{g}_R$ are the suitable left and right corrections functions. More details on the procedure and the nature of the correction functions are given in the Appendix of the present work.

Finally, the total continuous transformed solution $\hat{\mathbf{u}}$ inside the element is constructed as the sum of the discontinuous solution and the correction term, as per equation 2.2.23.

$$\hat{\mathbf{u}} = \hat{\mathbf{u}}^D + \hat{\mathbf{u}}^C = \hat{\mathbf{u}}^D + (\hat{\mathbf{u}}_L^I - \hat{\mathbf{u}}_L^D)\mathbf{g}_L + (\hat{\mathbf{u}}_R^I - \hat{\mathbf{u}}_R^D)\mathbf{g}_R. \tag{2.2.23}$$

Following the computation of the total, transformed solution inside the element (in a manner analogous to the 1D FR approach outlined in section 2.1), the expression of the total discontinuous transformed flux $\hat{\mathbf{f}}^D$ inside the element can be constructed. In the current case of an advection-diffusion type problem the flux is a function of both solution $\hat{\mathbf{u}}$ and its spatial derivatives $\frac{\partial \hat{\mathbf{u}}}{\partial \hat{x}}$. Therefore, at each solution point i, discontinuous flux $\hat{\mathbf{f}}_i^D$ is evaluated as $\hat{\mathbf{f}}_i^D = \hat{\mathbf{f}}^D(\hat{\mathbf{u}}_i, \frac{\partial \hat{\mathbf{u}}}{\partial \hat{x}}|_i)$. The expression of the discontinuous transformed flux inside the element is constructed by use of the familiar polynomial projection in Lagrange space.

$$\hat{\mathbf{f}}^D = \sum_{i=0}^{K} \hat{\mathbf{f}}_i^D l_i. \tag{2.2.24}$$

The construction of the total expression of the flux inside requires the construction of a correction term. This term is constructed following the calculation of a common interface flux between adjacent elements. Working in physical space, the common interface flux will have an inviscid and a viscous component.

$$\mathbf{f}_{comm} = \mathbf{f}_{comm}^{inv} - \mathbf{f}_{comm}^{visc}. \tag{2.2.25}$$

In 2.2.25, common inviscid flux $\mathbf{f}_{comm}^{inv}$ will be computed via a suitable Riemann solver as discussed in section 2.2. In the LDG approach, the viscous part $\mathbf{f}_{comm}^{visc}$ of the flux will have the following form.

$$\mathbf{f}^{visc}_{comm} = \frac{\mathbf{f}^{visc,D,L} + \mathbf{f}^{visc,D,R}}{2} + \tau(\mathbf{u}^D_L - \mathbf{u}^D_R) + \beta(\mathbf{f}^{visc,D,L} - \mathbf{f}^{visc,D,R}). \tag{2.2.26}$$

In 2.2.23, $\mathbf{f}^{visc,D,L}$ and $\mathbf{f}^{visc,D,R}$ are the discontinuous fluxes at the left and right sides of the element interface respectively and τ is a penalty parameter controlling the jump in the solution.

Having computed the common interface fluxes, the correction term $\hat{\mathbf{f}}^C$ inside the element is constructed via a procedure directly analogous to equation 2.2.22.

$$\hat{\mathbf{f}}^C = (\hat{\mathbf{f}}_{comm,L} - \hat{\mathbf{f}}^D_L)\mathbf{h}_L + (\hat{\mathbf{f}}_{comm,R} - \hat{\mathbf{f}}^D_R)\mathbf{h}_R. \tag{2.2.27}$$

In equation 2.2.27, $\mathbf{h}_L$ and $\mathbf{h}_R$ are the left and right flux correction functions respectively.

Finally, the total transformed and continuous flux $\hat{\mathbf{f}}$ inside the element will take the following form.

$$\hat{\mathbf{f}} = \hat{\mathbf{f}}^D + \hat{\mathbf{f}}^C. \tag{2.2.28}$$

Having computed the total transformed flux inside the element, the divergence of it will be marched in time in analogous manner to equations 2.1.21 and 2.1.22.

Chapter 3

Computational Implementation

3.1 Introduction

The FR method outlined in Chapter 2 has been computationally implemented in the form of PyFR [17], [23]. The name PyFR is a portmanteau of the word *Python* and the abbreviation *FR* for Flux Reconstruction, expressing the fact that the solver employs the flux reconstruction method for the solution of the governing system of equations, while making extensive use of the *Python* programming language. The code is an open-source platform and is released under a BSD-type licence [17]. In its current version, PyFR solves the three-dimensional, compressible Navier-Stokes equations without the use of a sub-grid scale model, a technique that can be referred to as under-resolved DNS or implicit LES. The FR implementation for the spatial discretisation can recover several types of schemes via the definition of more than one type of correction function (see Chapter 2), such as the popular DG method. Suitable Riemann solvers for the interface fluxes such as the Roe solver [21] are used and the polynomial order of the FR representation can be set via control of the number of solution points inside each element. It is noted here that, in its current form, PyFR can only specify the same polynomial order for every type of element in a hybrid mesh. An important feature of the code is the use of a transformed, reference space for the flux projection and the calculation of its divergence. This characteristic facilitates the computational implementation of the FR approach and allows for sets of matrix-vector multiplications to be collectively cast into single matrix-matrix multiplications.

The use of Python for the main coding of PyFR codebase provides high code readability and high flexibility for adding new features and modules. The implementation of the FR approach has be designed in a manner that allows the casting of the majority of the operations in the form of matrix-matrix multiplications. These operations constitute the compute-intensive part of a simulation, with the remaining operations such as flux evaluations on solution points being pointwise and, thus, comparatively inexpensive for high polynomial orders [18]. However, the speed of pointwise operations is predominantly bound by memory bandwidth rather than processing FLOPs [18]. The latter can potentially lead to situations where the contribution of pointwise operations into the total wall-clock time is of the same order as the contribution of matrix-matrix multiplications.

3.2 Implementation

3.2.1 Functionality

PyFR supports two types of governing equation systems in two and three dimensions, namely Euler and Navier-Stokes. The code was designed from the outset to simulate compressible flow. However, as of the release of version 1.6.0, the solver can simulate incompressible flow via the use of artificial compressibility and pseudo-time stepping.

The unstructured computational grids can comprise of triangles and quadrilaterals or combinations thereof in 2D, hexahedra, tetrahedra, prisms and pyramids or combinations thereof in 3D.

The spatial order of the FR scheme is controlled via the number of solution points inside each element, a feature that, in principle, allows arbitrarily high spatial order. In practice, the order of the scheme will always be constrained by the available computational capacity and the size of the problem. Furthermore, since the coordinates of the solution points in reference space must be hard-coded in PyFR, there is currently no true, out-of-the-box support for arbitrarily high polynomial order.

In the simulation of compressible flow, time stepping is achieved via a series of explicit integration schemes, namely the explicit (forward) Euler method and the RK34, RK4, RK45 Runge-Kutta multi-stage schemes [24], [25]. Since the

implementation of compressible flow simulation was fundamentally designed from the onset to facilitate massive parallelisation, the choice of explicit time-stepping was considered critical. Therefore, PyFR does not currently support implicit time marching for compressible flow simulation. It is noted here that recent, released versions of PyFR support implicit time integration for incompressible flow simulation. In the case of Runge-Kutta time stepping, each RK stage is dependent upon only the current iteration. As a result, an implementation of a procedure that adjusts the time-step size according to stability considerations at the current iteration is possible. In PyFR, one can opt to use such variable time-stepping size via the use of a suitable controller. The controller adjusts the time-step size at each iteration by approximating the truncation error of the scheme's current time-step and using this information to adjust the following time-step size in a manner that increases stability. Thus, if the approximated error of the current solution is higher than a specified tolerance, a smaller-sized time step is introduced for the following iteration. The approximation of the error is performed via the integration of the solution for the estimation of a suitable norm, such as the L_2 and L_∞ norms. Reference to [18] provides a detailed description of the procedure.

PyFR has been designed as a massively-parallelised solver that can target a variety of computational multi/many-core architectures. The current version (1.6.0) supports Nvidia CUDA, OpenMP, OpenCL and Intel MIC (Xeon Phi) platforms. Computations can be performed in single or double precision.

The basic characteristics of PyFR are summarised in the following table, as of May 2017.

Governing Equations:	Euler, Navier-Stokes
Precision:	Single, Double
Dimensions:	2D, 3D
Computational Grid Type:	Structured, Unstructured
Element Types 2D:	Triangles, Quadrilaterals
Element Types 3D:	Hexahedra, Tetrahedra, Prisms, Pyramids
Spatial Order of Accuracy:	Arbitrary
Steppers (Compressible Flow):	Forward Euler, RK34, RK4, RK45
Steppers (Incompressible Flow):	Backward Euler, BDF2, BDF3
Pseudo Steppers (Incompressible Flow):	Forward Euler, RK3, RK4
Platforms:	Nvidia CUDA, OpenMP, Nvidia OpenCL, Intel MIC

Table 3.2.1: Tabulation of main functionality of PyFR, version 1.6.0. The reader can also refer to [23].

Finally, PyFR supports Python's parallelised HDF5 file format for input (e.g. computational mesh) and output (e.g. solution file).

3.2.2 Main Code Blocks

At the beginning of a simulation, PyFR first generates a series of seven operator matrices. These matrices are invariant during the simulation phase and the constitute the operators needed for the matrix-matrix multiplications of the actual solution procedure. Because of their invariance, all operator matrices can be and are defined at the Python level via the mpmath library [26].

For each computational iteration, conserved variables are stored in a state matrix, with particular attention paid to the sequence with which data are stored in memory. In contrast to the, arguably, more intuitive Array of Structures (AoS) layout, PyFR employs the Array of Structures of Arrays approach (AoSoA). This results in more advantageous memory access patterns, facilitating the vectorisation of compute kernels.

As mentioned, matrix-matrix multiplications constitute the main compute-intensive

phase of a simulation. As a result, the operation is performed by kernels utilising optimised and widely used routines from a "Basic Linear Algebra Subprograms" (BLAS) library. These libraries are available for most computational platforms, including all mentioned in table 3.2.1. For instance, targeting Nvidia GPU hardware for the simulations of this thesis invoked cuBLAS which is implemented and maintained by Nvidia in the CUDA programming language.

Less compute-intensive operations such as flux evaluations on solution points and common interface flux calculations are treated by specialised, pointwise kernels. The Mako templating engine [27] is used in order to facilitate the creation of such kernels in a "Domain Specific Language" (DSL). The kernels are generated and interpreted at runtime. For example, pointwise kernels that target Nvidia GPUs and generated via Mako result in pure CUDA code via the PyCUDA module [28]. Mako templating was chosen as it allows for a number of different platforms to be targeted, without the need of developing and implementing specific code for each platform. It also facilitates the adoption of further platforms in future releases.

One of the most important features of PyFR is its ability to operate on a number of devices of the same or even different platforms (heterogeneous clusters). This is achieved through distributed memory parallelism, implemented via the "Message Passing Interface" (MPI). Python is used for the implementation of the entirety of the distributed memory parallelism functionality via mpi4py [29] and h5py [30] is used for the implementation of the HDF5 protocol for input and output. The ability to use PyFR on devices of different platforms simultaneously rests on the fact that the format of the data that is shared among the ranks of a parallelised simulation has been made platform-independent.

A detailed description of the operation and layout of PyFR, is provided in [17] and [18]. Section 3.3 is devoted to the description of how boundary conditions are implemented in PyFR and, in particular, to an implementation of non-reflecting, characteristic boundary conditions that are directly relevant to the subject of the present thesis.

3.3 Boundary Conditions

Considering the boundary conditions of a compressible flow simulation, the presence of non-physical, acoustic of wave reflection from domain boundaries can contaminate the flow field, the modelling accuracy of which being critical for the calculation of quantities such as velocity spectra.

Thus, this section presents a description of the BCs for the solution of the compressible Navier-Stokes available in PyFR; Characteristic, non-reflecting BCs especially relevant to the research presented in the thesis will be derived and their implementation outlined.

3.3.1 Solid Boundaries

In PyFR, solid boundaries can be modelled by means of two, weakly imposed, no-slip BCs: The adiabatic and the isothermal wall.

For an adiabatic wall, wall-normal temperature gradients must be zero, i.e. heat flux through the wall is thus zero. This is implemented in PyFR in the following manner.

We assume a vector of the 5 conserved variables (in the three-dimensional case), i.e. density variable, three momenta variables and the energy variable.

$$\mathbf{U}_w = [\rho_w \quad \rho u_w \quad \rho v_w \quad \rho w_w \quad \rho E_w]. \tag{3.3.1}$$

In 3.3.1, $\mathbf{U}_w$ is the vector of the conserved physical variables on a single point (note, not necessarily a solution point) on a solid wall. Furthermore, ρ_w, u_w, v_w, w_w and E_w represent density, the three components of velocity and energy on the wall, respectively.

Since (internal) energy is directly proportional to temperature, it is the wall-normal gradient of the energy that must be kept equal to zero by the adiabatic BCs. Naturally, the no-slip condition also requires that velocities on the wall must be equal to zero. Summarising, we arrive at the following expressions.

$$[\rho u_w \quad \rho v_w \quad \rho w_w] = [0 \quad 0 \quad 0], \tag{3.3.2}$$

$$\frac{\partial(\rho E_w)}{\partial n_w} = 0, \tag{3.3.3}$$

where 3.3.3, n_w is the wall-normal direction.

At each time iteration, PyFR must guarantee that momenta on the wall are zero while the gradient of internal energy along the wall-normal direction should also be zero. This is achieved via the widely adopted 'weak' approach: The fluxes of vector $\mathbf{U}_w$ are not kept fixed on the flux points of the element faces that are adjacent to the solid boundary but are rather obtained by a Riemann-solve procedure similar to the interface problem discussed in Chapter 2. For each time step, PyFR will place additional, or ghost, series of flux points placed on the 'right' side of the solid boundary. Here, right is taken to mean the boundary side of the wall-adjacent element which in this case is a 'ghost' side placed (geometrically) within the solid wall, as is shown in figure 3.3.1.

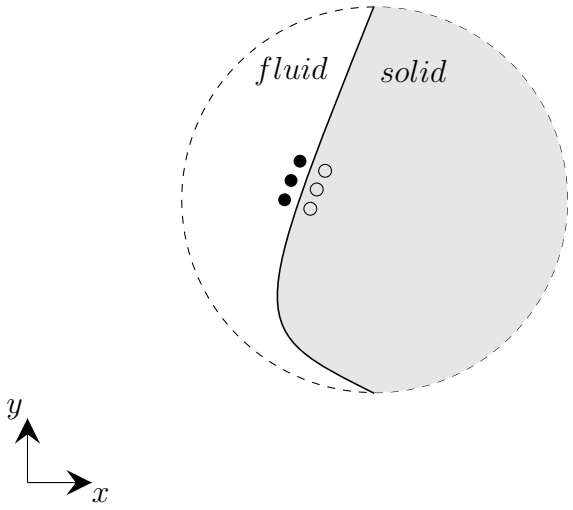


Figure 3.3.1: Imposition of wall boundary conditions via "ghost" flux points. Solid circles represent the flux points of an element interface adjacent to the boundary. 'Ghost' points are represented with hollow circles. A Riemann-solve is performed between the two sets of points in order for the boundary condition to be enforced.

The ghost flux points can be thought of as the mirror equivalent of the original inside the fluid domain. The BC is then imposed by performing a Riemann calculation between the corresponding pairs of flux points on either side of the interface on the solid boundary. The flow variables of the ghost state are set such that the computed interface fluxes will converge rapidly to the required values. In the case

of the velocity components, this is typically achieved by setting, at each time step, ghost-state velocities to have the opposite value of the velocities of the left side, i.e. the side of the Riemann-solve belonging to the physical element of the flow domain. Density is treated by simply translating the same value across the solid boundary and into the ghost element. The right state of energy E is such that 3.3.3 holds as accurately as possible following the Riemann-solve procedure at the boundary. The prescribed right state of E is such that pressure p and velocities u, v, w have zero gradients across the wall.

For an isothermal wall, temperature at the wall must be kept constant and equal to a reference value. Here, the imposition of the BC is identical to the adiabatic wall with the exception of how E_w is treated: Temperature and, therefore, energy must be kept constant. The 'ghost' state of the Riemann-solve corresponding to E is set such that the solution converges rapidly to E_w. The most widely used way of achieving this is to set $E = E_w$ at the corresponding 'ghost' state, with E being the energy at the corresponding solution point of the flow domain in the current iteration.

3.3.2 Far-Field Boundary

Depending on the freestream conditions, PyFR allows for a far-field boundary to be modelled according to two approaches.

Fixed Variables at the Far-Field

The simplest approach involves imposing the freestream values directly in an inflow or outflow far-field region according to the number of in-coming or out-going characteristics respectively. For subsonic flow, and assuming a three-dimensional domain, the inflow boundary has four incoming characteristics and a single out-going characteristic while the outflow boundary has a single in-coming characteristic and four out-going. As a consequence, the current approach prescribes the values of the four primitive variables $[\rho \quad u \quad v \quad w]$ at the inflow boundary and the value of the remaining variable, at the outflow boundary. In PyFR, the variable prescribed at a subsonic outflow region is pressure p. In the case of supersonic flow, all five of the characteristics are in-coming at the inflow region and out-going at the outflow region. As a

consequence, all five primitive variables are prescribed at the inflow boundary, with the outflow region effectively not being treated as a boundary. The following expressions summarise the simplest form of far-field boundary conditions in PyFR.

Subsonic Inflow:

$$\rho = \rho_{fs} \tag{3.3.4}$$

$$u = u_{fs} \tag{3.3.5}$$

$$v = v_{fs} \tag{3.3.6}$$

$$w = w_{fs} \tag{3.3.7}$$

Subsonic Outflow:

$$p = p_{fs} \tag{3.3.8}$$

Supersonic Inflow:

$$\rho = \rho_{fs} \tag{3.3.9}$$

$$u = u_{fs} \tag{3.3.10}$$

$$v = v_{fs} \tag{3.3.11}$$

$$w = w_{fs} \tag{3.3.12}$$

$$p = p_{fs} \tag{3.3.13}$$

In $3.3.4 - 3.3.13$, the subscript (fs) signifies the freestream condition.

Characteristic Boundary Conditions

The simple, inlet and outlet far-field BCs introduced in the previous subsection can prove problematic for simulations in which compressible phenomena are significant. It is never physically possible for a primitive variable to attain a constant value anywhere in a flow field, provided that a disturbance from the freestream conditions is present. Information exchange between any regions of a physical flow domain is always present in the form of waves. Therefore, flow variables at any point must be allowed to adjust by a finite amount in response to dynamic changes elsewhere

inside the domain. Furthermore, one of the most important problems is acoustic wave reflection at the boundary, a phenomenon which is numerically produced if fixed quantities are imposed at a far-field boundary, but one that is not part of the physics flow physics. Therefore, a method to dampen wave reflections and to model the boundary more appropriately should be sought. In PyFR, this comes in the form of Riemann-invariant, characteristic BCs. The improved treatment involves the use of an implementation following the work of Jameson *et al.* [31].

Based on a 1D analysis the Riemann invariant arises as

$$R_\infty = u_\infty \cdot \hat{n} - \frac{2c_\infty}{\gamma - 1}, \tag{3.3.14}$$

where R_∞ is the in-coming characteristic, u_∞ is the freestream velocity, $\hat{n}$ is the normal vector to the boundary facing outwards, c_∞ is the speed of sound at the freestream and γ is the ratio of specific heats of the compressible fluid. The characteristic for the outgoing information is represented by

$$R_i = u_i \cdot \hat{n} + \frac{2c_i}{\gamma - 1}, \tag{3.3.15}$$

where subscript (i) denotes information from the interior of the simulation domain.

Contrary to the basic approach of directly prescribing primitive variables at the boundary, the present method is based on the assumption that characteristics R_∞ and R_i remain constant in the idealised case of inviscid, one-dimensional flow governed by the Euler equations. Assuming that the boundary in question is located in an irrotational portion of the domain, the hypothesis of uniform flow, arguably, holds. Therefore the one-dimensional, inviscid characteristic analysis can be applied.

The use of 3.3.14 and 3.3.15 allows the velocity u_b and speed of sound c_b to be computed at the boundary.

$$u_b \cdot \hat{n} = \frac{1}{2}(R_i + R_\infty). \tag{3.3.16}$$

$$c_b = \frac{\gamma - 1}{4}(R_i - R_\infty). \tag{3.3.17}$$

Therefore, if the boundary is an outflow

$$u_b = u_i + [\hat{n} \cdot (u_\infty - u_i)]\hat{n}, \tag{3.3.18}$$

while for an inflow

$$u_b = u_\infty + [\hat{n} \cdot (u_\infty - u_i)]\hat{n}. \tag{3.3.19}$$

The system of equations is closed if one extrapolates the information associated with entropy, s in the interior of the domain. Through setting $s_i = s_\infty$, one can calculate density ρ and pressure p at the boundary as follows

inflow Boundary:

$$\rho_b^{\gamma-1} = \frac{\rho_\infty^\gamma}{p_\infty}, \tag{3.3.20}$$

outflow Boundary:

$$\rho_b^{\gamma-1} = \frac{\rho_i^\gamma}{p_i}, \tag{3.3.21}$$

with pressure arising from

$$p_b = \frac{(\gamma-1)^2(R_i - R_\infty)\rho_b}{16\gamma}. \tag{3.3.22}$$

The above relationships express the fact that the set $[\rho_b \quad u_b \quad p_b]$ is allowed to fluctuate according to out-going information from the interior of the domain $[\rho_i \quad u_i \quad p_i]$. In similar fashion to the technique used for the imposition of other BCs in PyFR, $S_b = [\rho_b \quad u_b \quad p_b]$ is used as the "ghost" state for the Riemann-solve on the boundary. The 'left' state is $S_i = [\rho_i \quad u_i \quad p_i]$.

In accordance to the description given in section 3.3.3, a pointwise kernel through the Mako templating engine must be defined.

```
1  # -*- coding: utf-8 -*-
2  <%namespace module='pyfr.backends.base.makoutil' name='pyfr'/>
3
4  <% gmo = c['gamma'] - 1.0 %>
5  <% gamma = c['gamma'] %>
6
7  <%pyfr:macro name='bc_rsolve_state' params='ul, nl, ur, ploc, t'>
8      fpdtype_t cs = sqrt(${gamma}*${c['p']}/${c['rho']});
9      fpdtype_t s = ${c['p']}*pow(${c['rho']}, -${gamma});
10     fpdtype_t ratio = cs*${2.0/gmo};
11
12     fpdtype_t inv = 1.0/ul[0];
13     fpdtype_t V_e = ${' + '.join('{0}*nl[{1}]'.format(c['uvw'[i]], i)
14                                  for i in range(ndims))};
15     fpdtype_t V_i = inv*(${' + '.join('ul[{1}]*nl[{0}]'.format(i, i + 1)
16                                       for i in range(ndims))});
17     fpdtype_t p_i = ${gmo}*ul[${nvars - 1}]
18                   - ${0.5*gmo}*inv*${pyfr.dot('ul[{i}]', i=(1, ndims + 1))};
19     fpdtype_t c_i = sqrt(${gamma}*p_i*inv);
20     fpdtype_t R_e = (fabs(V_e) >= cs && V_i >= 0)
21                   ? V_i - c_i*${2.0/gmo}
22                   : V_e - ratio;
23     fpdtype_t R_i = (fabs(V_e) >= cs && V_i < 0)
24                   ? V_e + ratio
25                   : V_i + c_i*${2.0/gmo};
26     fpdtype_t V_b = 0.5*(R_e + R_i);
27     fpdtype_t c_b = ${0.25*gmo}*(R_i - R_e);
28     fpdtype_t rho_b = (V_i < 0)
29                     ? pow((1.0/(${gamma}*s))*c_b*c_b, ${1.0/gmo})
30                     : ul[0]*pow(ul[0]*c_b*c_b/(${gamma}*p_i), ${1.0/gmo});
31     fpdtype_t p_b = ${1.0/gamma}*rho_b*c_b*c_b;
32
33     ur[0] = rho_b;
34 % for i in range(ndims):
35     ur[${i + 1}] = (V_i >= 0)
36                  ? rho_b*(ul[${i + 1}]*inv + (V_b - V_i)*nl[${i}])
37                  : rho_b*(${c['uvw'[i]]} + (V_b - V_e)*nl[${i}]);
38 % endfor
39     ur[${nvars - 1}] = p_b*${1.0/gmo}
40                      + 0.5*(1.0/ur[0])*${pyfr.dot('ur[{i}]', i=(1, ndims + 1))};
41 </%pyfr:macro>
```

Figure 3.3.2: PyFR pointwise kernel for the Riemann-Invariant characteristic BCs. Lines 12 to 31 compute the relevant quantities defined in equations 3.3.14 to 3.3.22. Lines 33 to 40 then impose the boundary condition.

Figure 3.3.2 contains the kernel-generating code written in a domain specific language (DSL) using the Mako templating engine that imposes the Riemann-invariant BCs along the far-field boundary at every iteration. The DSL is comprises python and a platform-specific language such as CUDA for Nvidia GPUS. The operation of the kernel does not require user specification of the type of far-field boundary it will act upon, i.e. inflow or outflow. The distinction between the two types is made automatically, without the need of specialised code for each case. It is important to stress here that the characteristic analysis does not depend on the freestream Mach number M_∞. Supersonic and subsonic freestream conditions are, consequently, treated in the same manner as well.

An illustration of the implementation of Riemann-invariant BCs can be given by way of a two-dimensional simulation of the problem that will be introduced in detail Chapter 4.

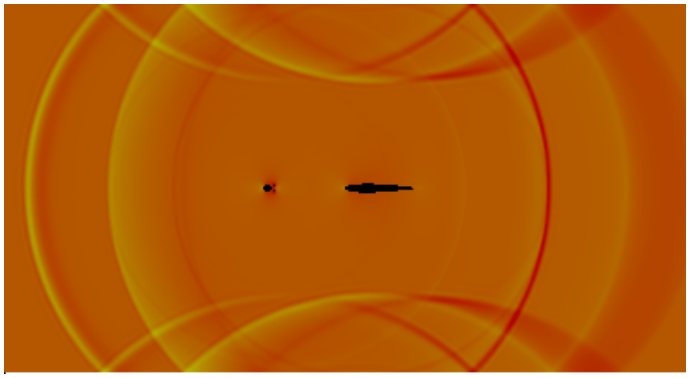


Figure 3.3.3: Acoustic waves generated and reflected by the domain boundaries at the start of the simulation, coloured by pressure magnitude p. Two-dimensional simulation of Rod-Aerofoil configuration with cold start (initial conditions equal to the freestream) and boundary conditions imposed as per section 3.3.2.1. Reynolds number $Re = 480,000$, Mach number $M = 0.21$. PyFR version 0.1.0 [23]. Far-field boundary conditions: $[rho_{fs} \quad u_{fs} \quad v_{fs}]$ imposed at the inlet (left-hand side edge, upper and lower edges), $[p_{fs}]$ imposed at the outlet (right-hand side edge). Acoustic wave reflection from the domain boundary is evident.

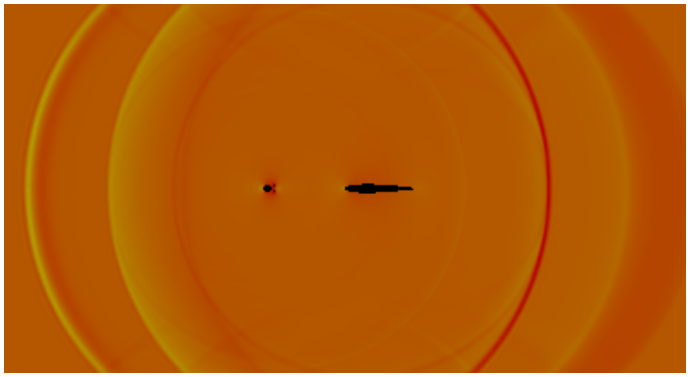


Figure 3.3.4: Physically realistic acoustic waves generated at the start the simulation, coloured by pressure magnitude p. Two-dimensional simulation of Rod-Aerofoil configuration with cold start (initial conditions equal to the freestream) and boundary conditions imposed as per section 3.3.2.2. Reynolds number $Re = 480,000$, Mach number $M = 0.21$. PyFR version 0.1.0 [23]. Colour map rendered in Paraview [32]. Riemann-Invariant boundary conditions imposed on all edges of the domain. Colouring scale of figure 3.3.3, acoustic wave reflection is dampened.

Figures 3.3.3 and 3.3.4 contrast instantaneous pressure fields p, at a time shortly following a "cold" start of a two-dimensional simulation of the rod-aerofoil problem. Image 3.3.4 visibly illustrates the superior performance of the Riemann-Invariant far-field BCs. Specifically, the Riemann-invariant BC is able to prevent the artificial reflection of pressure waves at the outer boundary. However, more sophisticated apporaches that can form part of future research include the Navier-Stokes Characteristic Boundary Conditions (NSCBC).

3.4 Summary

In Chapter 3, an overview of the computational implementation of the FR method, PyFR, was provided. The geometrical flexibility of the code (support for unstructured grids), as well as the functionality that allows its deployment on multiple and/on heterogeneous, massively parallel platforms was underlined. Particular attention was paid to how boundary conditions are implemented in PyFR. A detailed analysis of the implementation of Riemann-Invariant characteristic BCs for the far-field was given. The motivation behind the adoption of the Riemmann-Invariant BCs was based on the desire for physical realism in compressible flow simulation central to the research presented in the following chapters. The sigificant impact of the use of Riemann-Invariant BCs was demonstrated via the qualitative assessment of the reflection of strong acoustic waves. In Chapter 4, the rod-aerofoil problem and the challenges addressed by PyFR in simulating the flow are presented.

Chapter 4

The Rod-Aerofoil Test Case

4.1 Geometry

The Rod-Aerofoil configuration shown in figure 4.1 consists of a cylindrical rod and a symmetrical aerofoil positioned symmetrically in tandem behind the rod. The aerofoil section has the same shape as that of the NACA 0012 and the rod has a diameter of $0.1C$, where C denotes the chord length of the aerofoil.

Consider a two-dimensional cross-section of the layout such that the cross-sectional area of the rod becomes a circle. The line defining the chord of the aerofoil intersects the centre of the circle. The distance between the circle centre and the leading edge of the aerofoil is equal to C. The cross-section is depicted in the following figure

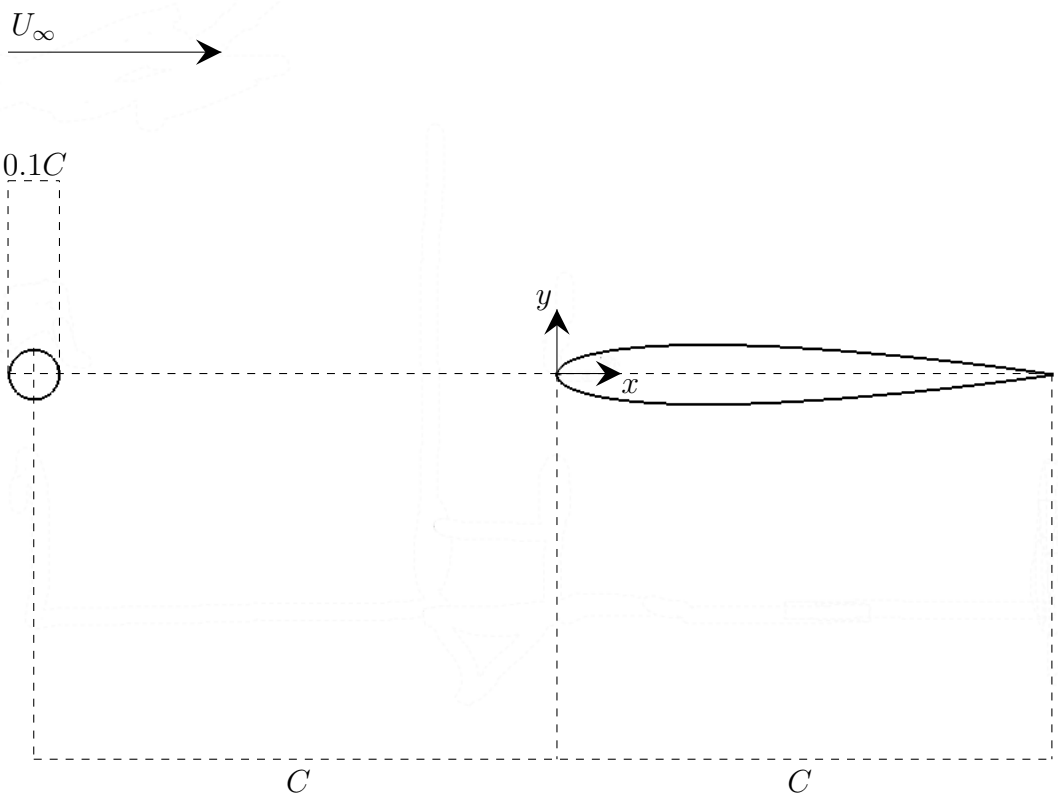


Figure 4.1.1: Cross-section of the rod-aerofoil configuration. U_∞ denotes the freestream, incoming velocity.

Using the freestream velocity U_∞, the freestream density ρ_∞ and the aerofoil chord C, the Reynolds number is $Re = 480,000$, and the Mach number is $M = 0.21$. In physical dimensions, if $C = 0.1m$ and freestream pressure $P_\infty = 98,900Pa$, the above Reynolds and Mach numbers correspond to a freestream velocity of $U_\infty = 72.0m/s$, a freestream temperature of $T_\infty = 293K$ and a freestream density of $\rho_\infty = 1.2kg/m^3$ [19]. In discussions and considerations to follow in the present work, time t and velocity u are non-dimensionalised as follows

$$t_{nd} = t\frac{U_\infty}{C}, \tag{4.1.1}$$

and

$$u_{nd} = \frac{u}{U_\infty}, \tag{4.1.2}$$

with t_{nd}, u_{nd} the non-dimensional time and velocity respectively.

The flow characteristics associated with the rod-aerofoil problem constitute a

challenging benchmark against which the performance of any CFD method can be assessed [19]. Section 4.2 will summarise previous efforts directed at the present geometry before the results of the present work are reported and discussed.

4.2 Previous Studies

4.2.1 Experimental Study

Jacob et al. [19] performed an experiment to measure and document the flow dynamics around the rod-aerofoil configuration. The wind tunnel as a whole was placed in a large anechoic chamber. The spanwise dimension of the configuration was equal to $0.3C$. A range of Reynolds and Mach numbers were studied, with most published results arising from the default condition of $Re = 480,000$ and $M = 0.21$. The atmospheric pressure was measured as $p = 98,900Pa$. One of the fundamental quantities of the flow recorded, following the convergence of the turbulence statistics, was a vortex shedding frequency in the rod wake of $1.3kHz$, corresponding to a Strouhal number of $St = 0.19$. Particle image velocimetry (PIV) and hotwire measurements were performed in order to extract instantaneous flow-field data. Time-averaged values of the velocities and RMS values of their fluctuations were derived. The results obtained demonstrated a highly complex flow field, dominated by large-scale vortical structures generated by the cylinder wake, as well as a range of smaller scales produced primarily via the impingement of the vortices onto the aerofoil leading edge. The complexity of the flow field was quantified via time-averaging of velocities and Reynolds stresses as well as spectra of velocities and pressures. The authors also performed a series of CFD simulations as an initial approach into comparison and validation. The CFD simulations utilised two of the most widely used approaches into turbulence modelling, RANS and LES. These studies are discussed alongside a number of other computational studies in the following subsection.

4.2.2 CFD Studies

The CFD studies performed by Jacob *et al.* [19] utilised unsteady RANS for the compressible Navier-Stokes equations. The solver used a 4^{th} order, centred, finite-volume scheme for the convective flux and a 2^{nd} order scheme for the diffusive flux. The solution was advanced in time via an explicit, five-stage, Runge-Kutta scheme. The 2D RANS computation was performed on a structured mesh of approximately $70,0000$ grid points, which corresponds to approximately $280,000$ degrees of freedom. Grid-point spacing in the vicinity of the solid wall corresponded to a $y^+ < 8$ for the boundary layers. A time step of $\delta t = 5 \times 10^{-8} sec$ was used. The 3D RANS computation made use of a denser grid in the vicinity of the solid walls, providing a near-wall distance of $y^+ < 1.3$. The number of degrees-of- freedom for the three-dimensional case was approximately 10^6. The RANS computation yielded a Strouhal number for the vortex shedding frequency in the rod wake was over predicted at 0.24 for the 2D case (were vortex dynamics are not properly resolved) and 0.23 for the 3D case, thus both predicted values were significantly higher than the one measured, a behaviour that arguably reflects the inability of both computations to properly resolve the vortex dynamics. With LES subgrid scale modelling employed, vortex shedding was predicted significantly more accurately, with an a computed Strouhal number of 0.19, reflecting the ability of LES to capture the turbulent dynamics. All unsteady flow modes were significantly more accurately modelled as well.

Magagnato et al. [33] were amongst the first to focus both on the aerodynamics and the aeroacoustics of the problem. Their studies used three approaches, a 2D RANS simulation with a $k-\tau$ model for turbulence, LES with the Smagorinsky and Lilley subgrid scale model and finally LES with an adaptive $k-\tau$ model. Sound calculations were performed with the Ffowcs-Williams Hawkings analogy (FW-H) [34]. The 3D domain was meshed in a structured manner facilitating a finite volume mesh and keeping the grid points closest to the wall spaced such that $y^+ =\sim 1$ along the aerofoil surface. Spatial accuracy was 2^{nd} order and both implicit and explicit time integration was tested. The densest grid was composed of approximately 3.2×10^6 elements, with the spanwise dimension of the domain equal to two rod diameters. The inlet boundary was modelled through the use of Riemann-invariant,

non-reflecting boundary conditions. The 2D studies presented a loss of accuracy in capturing vortex dynamics. The 3D LES simulation was able to reproduce a more accurate Karman vortex street and performed significantly better across a broader range of scales. While averaged aerodynamic and aeroacoustic results produced by LES were in good agreement with experiment, frequency spectra of instantaneous velocities lacked in areas were flow dynamics were not properly resolved. 2D RANS simulations produced significantly less accurate results. An adaptive $k-\tau$ model for LES was found to perform generally better than the standard Smagorinsky, albeit at a greater computational cost.

Peth *et al.* [35] approached the problem by dividing it into two computational tasks. First, the flow field was calculated by an implicit LES (DNS with no subgrid scale modelling) incompressible solver that, due to the relatively low Mach number, treated the flow as purely hydrodynamic, with compressibility effects neglected. Secondly, all acoustic computations were performed by solving a linearised version of the compressible Navier- Stokes called the Linearised Perturbed Compressible Equations (LPCE). The hydrodynamic part of the problem is solved in 3D while all acoustic computations were performed in 2D. The LES grid was structured composed of $\sim 3.14 \times 10^6$, with $y^+ =\sim 10$ in the vicinity of the solid wall. The LPCE grid consisted of $\sim 160,000$ elements. The hydrodynamic results agreed well with experiment, with a computed Strouhal number value being $St = 0.185$. It was found that acoustic results arising from the linearised approach were heavily dependent on grid resolution with the highly resolved cases proving reasonably reliable for a broad range of frequencies according to the authors.

An application of Detached Eddy Simulation (DES) to study the present problem was the the work by Greschner et al. [36]. This hybrid modelling approach for the flow calculation consisted of a RANS-type modelling for near-wall regions (in the absence of a law-of-wall model) and an LES-type subgrid scale modelling for the remaining domain. The acoustic spectra were extracted by an FW-H analogy implementation. The authors used a structured grid on which DES computations were performed through a finite volume formulation, with approximately 12.5×10^6 degrees-of-freedom. DES was shown to have performed well for the aerodynamic field and the FW-H implementation predicted sound generation and propagation

similarly well.

Berland *et al.* [37] attempted to perform the full aerodynamic and aeroacoustic simulations using a single compressible solver. This approach of acoustic modelling is referred to as Direct Noise Calculation (DNC) and, in similar fashion to Direct Numerical Simulation (DNS), it is computationally expensive. The simulation made use of a massively-parallel, explicit LES solver, with spatial and temporal resolution being suitable for aeroacoustic phenomena to be properly captured. Space discretisation was a 6^{th} order, finite difference scheme. Additional selective solution filtering was used, in the form of spectral-like filtering stencils, in order to dampen spurious oscillations due to the high-order nature of the scheme. The solution was advanced in time by 10^6 time steps on 1024 CPU cores, with time step of $\delta t = 6.5 \times 10^{-8} sec.$ The computational grid resulted in approximately 10^8 degrees-of-freedom. The authors concluded that a good agreement between experiment and simulation was shown, both in the aerodynamic turbulent statistics and the broadband noise spectra. This seems to indicate that the use of direct numerical simulations for the resolution of the entirety of turbulent scales performs well and is justifiable when access to massively-parallelised codes and architectures is possible.

The work of Galdeano *et al.* [38] was a further example of the use of DES coupled with a FW-H solver for the acoustic calculations. The simulations made use of an unstructured, 3D mesh consisting of $\sim 10^7$ nodes. In accordance with previous findings, the team concluded that DES performs reasonably well in both aerodynamic results and as a base for an FW-H analogy to be deployed. Acoustic spectra were demonstrated to correlate well with experiment, over a broad range of frequencies.

Use of LES modelling was also made by Eltaweel *et al.* [39]. The simulations were performed on unstructured grids of approximately 2.2×10^6 elements. The LES solver treated the low Mach number flow as incompressible and the acoustic spectra were computed *a posteriori* by solving Lighthill's equations with a Boundary Element Method (BEM). It is worth noting that the computational domain discretisation utilised a multi-layer approach, in order for waves and flow discrepancies to be properly dampened in the far-field. Here again, the authors concluded that LES simulation of the Rod-Aerofoil problem is suitable for aeroacoustics as well as

turbulent calculations. Good agreement with experiment was indeed demonstrated.

One of the more recent studies of the test case was conducted by Giret *et al.* [40]. The authors made use of an LES solver coupled with a FW-H analogy approach. The explicit LES solver used a dynamic Smagorinsky-type subgrid scale model and computations were performed on an unstructured, 3D grid. The simulation made use of non-reflecting, Navier-Stokes characteristic boundary conditions (NSCBC). Explicit time stepping was used, permitting a time step of $\delta t = 2.5 \times 10^{-7} sec.$ The solution was advanced to a physical simulation time of $t = 0.15 sec$ on 1024 CPU Intel Nehalem CPU cores.

Finally, Agrawal *et al.* [41], [42] simulated the flow through the use of two LES solvers. The first was the solver Charles which solved the compressible Navier-Stokes and the second was the solver PisoFoam of the OpenFoam framework which solved the incompressible Navier-Stokes. The acoustic field was approximated via the FW-H analogy applied on the computation that the compressible solver Charles produced and also via Amiet's theory applied on the computation that the incompressible solver PisoFoam produced. The computational grid of the 2016 study [42] comprised 19×10^6 cells and was fully structured. The results were computed via data produced from simulating approximately 104 flow passes over an aerofoil chord. The solvers were shown to be in close agreement regarding the averaged-flow results but the agreement with experiment was shown to be moderate with over-prediction of the velocity deficit and turbulence intensity in the wake of the cylinder. In the near-field velocity spectra, the cylinder Strouhal number prediction as well as the tonal peak level and spectral fall-off were shown to produce good agreement with experiment.

4.2.3 Summary

Previous CFD studies of the rod-aerofoil configuration are summarised in the following table.

Study	*Aerodynamics*	*Acoustics*	*Grid*	*DoF*	St
[19]	Comp. RANS/LES	FW-H	2D/3D Struct.	10^7	0.19
[33]	Comp. LES	FW-H	3D Struct.	-	-
[35]	Comp. LES	LPCE	3D Struct.	2.4×10^6	0.19
[36]	Comp. DES	FW-H	3D Struct.	1.9×10^6	0.20
[37]	Comp. LES	DNC	3D Struct.	2×10^7	0.19
[38]	Comp. DES	FW-H	3D Unstruct.	10^7	0.19
[39]	Incomp. LES	Lighthill	3D Unstruct.	2.23×10^7	-
[40]	Comp. LES	FW-H	3D Unstruct.	4.25×10^6	0.19
[41], [42]	Comp./Incomp. LES	FW-H	3D Struct.	19×10^6	0.19

Table 4.2.1: Tabulation of main CFD characteristics of previous studies of the rod-aerofoil problem. Predicted Strouhal numbers are included.

The results obtained in the studies of table 4.2.1 provide two main conclusions that can be drawn. The first centres on the fact that the quality of an aerodynamic simulation of this type is highly dependent on the accuracy with which the simulation models unsteady, turbulent phenomena. This is evident by the results of an unsuitable or otherwise limited turbulence model relative to better resolved simulations, as was the case between RANS and LES approaches. The second observation centres on the fact that high resolution, direct numerical simulations and unstructured grids do not necessarily offer improvements in efficiency. There are various reasons for this, with one of the most important being computational cost. As is the case with Direct Noise Calculation [37], simulations can become significantly expensive for relatively small amounts of increase in accuracy. One of the objectives of the current work is to assess whether advancements in the area of direct computation will rely on suitable combination of several novel approaches such as high-order FR schemes on unstructured grids. The novelty that is introduced in the present work is the use of a high-order solver (PyFR) and unstructured, 3D meshing to perform DNS of the rod-aerofoil problem. The following chapters are devoted to demonstrating the performance of this approach.

4.3 Introduction to the Computational Grid

In section 3.3.2 the importance of using non-reflecting boundary conditions for the purpose of performing compressible flow simulation was discussed. In addition to simulations utilising the Riemann-invariant characteristic BCs, for the purposes of dampening non-physical wave reflection, the present research employed computational domains with a circular outer boundary along the $x-y$ ($x-y$ defined as per figure 4.1.1). The choice a circular boundary at the far-filed reflected the intention to avoid spurious behaviour of the numerical scheme and the boundary conditions in the vicinity of sharp corners [43]; such a phenomenon can potentially result in an increase in the intensity of non-physical wave reflection.

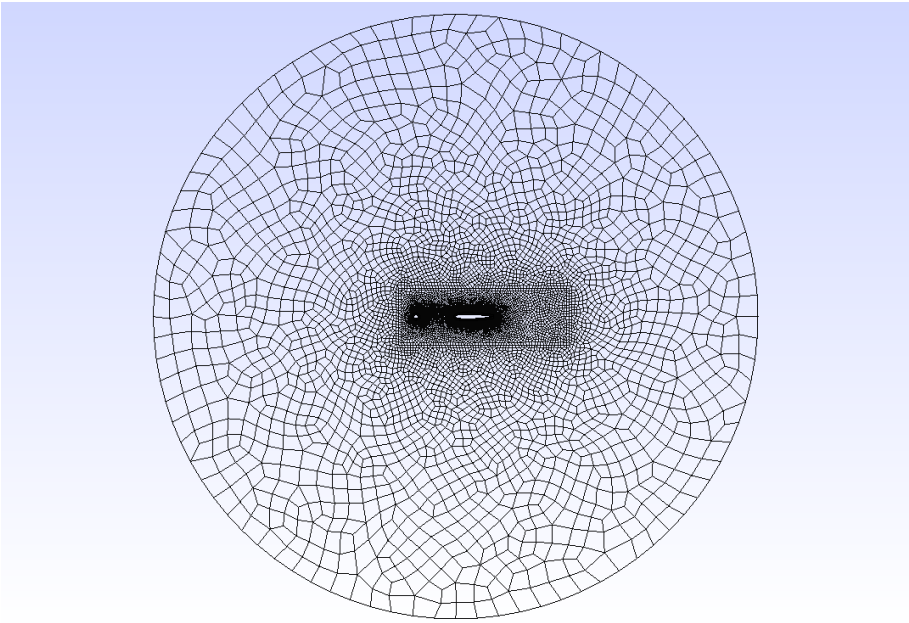


Figure 4.3.1: Cross-section of a computational grid across the $x-y$ plane (see figure 4.1.1). This grid is comprised of approximately $120,000$ quadrilateral elements on the $x-y$ plane. Meshing was done via Gmsh [44].

A further important characteristic of the computational grids is the meshing approach for the boundary layer regions, which were meshed in a structured manner. This approach was chosen in order to allow adequate control over the resolution requirements near walls, found to have significant effect on the quality of the simulation as will be demonstrated in chapter 5. As will also be mentioned further on, the quadrilateral 2D interfaces of the hexahedral elements where cubically curved. The curved elements were considered a necessity if one was to avoid the discretisation

of the aerofoil profile into an unfeasibly high number of elements whilst retaining sufficient accuracy of geometry representation.

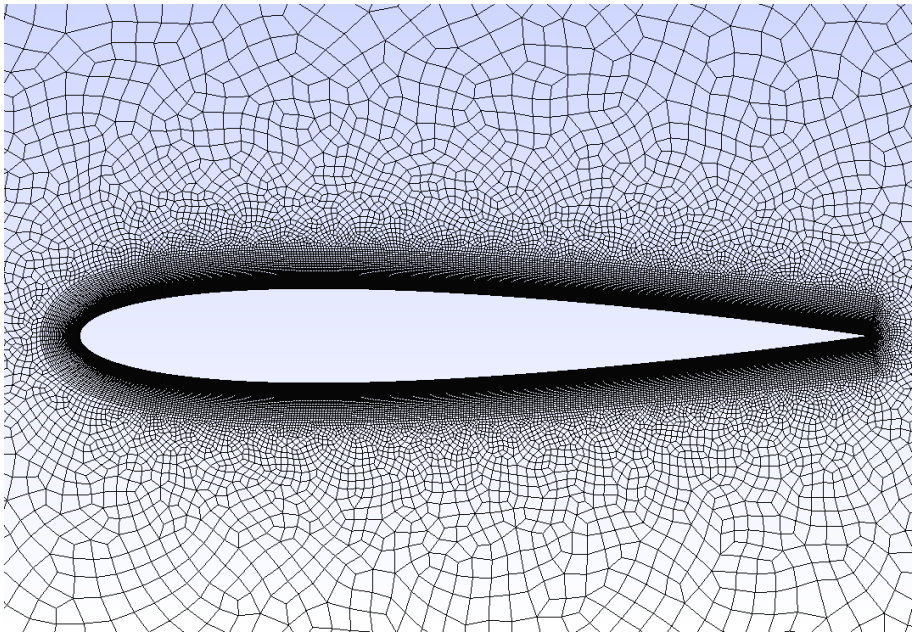


Figure 4.3.2: Magnified version of the mesh around the aerofoil revealing the structured boundary layer approach.

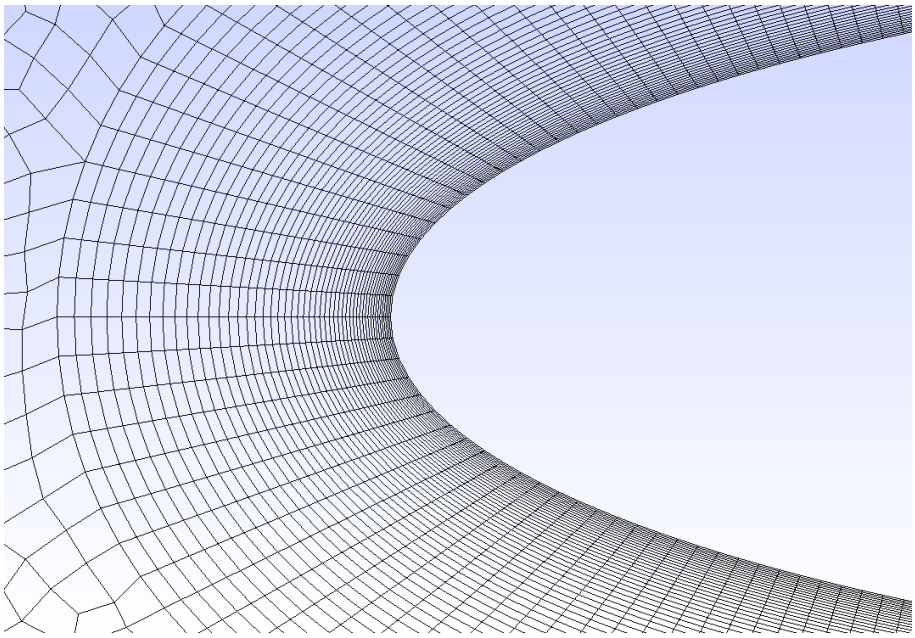


Figure 4.3.3: Magnified view around the aerofoil leading edge.

The quadrilateral elements shown in figures $4.3.1 - 4.3.3$ are cross-sections of the hexahedral elements of the three-dimensional domain. The span-wise discretisation was produced through extrusion of the unstructured 2D $x - y$ mesh across the spanwise z direction. A single extrusion step thus results in a single "slice" of hexahedra. The number of identical slices as well as their thickness can then define the spanwise resolution. All hexahedral elements employed a high-order (curved) representation of their boundaries in the $x-y$ plane. A polynomial order of 3 for the representation was chosen in order to facilitate the FR simulations with polynomial order $P = 3$, adopted for most of the present simulations. High order meshing was achievable through Gmsh, an open-source meshing application [44].

The exact mesh configurations was found to have significant effect on the quality at which flow dynamics were predicted. In particular, it was found that the resolution of the boundary layer region as well as the spanwise resoluton were the two most influential factors. This relationship is discussed in Chapter 5 and a suitable meshing strategy covering the aforementioned areas for the computational needs of the rod-aerofoil problem is identified.

Chapter 5

Computational Grid and Resolution

In order for a meshing strategy of suitable type and adequate resolution to be identified, specifically targeted studies of flows around cylinders were undertaken. The studies considered the challenges that are involved in the modelling of separation and vortex shedding.

The large scale structures of vortex shedding dynamics are predominantly two-dimensional in nature [54]. Consequently, a two-dimensional analysis of the dependence of large-scale vortical structures on boundary layer modelling was sought with the aim of identifying a reference range of resolution that results into correct Strouhal number St prediction. As can be confirmed by both experiment [19] and simulation [33], [35], [36], [37], [38], [39], [40], the expected Strouhal number for a flow of this type and Reynolds number ($Re = 480,000$) around a cylinder is of the $St = 0.19 - 0.20$ range. The breakdown of coherent vortical structures is, however, predominately a three-dimensional phenomenon and its impact on spanwise resolution will be investigating in section 5.2.

5.1 Two-Dimensional Cylinder Studies

Assessment of the resolution of the boundary layer region was made via the accuracy at which the cylinder Strouhal number was predicted in two-dimensional simulations. The cylinder-diameter-based Reynolds number Re_c of the simulations was equal to

the rod-diameter-based Reynolds number of the full rod-aerofoil problem outlined in section 4.1. This particular Reynolds number was chosen since it has been most extensively simulated by other studies. Since for the rod diameter d we have $d = 0.1C$ as per figure 4.1.1, one can write

$$Re_c = 48,000. \tag{5.1.1}$$

The $2D$ cylinder domain was meshed using Gmsh [44]. As mentioned, the boundary layer regions were meshed in a structured manner whilst externally the entire domain was unstructured. The domain utilised quadrilateral elements with the far-field boundary being circular and modelled via the Riemann-invariant BCs described in section 3.3.2.

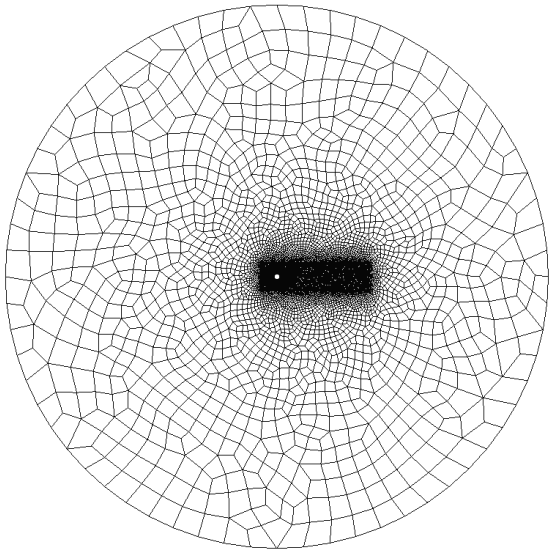


Figure 5.1.1: An example that depicts a two-dimensional mesh around the cylinder with $\sim 12,000.$ quadrilaterals.

Figure 5.1.1 shows one of the computational grids used for the 2D cylinder simulations. All grid iterations utilised unstructured quadrilaterals for the entire domain except the boundary layer region.

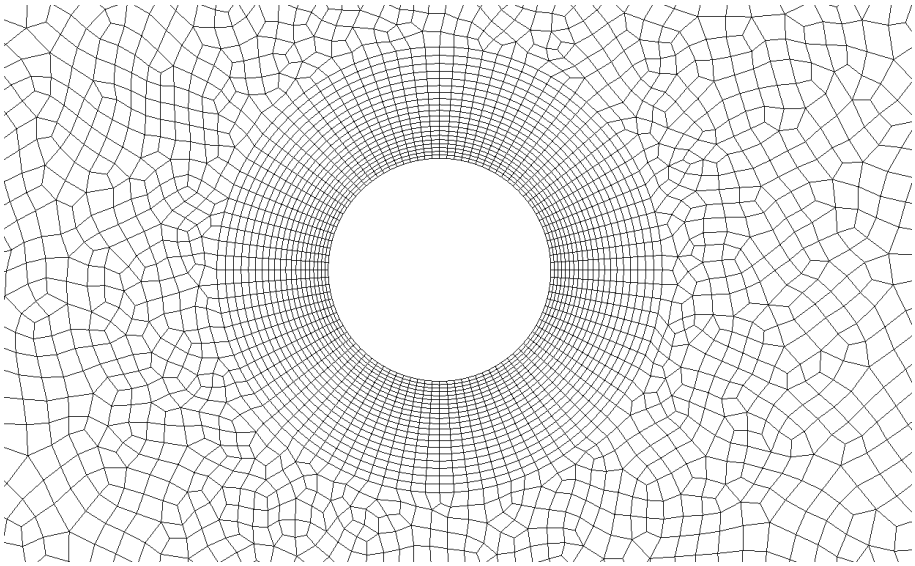


Figure 5.1.2: Structured boundary layer region of figure 5.1.1. This particular example features a division of 100 elements in the circumferential direction and 21 elements in the radial direction. Expansion ratio along the radial direction was 1.05. This example yielded unsuccessful simulations (blow-ups). All meshes that resulted in successful computations are illustrated later and the reader can appreciate the visual differences in resolution.

All grid iterations utilised a radius of $50d$ for the circular far-field boundary. Furthermore, all grid iterations utilised boundary layer regions which where structured, as can be seen on figure 5.1.2. The strategy for controlling the number and aspect ratios of quadrilateral elements inside the structured boundary layer region included defining two parameters. The first parameter was the number of quadrilaterals stacked in the radial direction, i.e., the number of layers the boundary region was subdivided into. The second parameter specified the number of elements in the circumferential direction, i.e. the number of elements the cylinder's perimeter was subdivided into. The expansion ratio for the radial direction was equal to 1.05. In preliminary simulations, it was found that more rapid expansion ratios of the order of approximately 1.1 resulted in severe instabilities although it is noted here that a rigorous study of the effect of the expansion ratio on a mesh of similar type as the one ultimately used was not undertaken. Consequently, a future study towards this direction could result in useful conclusions and/or possible reductions in resolution.

The version of PyFR used for the 2D cylinder studies was 0.2.0 [23]. In order for simulation start-up phenomena (such as the violent acoustic waves generated immediately after a cold start) not to cause stability problems, all simulations

utilised a technique whereby the start-up period of the computation was divided into phases with progressive increase in spatial order. The simulations consisted of the following phases.

- Phase 1: Start of simulation from initial conditions equal to the freestream, non-dimensional conditions (cold startup): $u_{fs} = 1.0$, $v_{fs} = 0.0$, $u_{fs} = 1.0$, $\rho_{fs} = 6106.086$, $p_{fs} = 98900.0$. The Prandtl number was $Pr = 0.72$ and the dynamic viscosity $\mu = 0.127210125$. The ratio of specific heats was set as $\gamma = 1.4$. The non-dimensional freestream conditions as well as the Prandtl number and the dynamic viscosity were such that $Re_c = 48,000$ and $M = 0.21$. The simulation advanced by 4 non-dimensional time units, equal to 40 flow passes over a cylinder diameter using the convention of equation 4.1.1. The polynomial order of FR scheme was $P = 1$.

- Phase 2: Simulation is advanced a further 4 non-dimensional time units with $P = 2$.

- Phase 3: Simulation is advanced a further 4 non-dimensional time units with $P = 3$.

- Phase 4: Simulation is advanced a further 8 non-dimensional time units with $P = 3$. Data extracted.

The specific FR high-order scheme used was Discontinuous Galerkin (DG). Advancement in time was achieved via a five-stage, fourth order, RK45 scheme with a constant, non-dimensional time-step of $\delta t = 0.00002$ for $P = 3$. The solid wall of the cylinder was modelled via the no-slip, adiabatic wall boundary condition. The solution points inside the quadrilaterals were of the Gauss-Legendre type.

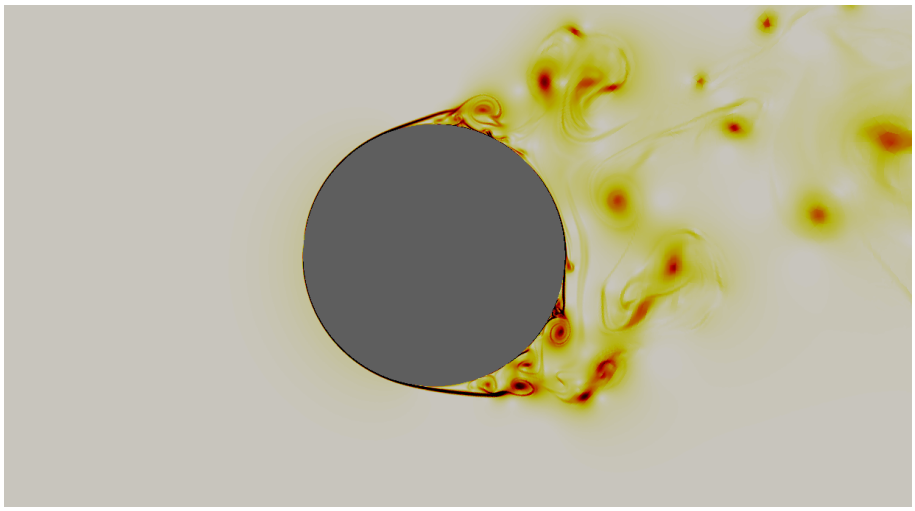


Figure 5.1.3: Flow around 2D cylinder of figure 5.1.3, boundary layer region with 300 circumferential and 51 radial divisions. Instantaneous vorticity magnitude at $t_{nd} = 16.0$, with the simulation having started at $t_{nd} = 0.0$ (equation 4.1.1). Strouhal number $St = 0.199$

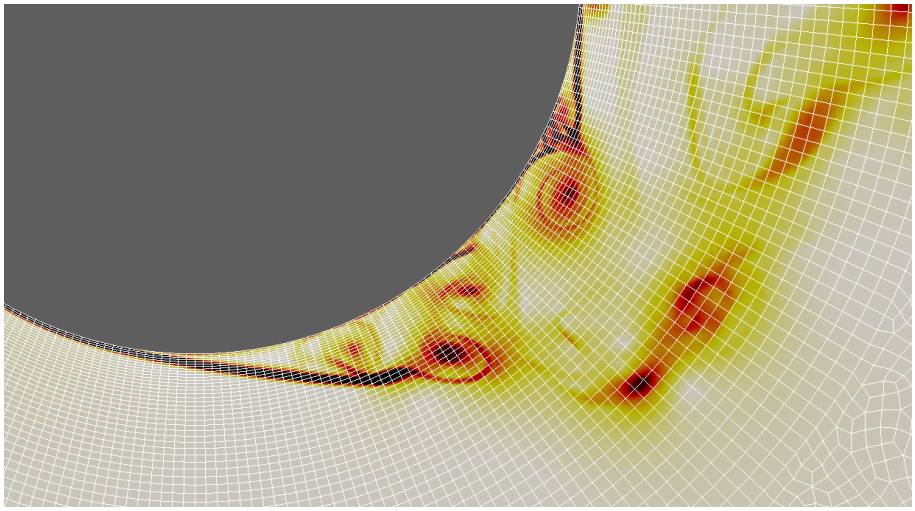


Figure 5.1.4: Flow around 2D cylinder, boundary layer region with 300 circumferential and 51 radial divisions. Zoom-in around the boundary layer region with the computational grid superimposed. Instantaneous vorticity magnitude at $t_{nd} = 16.0$, with the simulation having started at $t_{nd} = 0.0$ (equation 4.1.1). Strouhal number $St = 0.199$.

In order for the simulated Strouhal number of each computational grid iteration to be computed, the instantaneous values $x - component$ velocity u were extracted at point P_1 of figure 5.1.5. Velocity u was sampled every 5×10^{-3} time units.

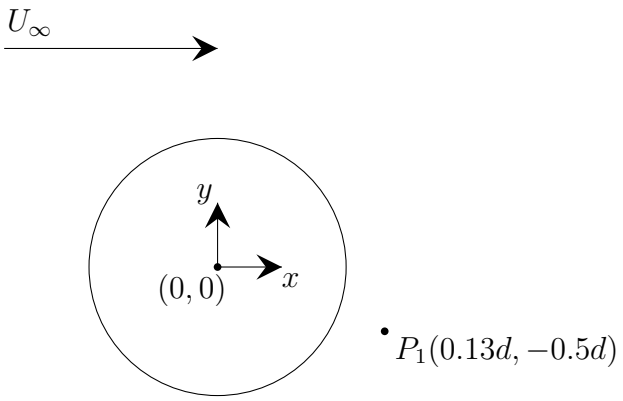


Figure 5.1.5: Position of point P_1. The centre of the cylinder is at $0, 0$ of the x, y plane and d denotes the cylinder diameter.

The various grid iterations of the boundary layer region that were simulated are depicted in figures 5.1.6 - 5.1.9.

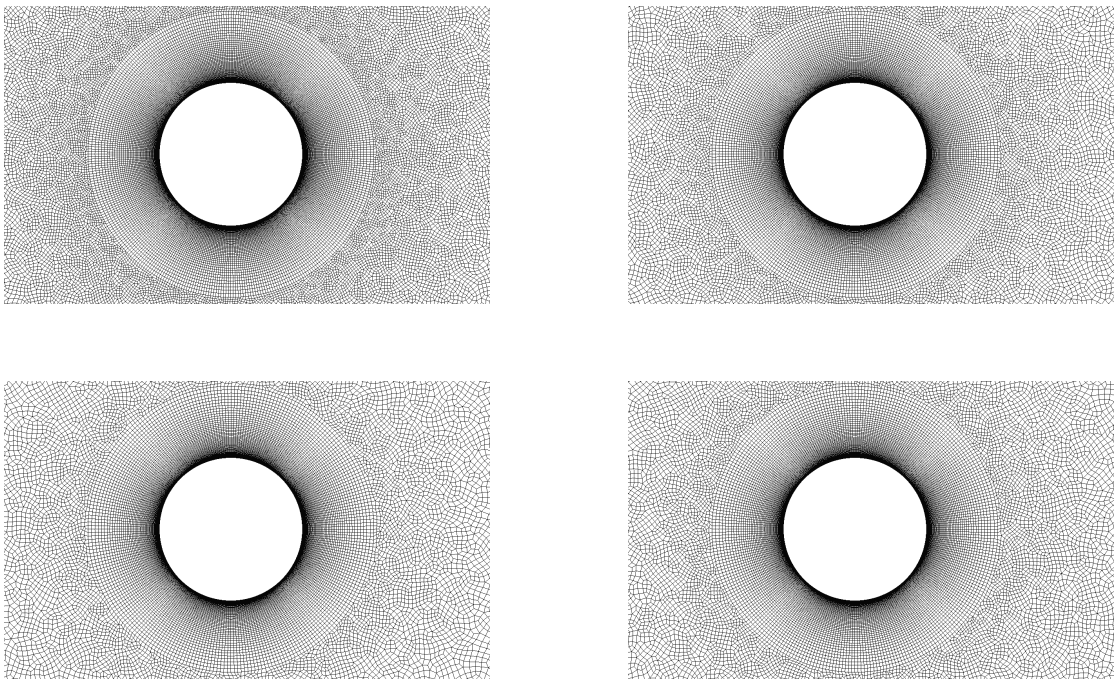


Figure 5.1.6: Boundary layer regions with 61 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 158,000$ elements, 300 circumferential subdivisions and $\sim 113,000$ elements, 280 circumferential subdivisions and $\sim 98,000$ elements, 260 circumferential subdivisions and $\sim 85,000$ elements.

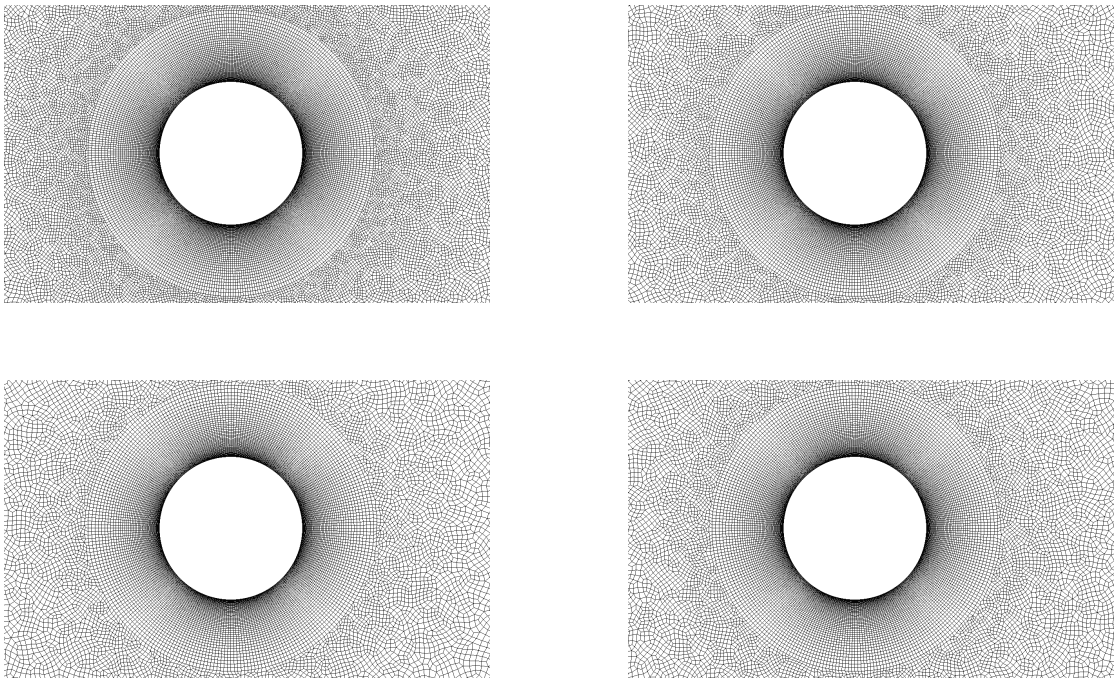


Figure 5.1.7: Boundary layer regions with 51 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 150,000$ elements, 300 circumferential subdivisions and $\sim 100,000$ elements, 280 circumferential subdivisions and $\sim 90,000$ elements, 260 circumferential subdivisions and $\sim 80,000$ elements.

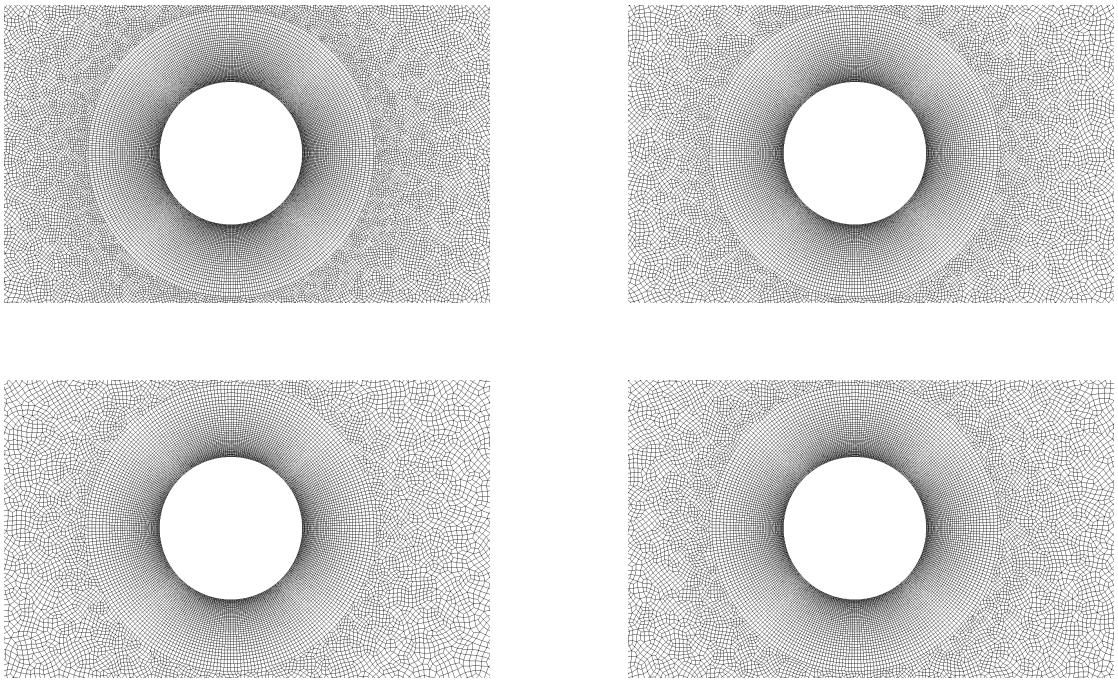


Figure 5.1.8: Boundary layer regions with 41 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 145,000$ elements, 300 circumferential subdivisions and $\sim 95,000$ elements, 280 circumferential subdivisions and $\sim 86,000$ elements, 260 circumferential subdivisions and $\sim 77,000$ elements.

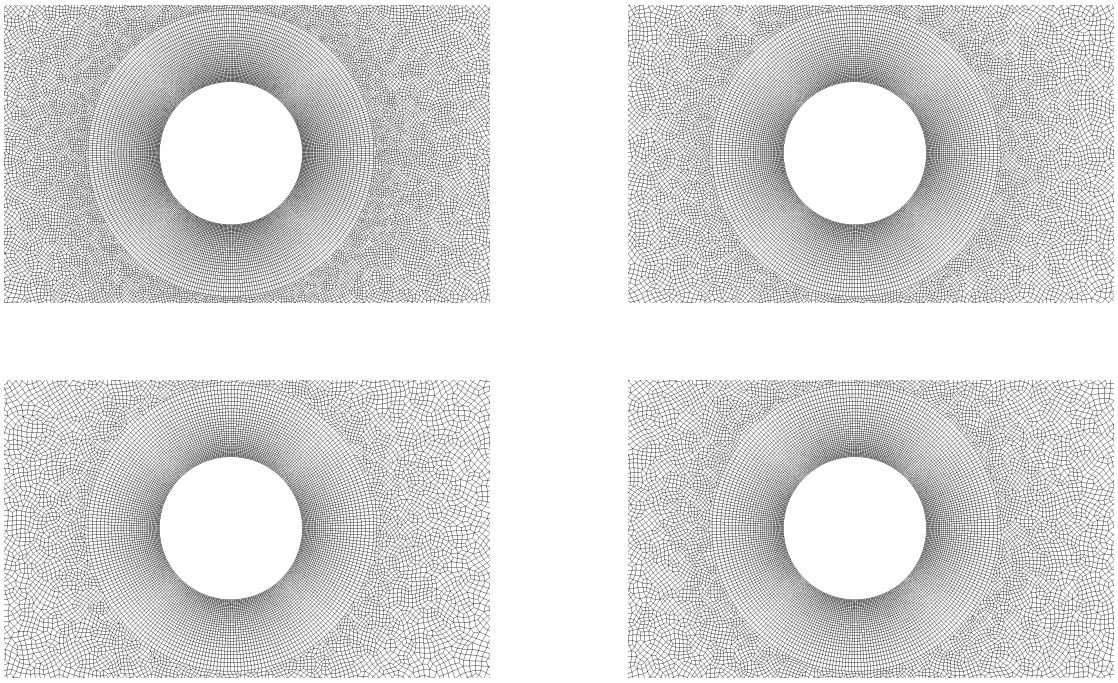


Figure 5.1.9: Boundary layer regions with 31 radial subdivisions. Clockwise from the top left corner: 360 circumferential subdivisions and $\sim 141,000$ elements, 300 circumferential subdivisions and $\sim 93,000$ elements, 280 circumferential subdivisions and $\sim 83,000$ elements, 260 circumferential subdivisions and $\sim 75,000$ elements.

For each grid iteration depicted in figures 5.1.6-5.1.9, a simulation was run and the Strouhal number St was estimated via the power spectrum of velocity u at point P_1.

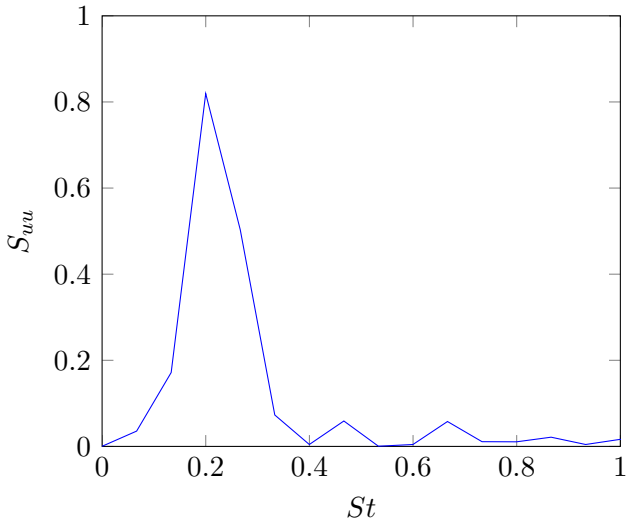


Figure 5.1.10: Power spectral density (PSD) of velocity u (S_{uu}) at point P_1. Grid with boundary layer region of 300 circumferential and 51 radial elements. Strouhal number $St = 0.199$.

The following table is a summary of the dependency of the Strouhal number prediction on the resolution of the boundary layer region. In addition to the sixteen meshes presented in the previous figures, another four meshes of radial resolution of 71 elements were performed. The final four boundary layer meshes of 71 radial elements were simulated via PyFR version 1.4.0 (as opposed to version 0.2.0 used for the initial sixteen meshes), i.e. the same version of the code as the one used in the final, large-scale simulation that will be analysed in Chapter 7. It is noted here that the validity of utilising the results from different versions of the code where checked via the simulation of the mesh with resolution $(61, 360)$ and obtaining the same result as the one listed in table 5.1.1.

	Circumferential Resolution	*360*	*300*	*280*	*260*
Radial Resolution	-	-	-	-	-
71	-	0.190	0.190	0.190	0.192
61	-	0.190	0.190	0.195	0.195
51	-	0.199	0.199	0.200	0.200
41	-	0.220	0.220	0.225	0.230
31	-	NaN	NaN	NaN	0.290

Table 5.1.1: Strouhal number prediction as a function of boundary layer resolution.

Inspection of table 5.1.1 reveals that not all circumferential and radial resolution pairs result in stable simulations. More specifically, if the number of layers across the radial direction is decreased bellow 40, the simulation often failed. More importantly, it is apparent that the quality of the prediction is more sensitive to radial than circumferential resolution. A drop in radial resolution is clearly detrimental to Strouhal number prediction quality whilst a change in circumferential resolution appears to have little effect. If one compares the case of $51 - 260$ for the radial and circumferential resolution respectively with the case of $61 - 360$ which is considered here ideal since the predicted Strouhal number agrees with the experimental $St = 0.19$ [19], there is only a $\sim 5\%$ deviation from experiment for approximately double the problem size ($\sim 158,000$ elements versus $\sim 77,000$ elements). Furthermore, the total size of the grid is mostly dependant on the circumferential resolution rather than the radial. Combining the last three observations, one can conclude that maintaining a radial boundary layer resolution of ~ 50 radial elements or higher is likely to negate the need for higher number of circumferential elements than 260 and a overly expensive simulation. As a result, a baseline boundary layer resolution of 51 radial and 200 circumferential elements was selected for the three-dimensional Rod-Aerofoil problem in order to keep the size of the mesh as low as possible.

5.2 Three-Dimensional Studies

5.2.1 Rod-Aerofoil

Once a strategy for the boundary layer resolution has been established, one can apply it to a simulation of the three-dimensional Rod-Aerofoil problem in order to investigate, in the first instance, the effect that three-dimensionality and the presence of the aerofoil downstream of the cylinder have on large scale vortical dynamics.

A 3D computational grid was defined following the findings of section 5.1. As a result, the boundary layer region around the cylinder included 200 circumferential and 51 radial elements with a radial expansion ratio of 1.05. The boundary layer around the aerofoil incorporated almost identical resolution, i.e., 51 elements in the surface-normal direction with element thickness in the surface-parallel direction being similar to the corresponding thickness of the cylinder boundary layer region.

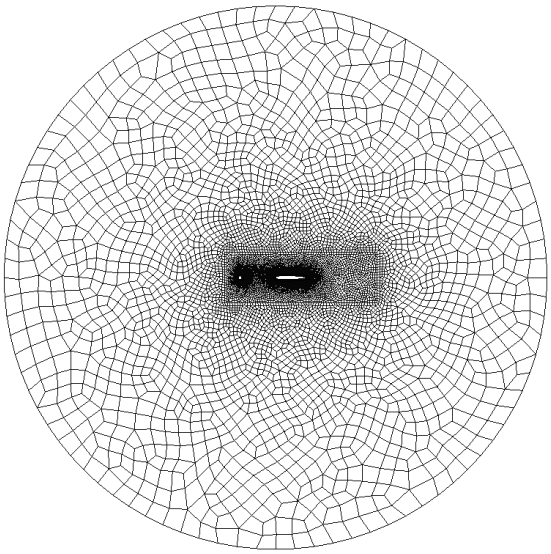


Figure 5.2.1: Computational grid of the preliminary Rod-Aerofoil simulation, $x - y$ plane. The number of hexahedral elements was approximately $270,000$.

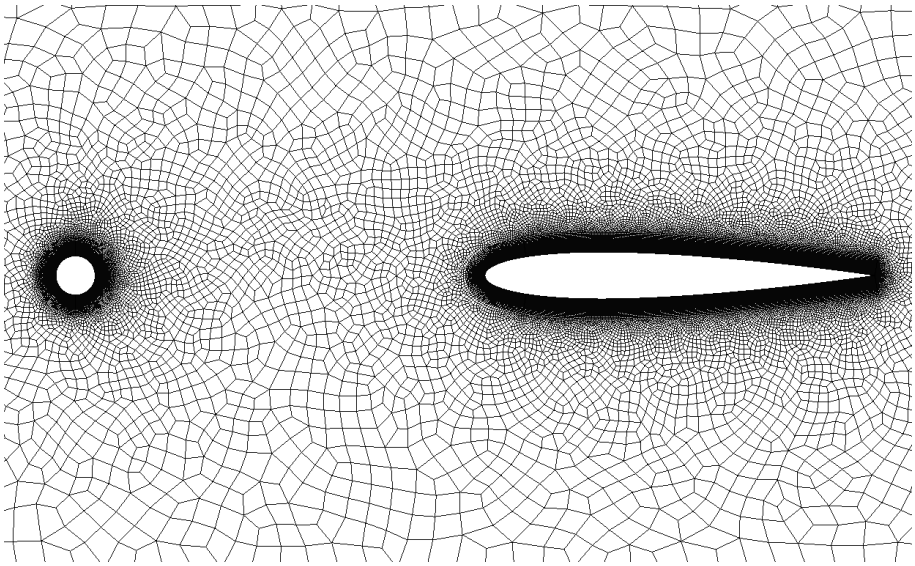


Figure 5.2.2: Zoom-in of figure 5.2.1 around the Rod-Aerofoil area.

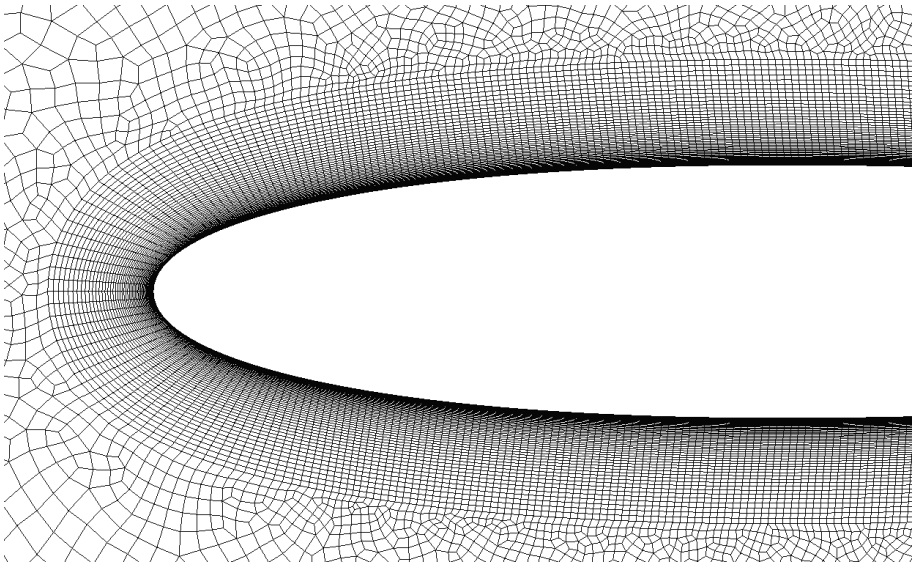


Figure 5.2.3: Zoom-in of figure 5.2.1 around the aerofoil leading edge.

The spanwise dimension of the grid was $0.25C$, divided into 5 slices perpendicular to the z dimension. The resultant mesh comprised of $\sim 270,000$ hexahedral elements. In figures 5.2.4 and 5.2.5 the structured approach for the spanwise meshing is evident.

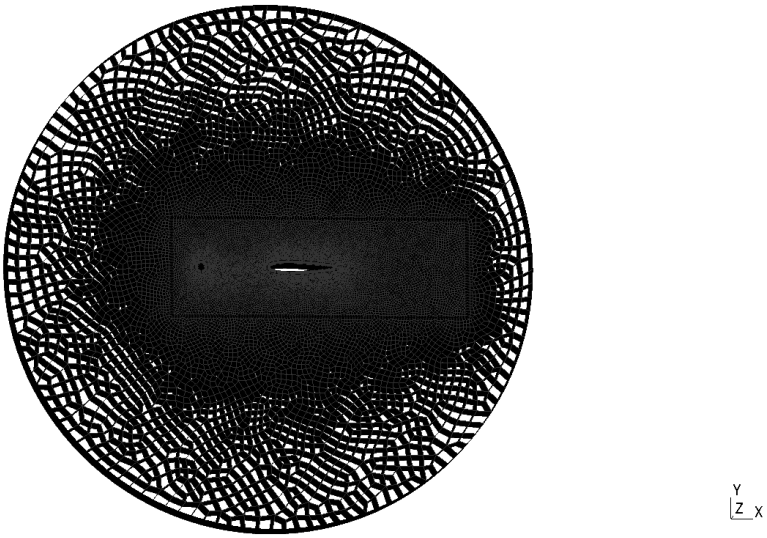


Figure 5.2.4: Three-dimensional depiction of the grid of figure 5.2.1. The sliced-span approach of the meshing is evident across the circular boundary.

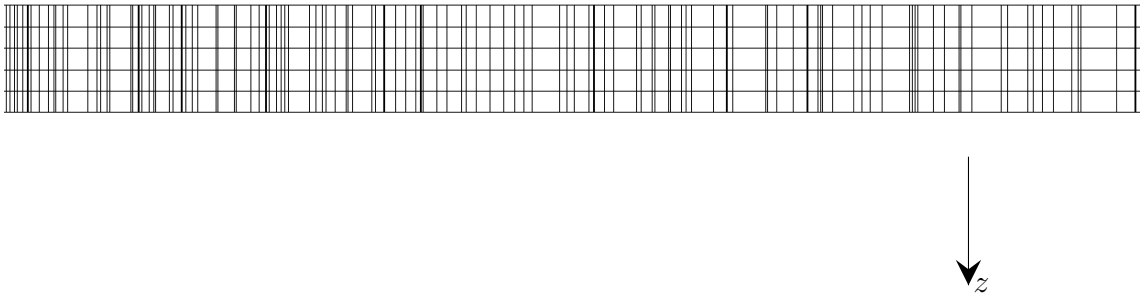


Figure 5.2.5: A $z-y$ cross-section of the mesh of figure 5.2.1 demonstrating the slices of the spanwise direction. Here, spanwise axis z is normal to the $x-y$ plane of figure 4.1.1 and is parallel to the plane of the page pointing downwards.

The simulation was undertaken via PyFR version 0.8.0. Using an order of $P = 3$ for the polynomial representation of the solution inside each hexahedral element, resulted in an approximately $90,000,000$ degree-of-freedom simulation. For reasons that will be analysed in Chapter 6 and in contrast to the 2D cylinder studies, anti-aliasing of both volume and surface fluxes was employed, using a quadrature degree of 11. This was chosen since, with $P = 3$, the number of solution points in one spatial direction inside a hexahedral element is 4. A quadrature degree of 11 for the over-integration procedure results in the utilisation of 6 points across each spatial dimension for the polynomial projection of the flux, i.e., higher accuracy than the one the FR scheme is able to provide alone at $P = 3$. Increasing the degree to more than 11 would however prove more expensive from a computational point of view. Time advancement was achieved via a five-stage, fourth order RK scheme, with the step size equal to $\delta t = 2 \times 10^{-6}$ non-dimensional time units. All the above result in a simulation wall-clock-time equal to approximately one week for a single flow pass over the chord on 20 Nvidia M2090 GPUs.

The strategy employed for the initialisation of the simulation was similar to that

described in section 5.1:

- Phase 1: Start of simulation from initial conditions equal to the freestream, non-dimensional conditions (cold startup): $u_{fs} = 1.0$, $v_{fs} = 0.0$, $u_{fs} = 0.0$, $\rho_{fs} = 6106.086$, $p_{fs} = 98900.0$. The Prandtl number was $Pr = 0.72$ and the dynamic viscosity $\mu = 0.0127210125$. The ratio of specific heats was set as $\gamma = 1.4$.The non-dimensional freestream conditions as well as the Prandtl number and the dynamic viscosity were such that the aerofoil-chord defined Reynolds number Re_a was $Re = 480,000$ and the Mach number $M = 0.21$. The simulation advanced by 8 non-dimensional time units, equal to 8 flow passes over an aerofoil chord using the convention of equation 4.1.1. The polynomial order of FR scheme was $P = 1$.

- Phase 2: Simulation is advanced a further 8 non-dimensional time units with $P = 2$.

- Phase 3: Simulation is advanced a further 30 non-dimensional time units with $P = 3$.

A snapshot of the resultant flow field of the simulation can be seen in figure 5.2.6.

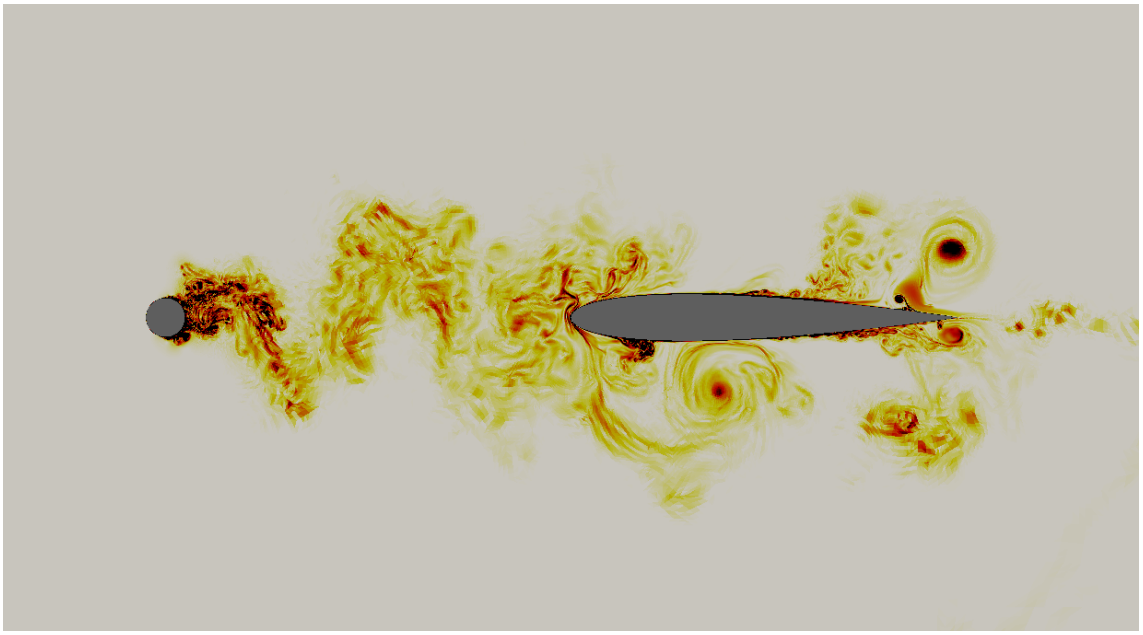


Figure 5.2.6: Instantaneous vorticity magnitude of the preliminary Rod-Aerofoil simulation. If at the start of the simulation we define non-dimensional time as $t_{nd} = 0.0$, the image corresponds to $t_{nd} = 16.5$.

In order to determine a reliable point in the simulation at which transient phenomena (i.e. start-up phenomena) can be assumed as having ceased, one can perform a convergence study.

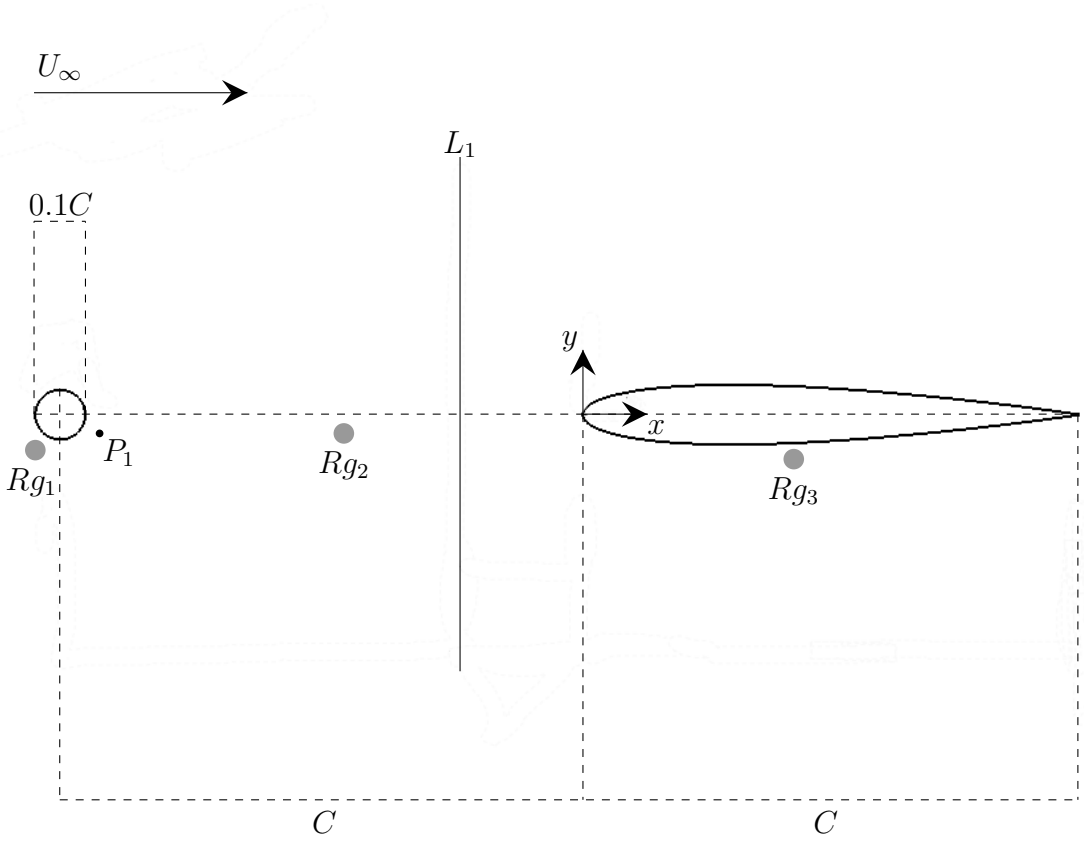


Figure 5.2.7: Definition of data extraction regions. Point P_1 has the same cylinder-relative position in the $x-y$ plane as in figure 5.1.5. Its spanwise location is the middle of the domain. Line L_1 is parallel to the y axis and its distance from the aerofoil leading edge is $0.255C$. Rg_1, Rg_2 and Rg_3 denote regions of differing three-dimensional aspect ratios of hexahedral elements.

The convergence study involved the time-averaging of $x-component$ of velocity, u, through a series of $P = 3$ solution files containing the instantaneous values of all conserved variables across the entire computational volume. These solution files were produced every 0.005 non-dimensional time units. An analysis of the profile of the averaged u across line L_1 of figure 5.2.7 was made: An averaging window of length equal to four time units was shifted along the time direction until time-averaged results converged to a single curve.

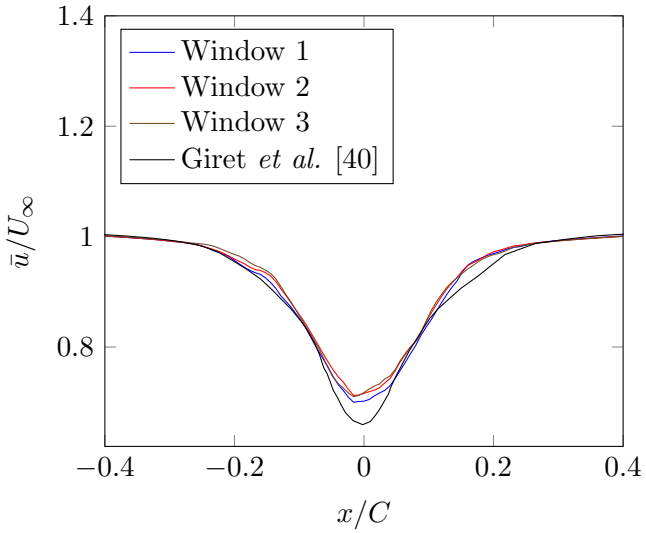


Figure 5.2.8: Convergence of three averaging windows. Window 1 corresponds to averaged u ($\bar{u}$), from $t_{nd} = 12.0$ to $t_{nd} = 16.0$. Window 2 corresponds to $t_{nd} = 14.0$ to $t_{nd} = 18.0$. Window 3 corresponds to $t_{nd} = 16.0$ to $t_{nd} = 20.0$.

Inspection of figure 5.2.8 reveals that $t_{nd} = 20.0$ can be considered an approximate point at which transient phenomena have ceased: The three curves corresponding to the three averaging windows have converged to a common shape at a reasonable degree. This implies statistically similar data were employed in the computation of the three curves. As a consequence, transient phenomena, that are expected to have exhibited statistical differences across each window can be thought of as having ceased. As a result, the data analysis presented below was performed on solution files from $t_{nd} = 20.0$ to $t_{nd} = 45.0$.

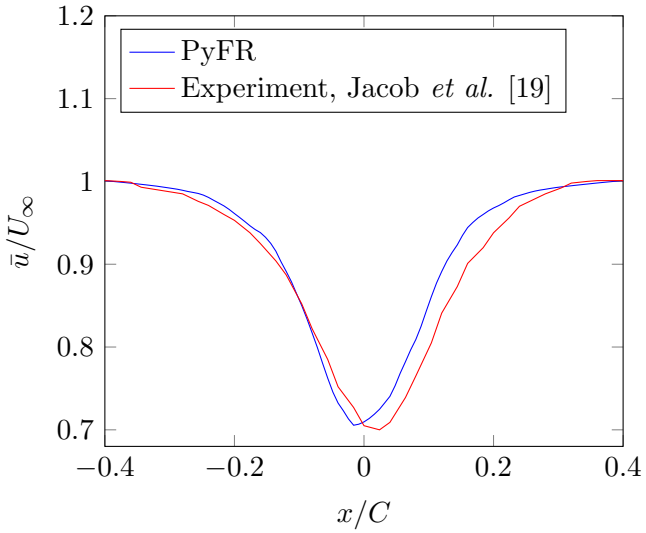


Figure 5.2.9: Time-averaged $x-$*component* of velocity u ($\bar{u}$) along line L_1 of figure 5.2.7. Averaging period equal to 25.0 time units ($t_{nd} = 20.0$ to $t_{nd} = 45.0$). Comparison with experiment.

Inspection of figure 5.2.9 reveals that PyFR agrees well with experiment in the wake of the cylinder. This can lead one to believe that the resolution of the three-dimensional computational grid is sufficient. However, if one considers a power spectrum for the estimation of the cylinder Strouhal number as per section 5.1, different behaviour is observed. Sampling instantaneous velocity u every 2×10^{-4} time units across from $t_{nd} = 20.0$ to $t_{nd} = 45.0$ on point P_1 of figure 5.2.7 results in the power spectrum of the following figure.

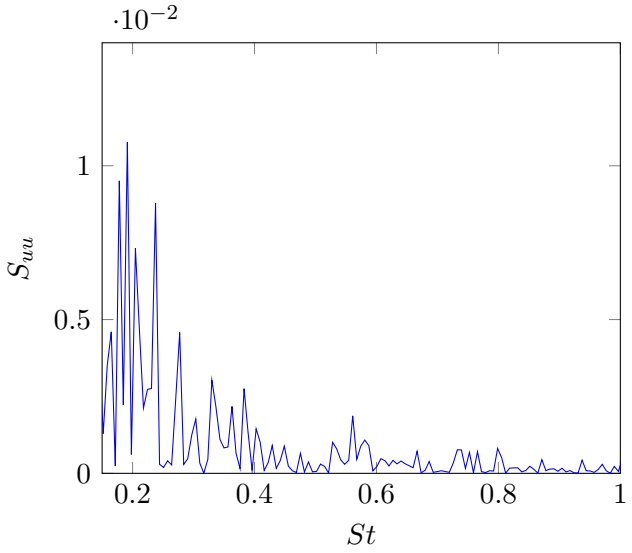


Figure 5.2.10: Power spectral density (PSD) of velocity u (S_{uu}) of preliminary Rod-Aerofoil simulation at point P_1 of figure 5.2.7.

Inspection of figure 5.2.10 reveals significantly different behaviour than the one depicted in figure 5.1.10. Whilst a peak at a Strouhal number of approximately 0.20 can be observed, the spectrum is also dominated by a series of unsteady phenomena at several different frequencies. A single powerful peak is not observed as per figure 4 (ii) of [19]. Visual inspection of figure 5.2.6 also reveals significant breakdown of large scale vortical structures that are apparent in the two-dimensional result of figure 5.1.3. As a result of the above observations, a hypothesis relating to the spanwise resolution can be put forward: The spanwise resolution is sufficiently low so that large scale vortices cannot be retained, breaking rapidly into smaller structures that contaminate the flow dynamics. This hypothesis is investigated in the following subsection.

5.2.2 Effect of Spanwise Resolution

Referring to figure 5.2.7, the aspect ratios of the hexahedral elements with respect to the spanwise direction (i.e. the ratio of the spanwise dimension of the element to the mean of its two $x-y$ dimensions) are as follows. If we denote the aspect ratio as AR, $AR_{Rg_1} \approx 70$, $AR_{Rg_2} \approx 1$, $AR_{Rg_3} \approx 70$. Subscript Rg_n denotes elements in the vicinity of the element at location Rg_n of figure 5.2.7.

One can investigate whether an aspect ratio of $AR \approx 70$ is sufficiently high

for rapid breakdown of large-scale vortical structures in the cylinder wake, via a three-dimensional, single-cylinder simulation of higher spanwise resolution.

A tenfold increase in the spanwise resolution was sought. In order to avoid excessive computational cost, the number of slices across the spanwise dimension was kept at five whilst the thickness of the domain was decreased by a factor of ten compared to the thickness of the rod-aerofoil domain of figure 5.2.5.

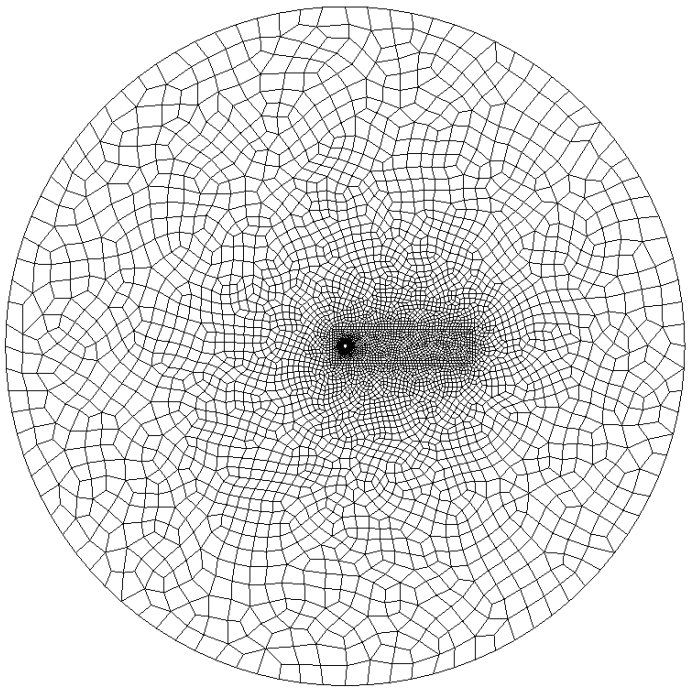


Figure 5.2.11: The $x - y$ plane of the computational grid of the 3D cylinder spanwise resolution study.

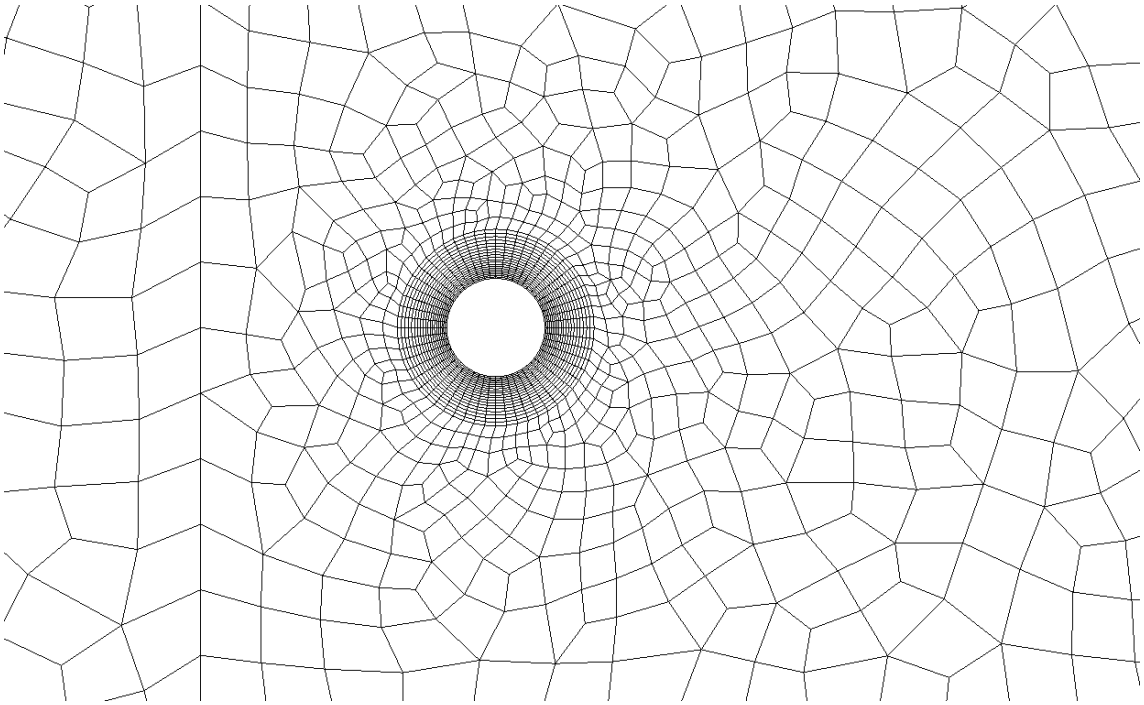


Figure 5.2.12: Zoom-in around the cylinder boundary layer of figure 5.2.10.

Inspection of figure 5.2.12 reveals a lower resolution of the boundary layer region than the one used in the previous study. The main reason behind this approach was a desire to investigate whether a tenfold increase in spanwise resolution is sufficient to retain a similar behaviour as the one depicted in figure 5.1.10 whilst employing a lower 2D boundary layer resolution than the one outlined through the findings summarised in table 5.1.1. As a result the boundary layer resolution was 21 radial and 60 circumferential elements, resulting in a mesh comprised of approximately $13,000$ hexahedra. The legitimacy of dropping the boundary layer resolution below the findings of the 2D cylinder studies would be tested in the Strouhal number prediction quality.

The numerical setup and the initialisation procedure of the simulation was almost identical to that presented in section 5.2.1. The sole difference was the absence of a need to stabilise the simulation via anti-aliasing. The simulation was performed using PyFR version 1.0.0. Freestream conditions were such that $Re_c = 48,000$.

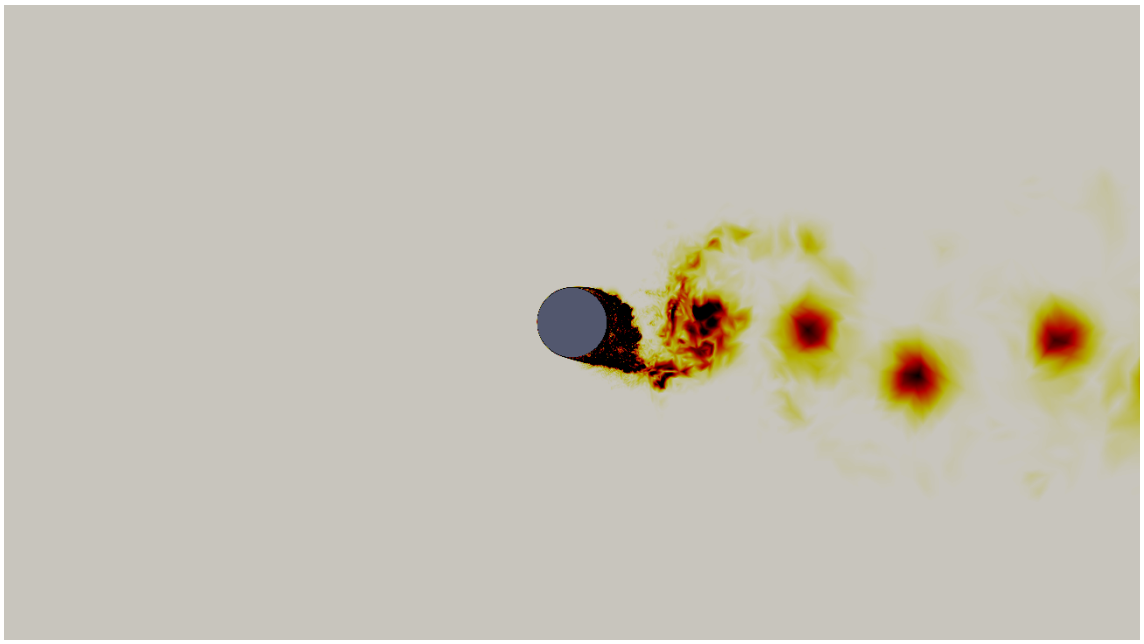


Figure 5.2.13: Instantaneous vorticity magnitude of the three-dimensional cylinder simulation at $t_{nd} = 20.0$.

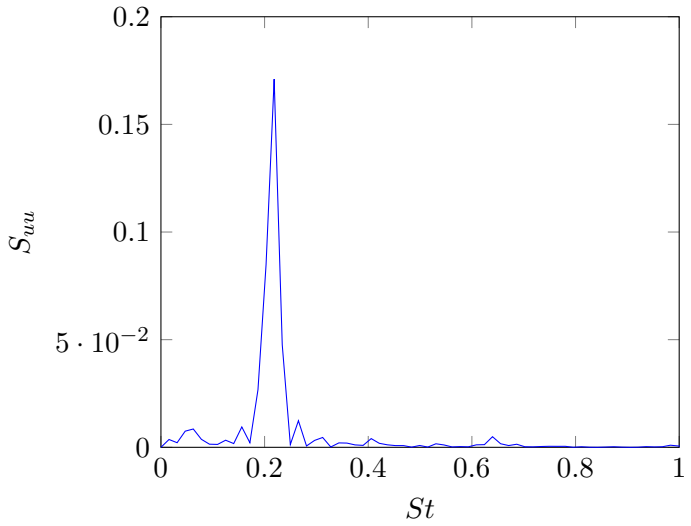


Figure 5.2.14: Power spectral density (PSD) of velocity u (S_{uu}) of three-dimensional cylinder simulation at point P_1 of figure 5.1.5. Data analysed from $t_{nd} = 20.0$ to $t_{nd} = 45.0$

.

A comparison of figures 5.2.14 and 5.2.10 reveals that the specification of a spanwise resolution such that $AR_{Rg_1} \approx 1$ has had a significant effect in terms of preserving the large-scale vortices. Further visual confirmation of this behaviour can be found by inspecting figure 5.2.13. As a result one can make two useful conclusions. Firstly, a spanwise resolution that is higher than the one employed in the preliminary

Rod-Aerofoil simulation is necessary in order for the large-scale vortical dynamics to be preserved in a manner required by the physics of the problem. Secondly, through careful re-calibration of the two-dimensional boundary layer resolution, one can arrive at a three-dimensional domain of lower size than the one that would be the direct result of the findings of section 5.1.

5.2.3 The Three-Dimensional Rod-Aerofoil Domain

As a consequence of the findings of the three-dimensional studies, this section will present the computational grid of the large-scale Rod-Aerofoil computation that will be analysed in Chapter 7.

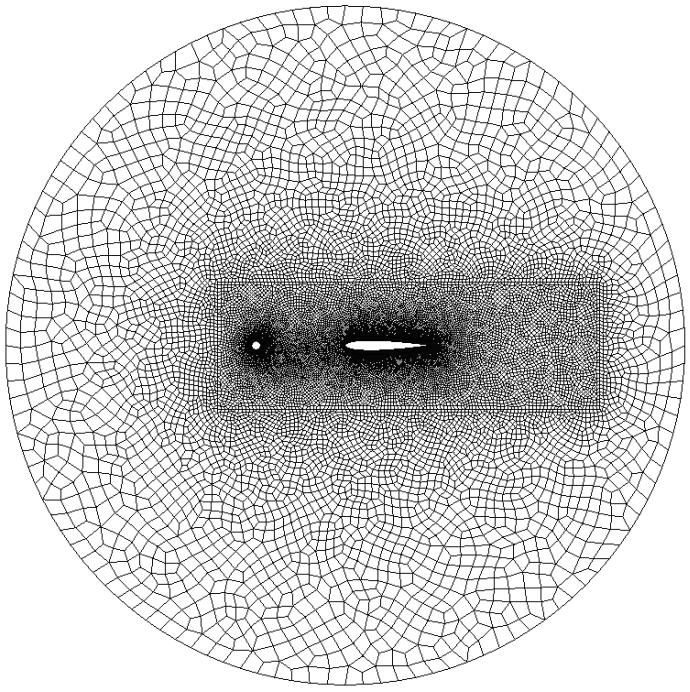


Figure 5.2.15: Computational grid of the large-scale Rod-Aerofoil simulation of Chapter 7, $x - y$ plane. The number of hexahedral elements was approximately $435,000$.

Figure 5.2.15 is depicts an $x-y$ cross-section of the computational grid. Through comparison with figure 5.2.1, the reader will note the halving of the radius of the far-filed circular boundary at $4C$. This was chosen so that the size of the computation could be reduced as much as possible, since the increased spanwise resolution would strongly impact the total number of elements.

The resolution of the boundary layer regions followed the three-dimensional

findings of the previous section, i.e. it employed a cylinder boundary of 64 circumferential and 21 radial elements with the aerofoil boundary having a nearly identical resolution.

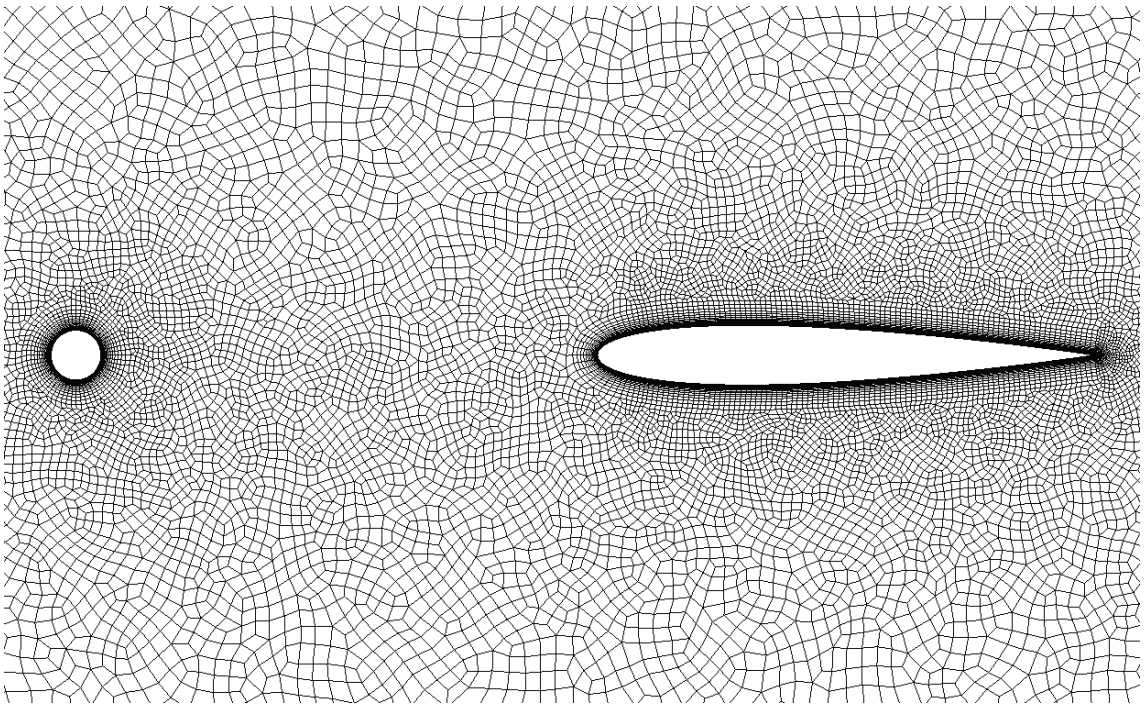


Figure 5.2.16: Zoom-in around the Rod-Aerofoil area of figure 5.2.15.

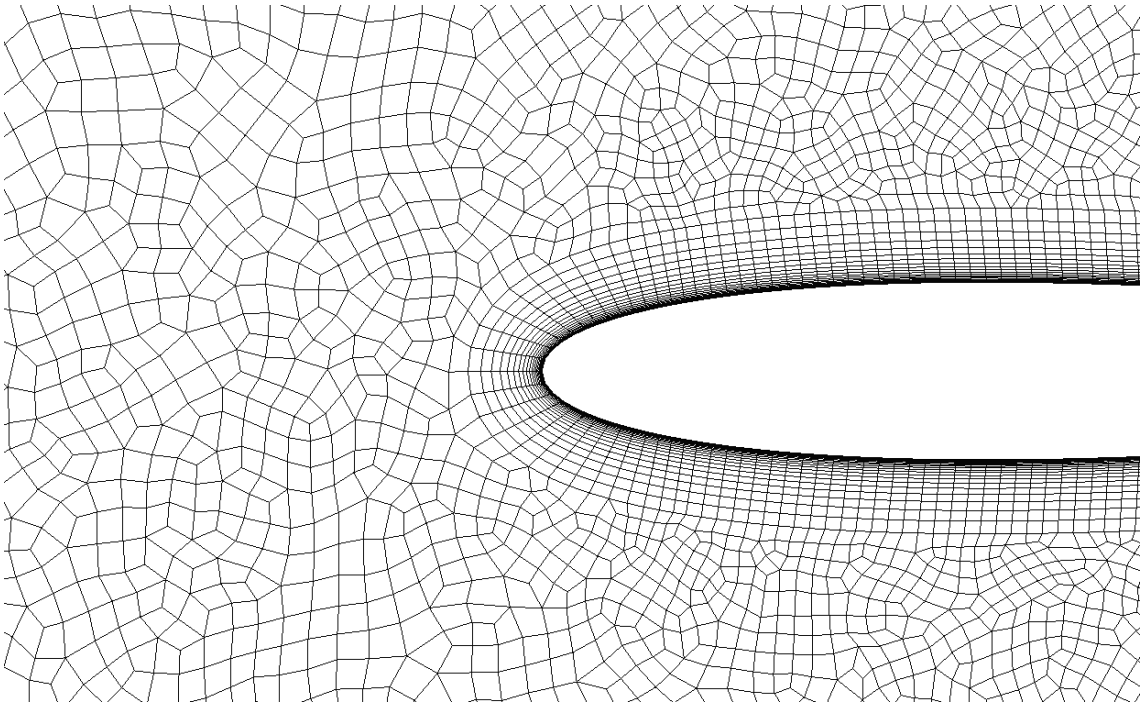


Figure 5.2.17: Zoom-in around the aerofoil leading edge of figure 5.2.15.

The spanwise dimension of the computational grid of the large-scale simulation

was $0.5C$. Combined with a division of 20 slices, this results in a twofold increase in the spanwise resolution. This increase in resolution can be appreciated via comparison of figures 5.2.5 and 5.2.18.

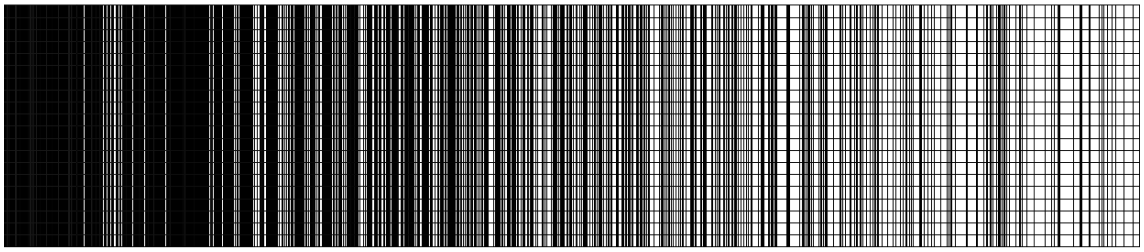


Figure 5.2.18: A $z-y$ cross-section of the mesh of figure 5.2.15 demonstrating the slices and thickness of the spanwise direction. Here, axis z is normal to the $x-y$ plane of figure 4.1.1.

As is the case with every mesh deployed for the purposes of the current thesis, all elements were cubically curved in order to facilitate a high-order computation of $P = 3$. The resultant number of hexahedral elements was approximately $435,000$. Combined with $P = 3$ for the polynomial order, the above grid results in an approximately $140,000,000$ degree-of-freedom simulation.

Chapter 6

Numerical Stability

One of the most important subjects of interest in current high-order CFD research is the stability of the simulation [5], [7], [8]. In the present work, simulation stability was addressed via two main strategies: Anti-aliasing and Entropy Stability schemes.

6.1 Aliasing-Induced Instabilities

Consider the following one-dimensional element Ω as per section 2.1.

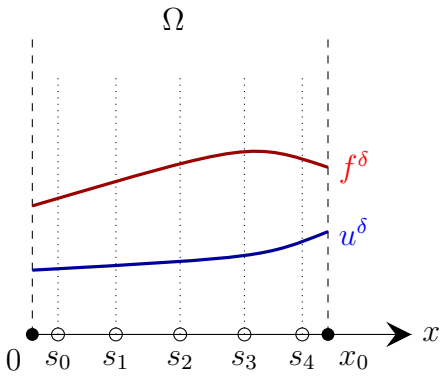


Figure 6.1.1: Arbitrary one-dimensional element in physical space x, $[0, x_0]$, with solution u^δ and flux f^δ marked. The element contains five solution points s_n, $n = 0, 1, ..., 4$.

We assume that flux f of figure 6.1.1 is a linear function of solution u

$$f(u) = au, \tag{6.1.1}$$

with a independent of u.

As detailed in Chapter 2, the FR procedure projects solution u into a polynomial space. Since projected flux f^δ is a linear function of projected solution u^δ, f^δ will also be a polynomial in the same space as u^δ.

However, if flux f is a non-linear function of solution u, as is the case with both the Euler and the Navier-Stokes equations, it is trivial to recognise that projection f^δ will generally not be a polynomial in the same space as projection u^δ. As a result, employing a point-wise, collocation projection of flux f^δ into a polynomial space via the known quantities at solution points s_n (as is the case with solution u^δ) will result in what is referred to as aliasing errors. Energy of unresolved modes will accumulate into the highest resolved modes of the simulation leading to instability. As investigated in detail by [45] and [46], this behaviour stems from the fact that inexact integration of the flux will overestimate the contribution of resolved modes since there is no explicit mechanism for the estimation of the contribution of the unresolved modes, i.e. the energy part associated with unresolved modes has to be accounted for by the resolved ones. Since the highest resolved modes are the closest to the unresolved ones, the weighting of the integration procedure will tend to overestimate their contribution.

One way of addressing this issue involves ensuring that flux f^δ is computed in a more accurate manner. At some additional computational cost, flux f^δ can be more accurately projected in the underlying polynomial space through a suitable Galerkin procedure outlined below.

$$f^\delta(u^\delta) \approx \sum_{i=0}^{m} \phi_i L_i. \tag{6.1.2}$$

In equation 6.1.2, f^δ is approximated via the set of $m+1$ Legendre polynomials L_i. We can now write the integration procedure for the calculation of ϕ_j.

$$\phi_i = \int_0^{x_0} f^\delta(u^\delta) L_i dx. \tag{6.1.3}$$

If the integral of expression 6.1.3 is computed exactly, then energies ϕ_i will correspond to the exact contribution of each mode L_i. Consequently, one seeks to perform the integration in 6.1.3 as accurately as possible. The integration can be computed numerically through a suitable quadrature operation applied on a set of quadrature points along x. If we define a set of quadrature points q_j, illustrated along

axis x in figure 6.1.1 and being generally different in size and distribution to the set of solution points, can be used to perform the quadrature rule for the estimation of 6.1.3. An increase in the accuracy of the quadrature for the approximation of 6.1.3 will generally result in an increase in the accuracy at which resolved energy modes are represented [45], [46]. The accuracy at which the integration of 6.1.3 can be performed is controlled by the size and type of the set of quadrature points specified. As a result, an increase in the quadrature degree is expected to ultimately result in an increase in stability, albeit at an increase of computational cost. For an one-dimensional consideration, and for quadrature points of the Gauss-Legendre type, the maximum polynomial degree that can be exactly integrated by the quadrature rule is $Q_d = 2q_p - 1$, with q_p being the number of quadrature points.

PyFR incorporates an anti-aliasing quadrature formula via user-specification of the type of distribution and number of quadrature points inside each element. These quadrature points are used in order to calculate the three-dimensional equivalent of 6.1.3 and 6.1.4 for the projection of the transformed flux inside an element as well as for the transformed flux at element interfaces.

The flow around the rod-aerofoil configuration is characterised by dynamics the under-resolution of which can result in severe instabilities. Of particularly challenging nature is the phenomenon of vortical structures generated in the cylinder wake impinging on the aerofoil surfaces and moving into domain regions of lower spatial resolution. Consequently one can expect aliasing-induced errors to be present.

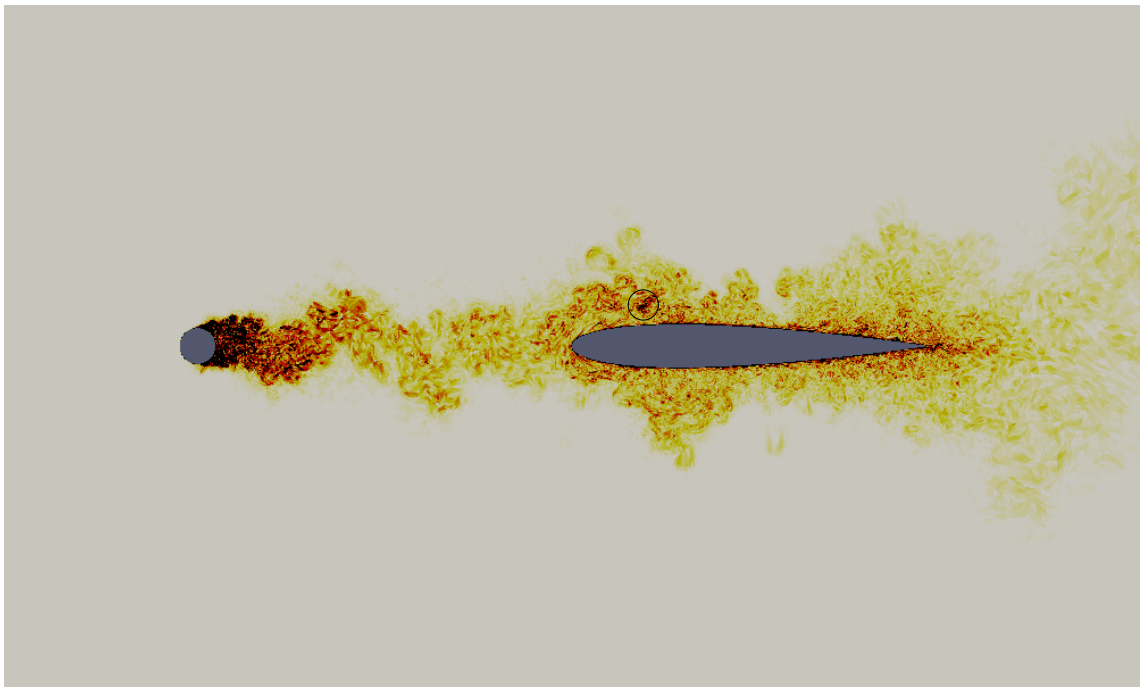


Figure 6.1.2: Instantaneous vorticity magnitude of the preliminary Rod-Aerofoil simulation. Snapshot at non-dimensional time $t_{nd} \approx 17.335$, approximately 0.002 time-units prior to simulation blowing-up. Circled is the vortical region that resulted in instability. Flow quantities such as velocity are at this stage rapidly growing towards infinity ($NaNs$). A desired consequence of using anti-aliasing would be the alleviation of the aforementioned phenomenon inside the circled region.

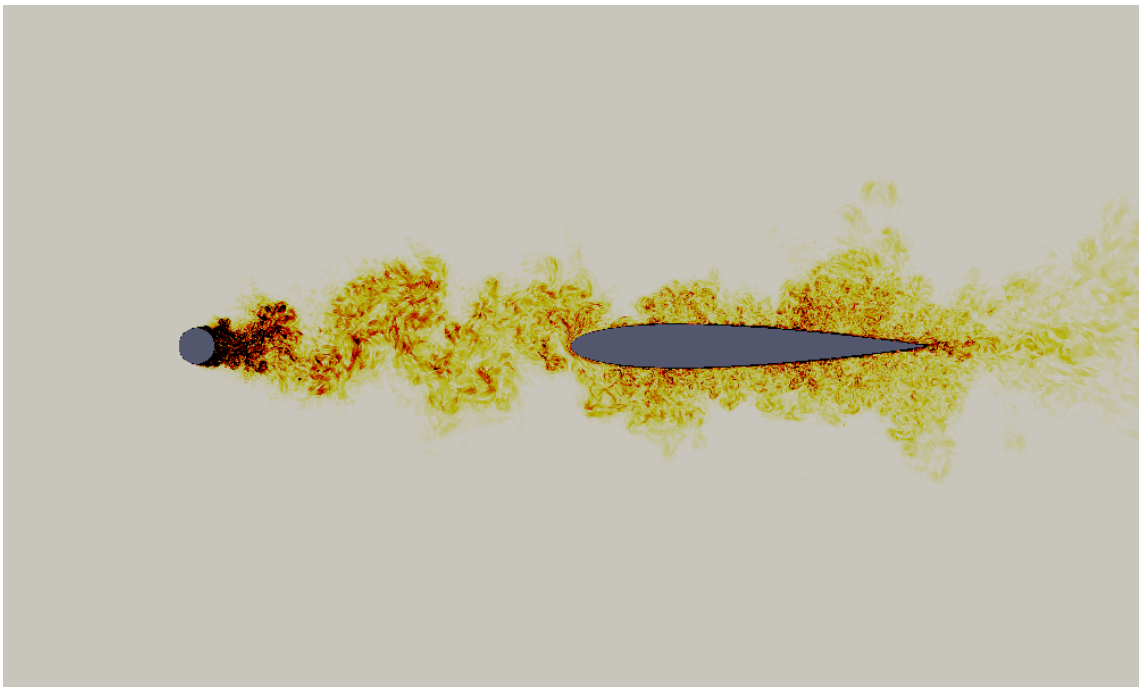


Figure 6.1.3: Instantaneous vorticity magnitude of the preliminary Rod-Aerofoil simulation with use of anti-aliasing. Snapshot at non-dimensional time $t_{nd} \approx 17.335$, approximately identical to figure 6.1.2. Note the suppression of the intensity of structures on the same colour scale, implying no immediate onset of instability.

In the Rod-Aerofoil simulations undertaken for the current thesis anti-aliasing was employed for both the volume and the surface fluxes inside each hexahedral element. The quadrature degree was $Q_d = 11$ and the quadrature points were of the Gauss-Legendre type. As a result, the number of quadrature points along a single dimension was $q_p = (Q_d + 1)/2 = 6 > 4 = n + 1$, with $n + 1$ the number of solution points along a single dimension for an FR polynomial order of $P = 3$. As a reference, the deployment of anti-aliasing of this degree resulted in approximately double the computational cost on 20 Nvidia M2090 GPUs for the preliminary Rod-Aerofoil simulation ($\sim 90,000,000$ degrees-of-freedom).

Figures 6.1.2 and 6.1.3 provide an illustration of the performance of anti-aliasing: Whilst retaining identical computational grid and simulation setup, it was possible to alleviate the problem of vortical regions that resulted into blow-ups.

As mentioned in previous sections, the choice of $Q_d = 11$ was motivated via two primary considerations. First, there was a need for the simulation to be stabilised. Second, there was a desire for the computational cost to be kept at the lowest levels possible. Anti-aliasing was tested with $Q_d = 9$ as well as $Q_d = 13$. It was found that choosing $Q_d = 9$ did not alleviate the problem of stability. Choosing $Q_d = 13$ did ensure stability (as one would expect since $Q_d = 11$ did ensure stability) whilst approximately requiring a further 20% increase in computational cost compared to the case of $Q_d = 11$. As a consequence, it was decided that $Q_d = 11$ forms a reliable and computationally realistic basis for the rod-aerofoil simulations with PyFR. Since it was found that utilising two extra quadrature points per dimension (compared to the number of solution points) for the over-integration was adequate, future simulations utilising other spatial FR orders can attempt to use this technique of $q_p = s_n + 2$ (with q_p, s_n the number of quadrature points and number of solution points per direction respectively) as a baseline. The aforementioned approach should be attempted if hexahedral elements are used. In the case of different element types, the above approach is not necessarily valid and a suitable study must be performed in order to arrive to adequate quadrature degrees.

6.2 Entropy Stability

Whilst the suppression of aliasing-induced instabilities proved necessary for the completion of simulations presented in the present work, more powerful instances of local instability were not able to be overcome for grid resolutions and quadrature degrees that resulted in simulations of manageable computational cost. More specifically, challenges that arose during the main study that will be presented in chapter 7, necessitated the use of a different approach, in parallel with anti-aliasing.

This approach came in the form of entropy stability. Part of the author's research towards a Ph.D. involved the implementation, into PyFR, of a solution limiter and a CFL number controller based in the work of Lv *et al.* [47]. The implementation was added to PyFR version 1.4.0 and it was extensively used for the results of Chapters 7 and 8. Furthermore, the implementation was achieved through collaboration in the research group.

A summary of the procedure of Lv *et al.* will now be presented. The authors sought to derive a DG scheme that mathematically guarantees that the solution is bounded by a reference entropy across every time-step of a high-order simulation of the Euler governing system. For the thermodynamic entropy s, pressure p, density ρ the following can be written.

$$s = \ln(p) - \gamma \ln(\rho) + s^0. \tag{6.2.1}$$

with s^0 being a reference entropy and γ being the ratio of the specific heats of the fluid.

A suitable procedure is needed such that, inside element e of the domain, the adjusted (limited) solution U_e^L will obey the following.

$$s(U_e^L(t)) \geq s^0(t). \tag{6.2.2}$$

i.e., the entropy of solution $U_e^L(t)$ should always be greater or equal than a reference (most usually freestream and generally dependant on time) entropy $s^0(t)$. The reference entropy for a simulation without discontinuities is set as the thermodynamic absolute entropy of the freestream flow (ideal gas).

The first stage of the algorithm involves the estimation for every element of

a CFL number that will constrain the time-step of the explicit scheme. This CFL number is denoted here as CFL^{EB}. Depending on the type of element and the order of the DG scheme, the value of CFL^{EB} can be analytically derived through the work of Lv *et al.*, the results of which have been summarised in table 1 of page 728 of [47]. Following the specification of a suitable CFL^{EB}, the algorithm calculates for each element e the following characteristic length L_e.

$$L_e = \min \frac{V_e}{\mid J_I^\theta \mid}. \tag{6.2.3}$$

i.e., L_e is the minimum of the ratios of the physical volume of the element V_e to the determinant of the surface Jacobian for each of its interfaces J_k^θ. Here I denotes the current element interface.

It is important to note here that computation of L_e does not depend on anything other than the geometrical properties of the computational grid and its quadrature points. As a result, PyFR computes L_e during start-up, resulting in no effect on wall-clock time during the actual simulation.

Once L_e has been determined for every element of the domain, the algorithm needs to perform the first time-dependant operation. This is the computation of a characteristic velocity λ for each element and each time step of the simulation.

$$\lambda = \max[v(U_{e,n})], \tag{6.2.4}$$

with $v_{e,n} = \mid u_{e,n} \mid + c_{e,n}$ being the characteristic speed of the flow at a solution point n of element e, u being the velocity vector and c being the speed of sound. The maximum permissible time-step for each element e is then estimated as

$$\delta t = \frac{CFL^{EB} L_e}{2\lambda}. \tag{6.2.5}$$

The simulation should then advance in time via a global time step δt_{min} which is the minimum of all δt of all elements.

The PyFR mako kernel responsible for the calculation of λ and δt for each element is presented in the figure 6.2.1

```
1   # -*- coding: utf-8 -*-
2   <%inherit file='base'/>
3   <%namespace module='pyfr.backends.base.makoutil' name='pyfr'/>
4
5   <%pyfr:kernel name='timestep' ndim='1'
6                 u='in fpdtype_t[${str(nvars)}]'
7                 le='in fpdtype_t'
8                 dt='out fpdtype_t'
9                 cfl_eb='scalar fpdtype_t'>
10
11      fpdtype_t rho, v[${ndims}], p, invrho, E, rhov[${ndims}], a, magu, lamda;
12      magu = 0.0;
13  % for j in range(nupts):
14      rho = u[${j*nvars}];
15      invrho = 1.0/u[${j*nvars}];
16      E = u[${(j+1)*nvars - 1}];
17  % for i in range(ndims):
18      rhov[${i}] = u[${i + 1 + j*nvars}];
19      v[${i}] = invrho*rhov[${i}];
20  % endfor
21      p = ${c['gamma'] - 1}*(E - 0.5*invrho*${pyfr.dot('rhov[{i}]', i=ndims)});
22      a = sqrt(${c['gamma']}*p/rho);
23      magu = sqrt(${pyfr.dot('v[{i}]', i=ndims)});
24      lamda = max(lamda, a + magu);
25  % endfor
26      dt = 0.5*cfl_eb*le/lamda;
27  </%pyfr:kernel>
```

Figure 6.2.1: PyFR pointwise mako kernel for the computation of the permissible time-step for each iteration. Lines 13-21 compute the primitive variables needed for expression 6.2.5. Lines 22 and 23 compute the needed characteristics speeds. Quantity λ is then computed in line 24 before line 26 returns the time step. As mentioned in chapter 3, this kernel is platform-independent and the low-level code that implements it generated at runtime.

If U_e^k is the solution of element e at current time-step k, then the form of the solution at time-step $k+1$ is

$$U_e^{k+1} = U_e^k + \delta t_{min}^k R^k, \tag{6.2.6}$$

with R^k being the residual of the time-integration scheme.

In parallel to the computation of a suitable time-step, the algorithm constructs a suitable limiter that bounds the solution U_e^{k+1} following each iteration k according to a user-specified reference entropy s_0. The limited $U_e^{L,k+1}$ is then used as the RHS in equation 6.2.6 for the advancement to U_e^{k+2} on which the limiting operation will be applied anew and so on.

The limiting operation at each time step and for each element e involves the calculation of two parameters. The first parameter is τ as defined below.

$$\tau_e^k = \min[0, \min[p(U_{e,n}^k) - e^{s^0 \rho^\gamma(U_{e,n}^k)}]], \tag{6.2.7}$$

with $U_{e,n}^k$ being the solution at solution point n of element e at time-step k.

The PyFR mako kernel responsible for the calculation of quantity $\tau_{e,n}^k = [p(U_{e,n}^k) - e^{s^0 \rho^\gamma(U_{e,n}^k)}]$ for each solution point is presented in the figure 6.2.2.

```
1  # -*- coding: utf-8 -*-
2  <%inherit file='base'/>
3  <%namespace module='pyfr.backends.base.makoutil' name='pyfr'/>
4
5  <%pyfr:kernel name='tau' ndim='2'
6                u='in fpdtype_t[${str(nvars)}]'
7                t='out fpdtype_t'>
8      // Compute pressure and density
9      fpdtype_t rho = u[0];
10     fpdtype_t v[${ndims}], p;
11     fpdtype_t invrho = 1.0/u[0], E = u[${nvars - 1}];
12     fpdtype_t rhov[${ndims}];
13 % for i in range(ndims):
14     rhov[${i}] = u[${i + 1}];
15     v[${i}] = invrho*rhov[${i}];
16 % endfor
17     p = ${c['gamma'] - 1}*(E - 0.5*invrho*${pyfr.dot('rhov[{i}]', i=ndims)});
18     t = p - exp(${s0})*pow(rho,${c['gamma']});
19 </%pyfr:kernel>
20
```

Figure 6.2.2: PyFR pointwise mako kernel for the computation of $\tau_{e,n}^k$ for each iteration and for each element. After suitable determination of the flow variables by lines 13-17, $\tau_{e,n}^k$ is returned by line 18.

Once τ has been computed one needs to compute the second parameter, ϵ.

$$\epsilon_e^k = \frac{\tau_e^k}{\tau_e^k - [p(\bar{U}_e^k) - e^{s^0 \rho^\gamma(\bar{U}_e^k)}]}, \tag{6.2.8}$$

with $\bar{U}_e^k$ being the volume-averaged solution of element e at time-step k.

The PyFR mako kernel responsible for the calculation of quantity ϵ for each element is presented in the figure below.

```
1  # -*- coding: utf-8 -*-
2  <%inherit file='base'/>
3  <%namespace module='pyfr.backends.base.makoutil' name='pyfr'/>
4
5  <%pyfr:kernel name='epsilon' ndim='1'
6                t='in fpdtype_t[${str(nupts)}]'
7                u_bar='in fpdtype_t[${str(nvars)}]'
8                epsilon='out fpdtype_t'>
9      // Compute the minimum
10     fpdtype_t taubar = 0.0;
11 % for i in range(nupts):
12       taubar = min(taubar, t[${i}]);
13 % endfor
14     fpdtype_t rho = u_bar[0];
15     fpdtype_t v[${ndims}], p;
16     fpdtype_t invrho = 1.0/u_bar[0], E = u_bar[${nvars - 1}];
17     fpdtype_t rhov[${ndims}];
18 % for i in range(ndims):
19     rhov[${i}] = u_bar[${i + 1}];
20     v[${i}] = invrho*rhov[${i}];
21 % endfor
22     p = ${c['gamma'] - 1}*(E - 0.5*invrho*${pyfr.dot('rhov[{i}]', i=ndims)});
23     epsilon = taubar/(taubar - (p - exp(${s0})*pow(rho,${c['gamma']})));
24 </%pyfr:kernel>
25
```

Figure 6.2.3: PyFR pointwise mako kernel for the computation of ϵ for each iteration and for each element. This kernel accepts as input the output of the kernel depicted in 6.2.2, τ. Line 12 will compute the minimum of all τ as required by the procedure. Lines 18 to 22 will then compute the needed flow variables that will then result in ϵ through line 23.

The limiting operation is then applied as

$$U_e^{L,k} = U_e^k + \epsilon(\bar{U}_e^k - U_e^k). \tag{6.2.9}$$

6.2.1 Validation of the Implementation

The utility of the implementation of the work of Lv *et al.* [47] was initially assessed via the simulation of a two-dimensional, supersonic ($M > 1.0$) oblique shock case.

The altered PyFR version 1.4.0 incorporating the implementation was used on a two-dimensional grid of triangular elements representing a straight channel incorporating an angled section that gives rise to oblique shock waves. The simple, supersonic inflow and outflow boundary conditions presented in Chapter 3 were employed with $M = 2.0$. The governing system solved was the Euler equations. The polynomial degree of the DG scheme was $P = 3$ with no anti-aliasing employed and time integration was achieved through the RK45 scheme.

The stabilisation of the resultant shock waves from a cold start (initial conditions equal to the freestream conditions) was sought via a limiting reference entropy that was equal to the freestream entropy s_∞.

$$s^0 = s_\infty = \ln(p_\infty) - \gamma \ln(\rho_\infty). \tag{6.2.10}$$

with p_∞ and ρ_∞ being the freestream pressure and density respectively.

It is noted here that the simulation failed at ~ 1.4 non-dimensional time units without the use of a suitable stabilising technique.

The test case was first run with an artificial-viscosity type limiter, the result of which at a steady state after 5.0 time units is depicted in figure 6.2.4. This limited the solution via the introduction of an artificial viscosity factor that filtered the solution and dampened spurious phenomena in order for the shock waves to be captured [51]. The maximum value for the artificial viscosity was set at 0.01.

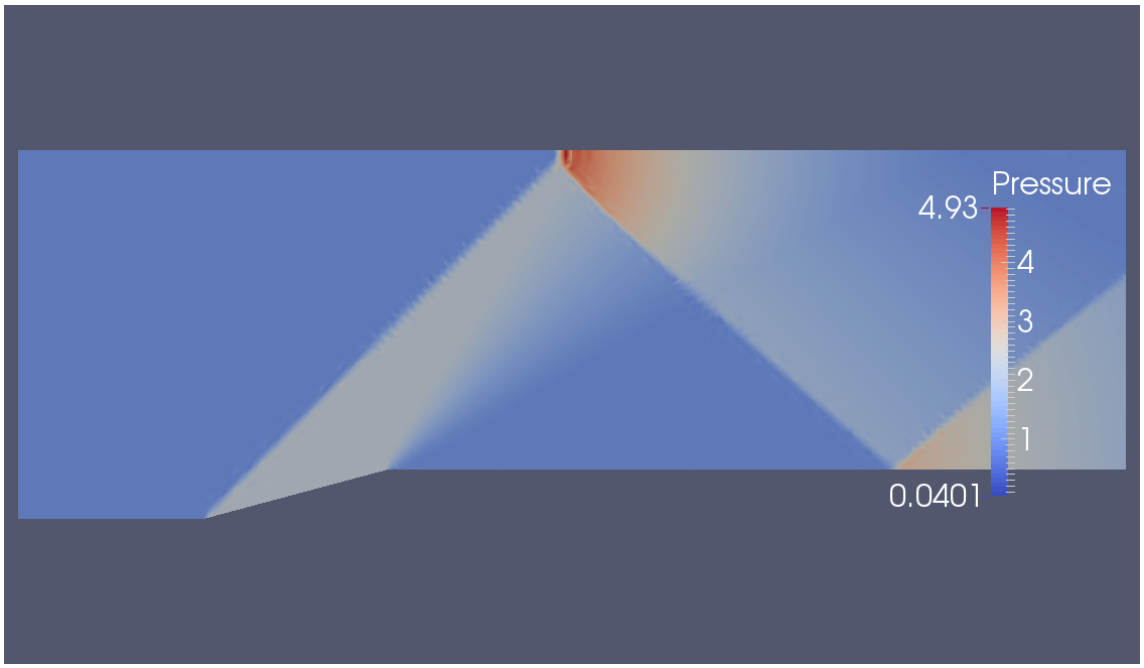


Figure 6.2.4: Non-dimensional pressure distribution at $t_{nd} = 5.0$ of the two-dimensional supersonic test case. Artificial-viscosity limiter used. Steady state reached, with oblique shock waves evident.

Inspection of figure 6.2.4 reveals shock waves clearly defined. The artificial viscosity method succeeds in stabilising the simulation and capturing the shock waves, which was its primary design purpose.

One can now asses the performance of the entropy limiter implementation relative to the performance of the artificial viscosity limiter. The result that is obtained via the use of the entropy limiter is depicted in the following figure.

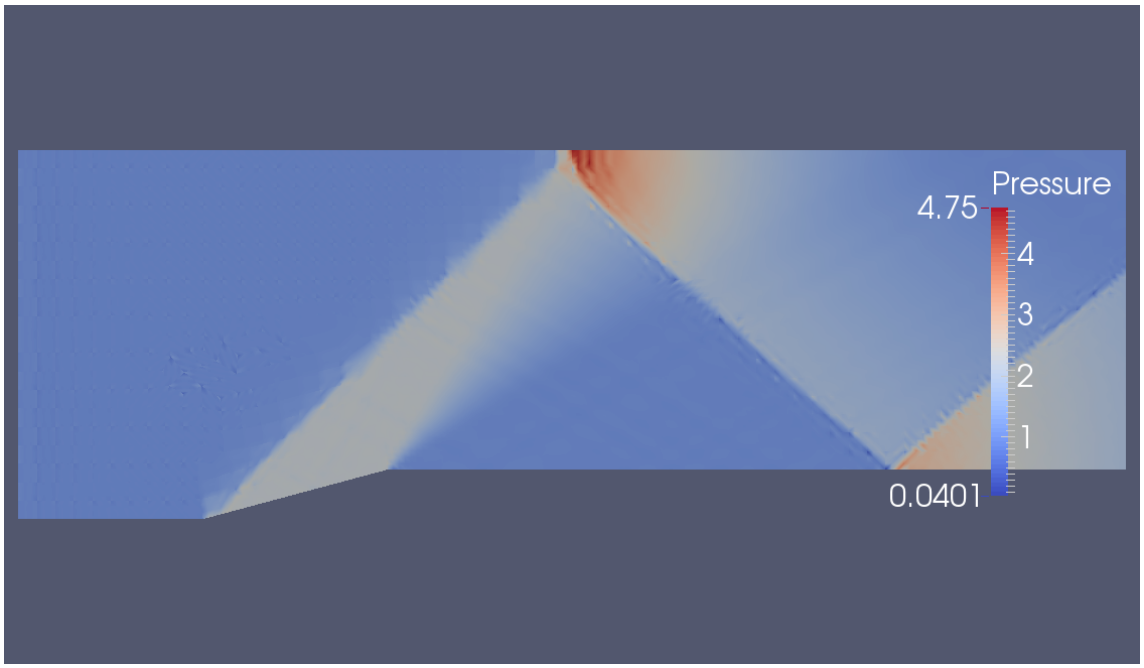


Figure 6.2.5: Pressure distribution at $t_{nd} = 5.0$ of the two-dimensional supersonic test case. Entropy limiter used. Steady state reached, with oblique shock waves evident.

Inspection of figure 6.2.5 visually confirms that the entropy stability limiter is successful in capturing the shock waves. A loss of definition in the outline of the shock waves however implies that the method is not as suited to shock-capturing as is the artificial viscosity limiter, for this type of problem. However, it is important to note that the entropy limiter is not specifically designed for the capturing of shock waves as is the case with the artificial viscosity limiter [47]. The fact that it succeeds in stabilising the simulation does however indicate that, in principle, the approach is potentially capable of ensuring the completion of simulations generally characterised by phenomena prone to instability.

As a result of the above findings, the entropy stability implementation was extensively used in the large-scale simulation of the following chapters. By utilising an extract of this simulation, a part the limiting procedure can be visualised. The following figure depicts an instantaneous snapshot of the large-scale simulation.

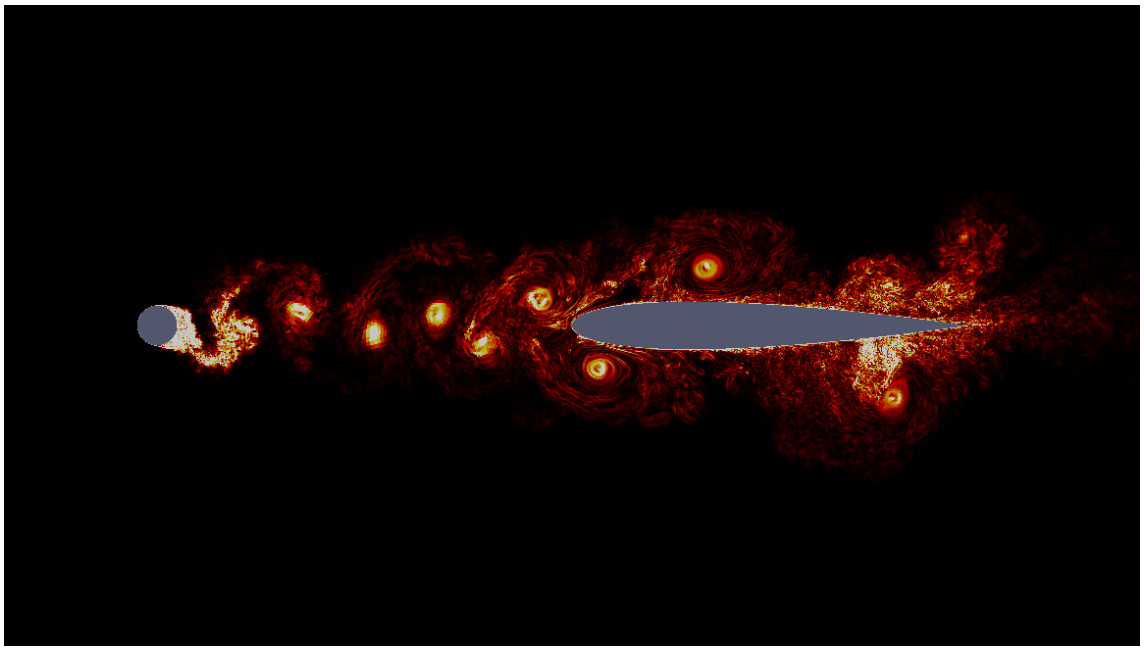


Figure 6.2.6: Instantaneous vorticity magnitude of the large-scale simulation of Chapter 7. Note the highly vortical structures that impinge into the aerofoil leading edge, often resulting in instabilities that anti-aliasing of quadrature degree $Q_d = 11$ is not able to alleviate.

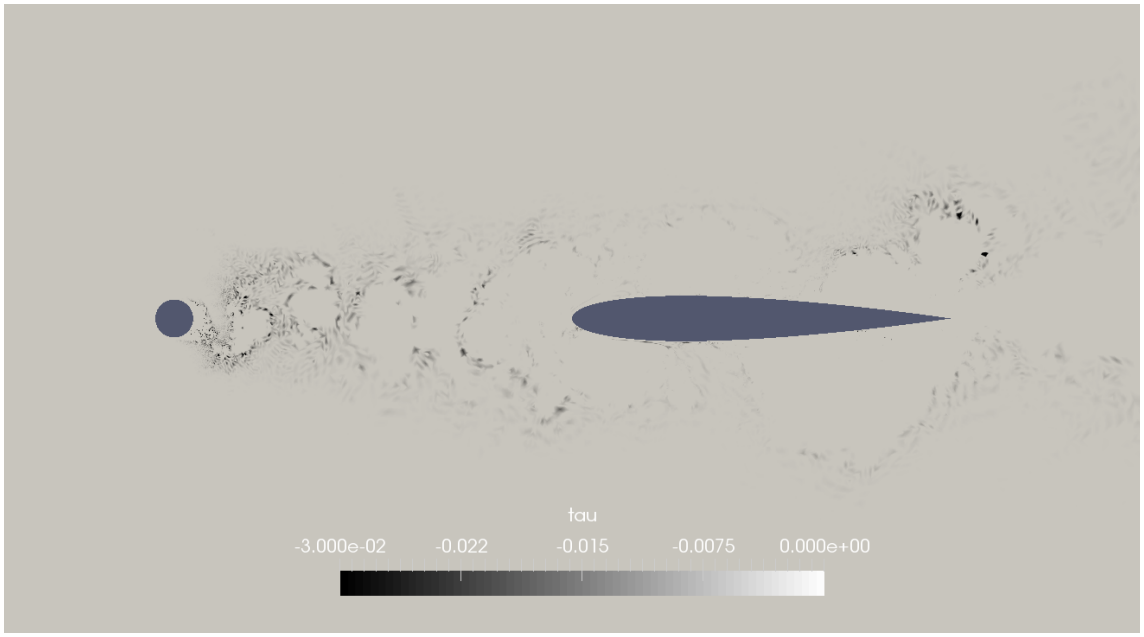


Figure 6.2.7: Distribution of quantity τ of the entropy limiter of instantaneous condition depicted in figure 6.2.6.

Inspection of figures 6.2.6 and 6.2.7 reveals the degree at which the entropy limiter is employed according to the severity of the flow dynamics at each solution

point: Areas with the strongest tendency to instability prompt a calculation of higher τ and, as a result, a more severe limiting operation. Furthermore, the reader can appreciate the highly selective nature of the approach, which is desired. The filtering of parts of the solution domain presenting no instability issues is conveniently avoided. Since the limiter essentially reduces the spatial accuracy of the scheme, its avoidance on areas that it is not needed preserves the original spatial accuracy.

The utility of the entropy limiter was important for the large-scale simulation of the rod-aerofoil problem, as its successful stabilisation would potentially only be possible by the use of higher quadrature order anti-aliasing. Anti-aliasing of order $Q_d = 11$ was employed but resorting to $Q_d > 11$ would result in significant computational expense. To the knowledge of the author, the present work constitutes the first attempt at simulating the Rod-Aerofoil case via high order FR schemes, in the absence of a sub-grid scale turbulence model whilst utilising stabilisation techniques such as the entropy stability implementation. Chapter 7 will present the findings of this work.

Chapter 7

Simulation of The Rod-Aerofoil Configuration

7.1 Computational Setup

The large-scale, i.e. highest-resolution, simulation of the rod-aerofoil problem was performed with PyFR version 1.4.0 [23], deployed on 200 Nvidia P100 GPUs of the Piz Daint cluster of the Swiss National Supercomputing Centre (CSCS). PyFR 1.4.0 was augmented via the inclusion of the entropy-stability scheme as outlined in Chapter 6.

The computational grid depicted in figure 7.1.1 had the following main characteristics.

- Structured hexahedral elements for the boundary layer regions with the cylinder boundary region divided into 64 circumferential and 21 radial elements.
- Unstructured hexahedral elements in the $x - y$ plane for the remainder of the domain.
- Cylindrical (circular in the $x - y$ plane) outer boundary, with a radius of $4C$.
- Spanwise thickness of $0.5C$, divided into 20 slices.
- Cubically curved elements in order to facilitate a simulation with polynomial order of $P = 3$.

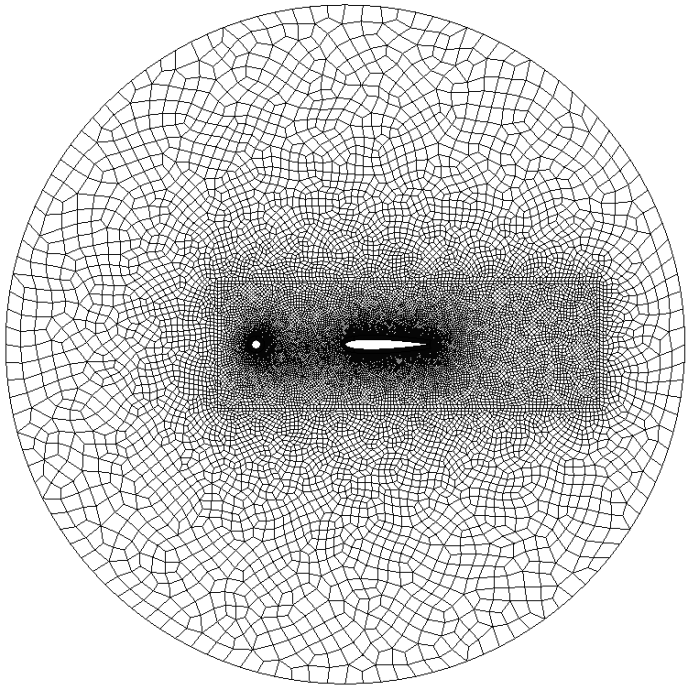


Figure 7.1.1: Computational grid of the large-scale Rod-Aerofoil simulation, $x-y$ plane. The number of tetrahedral elements (equal to the number of hexahedral elements per spanwise slice) is approximately $22,000$.

The correction functions of the FR procedure discussed in section 2.1.1 were such that the DG high-order scheme was recovered. The solution points were of the Gauss-Legendre type. For the main part of the simulation, that with polynomial order $P = 3$, volume and surface flux anti-aliasing was employed with a quadrature degree of 11. Time-stepping was performed via an RK45 stepper and all calculations were performed in double-precision.

Prior to a precise presentation of the phases of the simulation, a demonstration of strong scaling will be presented in order to demonstrate the efficiency of the code implementation for the present problem. Strong scaling refers to the wall-clock time reduction observed whilst the number of devices (i.e. GPUs in the current study) is increased for a problem of constant size. The strong scaling test was performed on Piz Daint's P100 GPUs for the computational grid and numerical setup presented above and utilising the polynomial order of $P = 3$. The result is shown in figure 7.1.2.

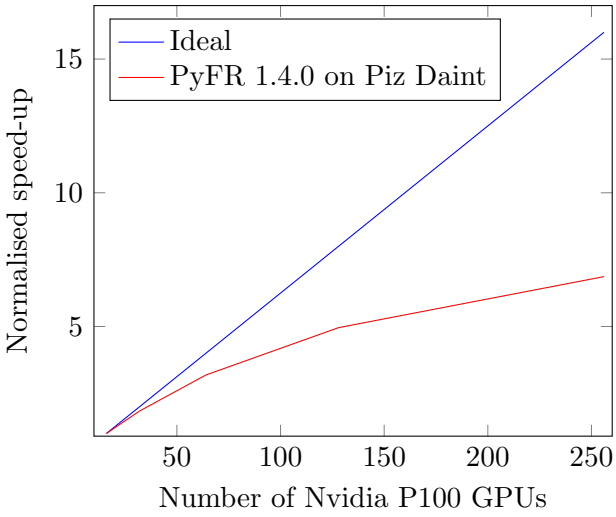


Figure 7.1.2: Strong scaling of PyFR on Piz Daint for the Rod-Aerofoil large-scale simulation. The speed-up is normalised by the performance on 16 GPUs, i.e. with wall-clock time equal to 1.0 for a constant number of iterations on 16 GPUs, the wall-clock time on 32 GPUs was reduced by a factor of ~ 1.84 (speed-up of ~ 1.84) for the same problem size and number of iterations. The red line was obtained via tests on 16, 32, 64, 128 and 256 GPUs. The time step size was kept constant at $\delta t = 2 \times 10^{-6}$.

.

When one inspects figure 7.1.2, a relatively large divergence from the ideal curve is noted. The approximate number of cuda cores on a single Nvidia P100 is $3,500$. The total number of elements in the simulation is approximately $435,000$. Therefore, at approximately 120 P100 gpus and upwards, the number of grid elements per cuda core falls lower than 1. At such a low and inefficient GPU loading, severe departure form ideal strong scaling is to be expected. This behaviour is clearly evident in the scaling plot.

In terms of wall-clock time, the approxiamtely $140,000,000$ degree-of-freedom, $P = 3$ simulation required approximately 6 hours on 200 of Piz Daint's Nvidia P100 GPUs for the completion of a single flow pass over the aerofoil chord with a constant non-dimensional time step size of $\delta t = 2 \times 10^{-6}$.

The simulation consisted of the following phases.

- Phase 1: Start of simulation from initial conditions equal to the freestream, non-dimensional conditions (cold startup): $u_{fs} = 1.0$, $v_{fs} = 0.0$, $u_{fs} = 1.0$, $\rho_{fs} = 6106.086$, $p_{fs} = 98900.0$. The Prandtl number was $Pr = 0.72$ and

the dynamic viscosity $\mu = 0.0127210125$. The ratio of specific heats was set as $\gamma = 1.4$.The non-dimensional freestream conditions as well as the Prandtl number and the dynamic viscosity were such that the aerofoil-chord defined Reynolds number Re_a was $Re = 480,000$ and the Mach number was $M = 0.21$. The simulation advanced by 8 non-dimensional time units, equal to 8 flow passes over an aerofoil chord. The polynomial order of FR scheme was $P = 1$. The non-dimensional time step size was kept constant at $\delta t = 10^{-5}$.

- Phase 2: The simulation was advanced by a further 8 non-dimensional time units with $P = 2$ and a constant time step size of $\delta t = 5 \times 10^{-6}$.

- Phase 3A: The simulation was advanced a further ~ 25 non-dimensional time units with $P = 3$ whilst utilising PyFR's adaptive time-stepping implementation described in section 3.2.1 [48], [49], [18]. Simulation instability was encountered at $t_{nd} \approx 41.3$.

- Phase 3B: The simulation is advanced a further ~ 7 non-dimensional time units with $P = 3$ whilst utilising the entropy-stability limiter and the entropy-stability time step size controller. Local instability was encountered at $t_{nd} \approx 48.5$.

- Phase 3C: The simulation was advanced by a further ~ 50 non-dimensional time units with $P = 3$ whilst utilising the entropy-stability limiter and a constant time step size of $\delta t = 2 \times 10^{-6}$. The simulation was able to reach $t_{nd} \approx 100.0$ without instabilities.

Phase 3C restarted the simulation from $t_{nd} \approx 48.0$. The reason for this was that the solution domain immediately prior to $t_{nd} \approx 48.5$ was contaminated by the imminent instability.

7.2 Results and Discussion

7.2.1 Overall Flow Features

Before the statistical flow quantities are presented, an overview of the qualitative flow characteristics are presented in this section.

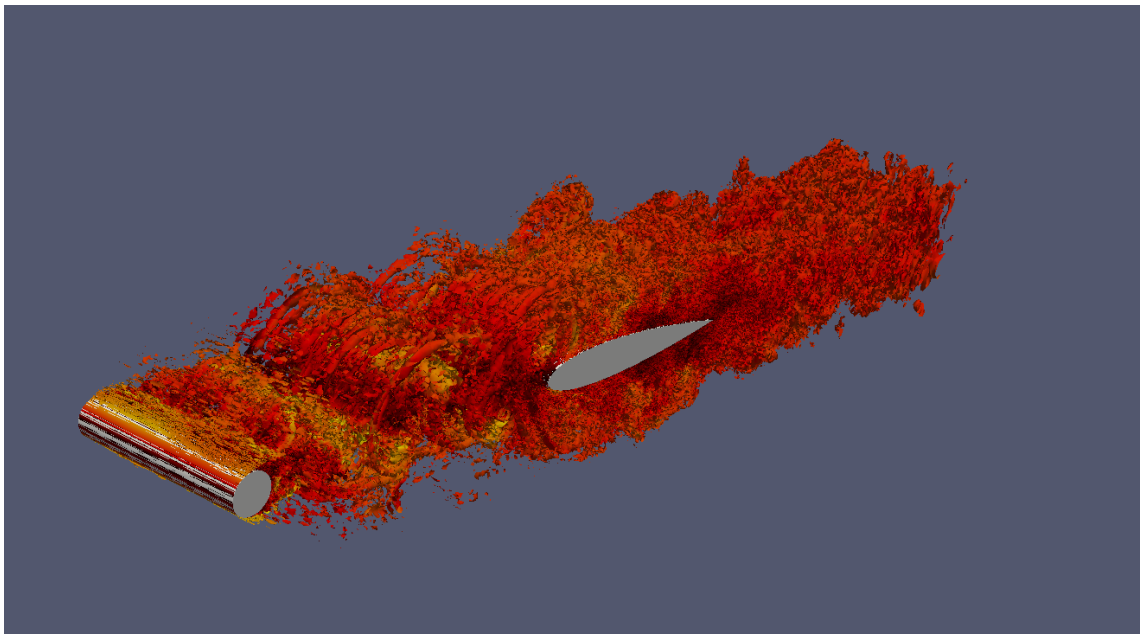


Figure 7.2.1: Instantaneous Q-criterion iso-surfaces with magnitude of 1.0. The latter, non-dimensional magnitude was chosen so as to allow the visualisation of flow structures of length scales comparable to the scales of the solid bodies. Colouring via velocity magnitude. Simulation time $t_{nd} = 100.0$.

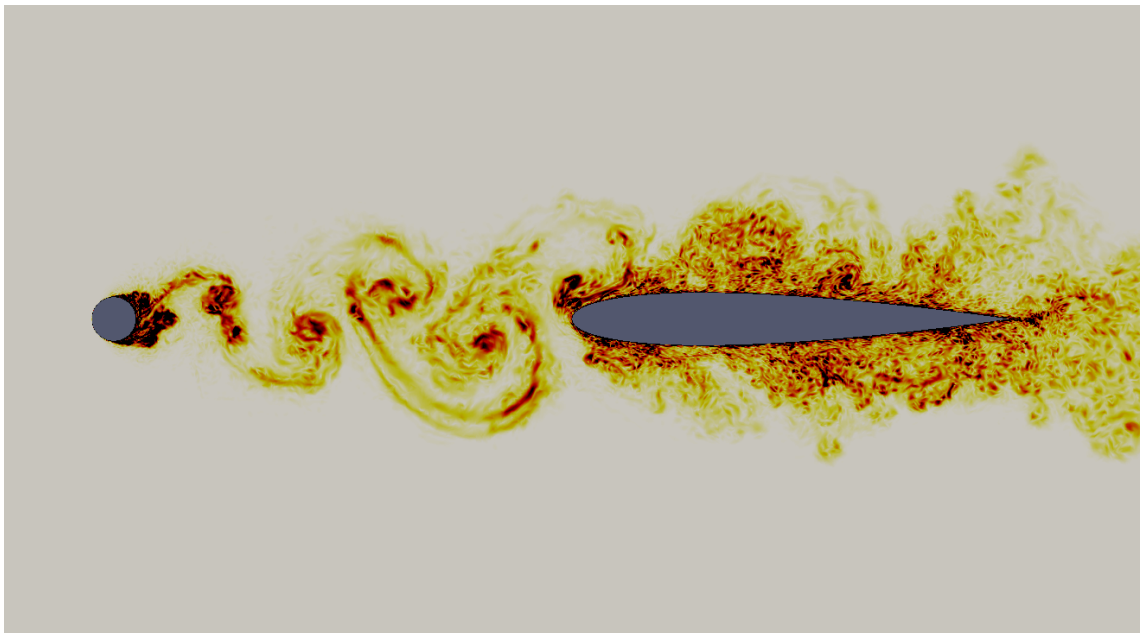


Figure 7.2.2: Instantaneous vorticity magnitude on plane normal to the spanwise axis and located in the middle of the spanwise thickness ($z = 0.25C$). Colour scale 1.0 to 100.0 non-dimensional units of vorticity (as velocity and length scales are non-dimensionalised via U_∞ and C.). Simulation time $t_{nd} = 100.0$.

Figures 7.2.1 and 7.2.2 illustrate the instantaneous flow condition at the non-dimensional time $t_{nd} = 100.0$, i.e. close to the end of the simulation. Time $t_{nd} = 100.0$ is also a snapshot within the range of statistically-settled flow dynamics.

Inspection of figures 7.2.1 and 7.2.2 allows two major observations. The first is that large-scale vortical structures contained in the wake of the cylinder are coherent until their impingement onto the aerofoil leading edge. The second is that impingement on the aerofoil leading edge results in a major break-down of the vortical structures, leading to an aerofoil wake populated by a larger range of turbulent scales.

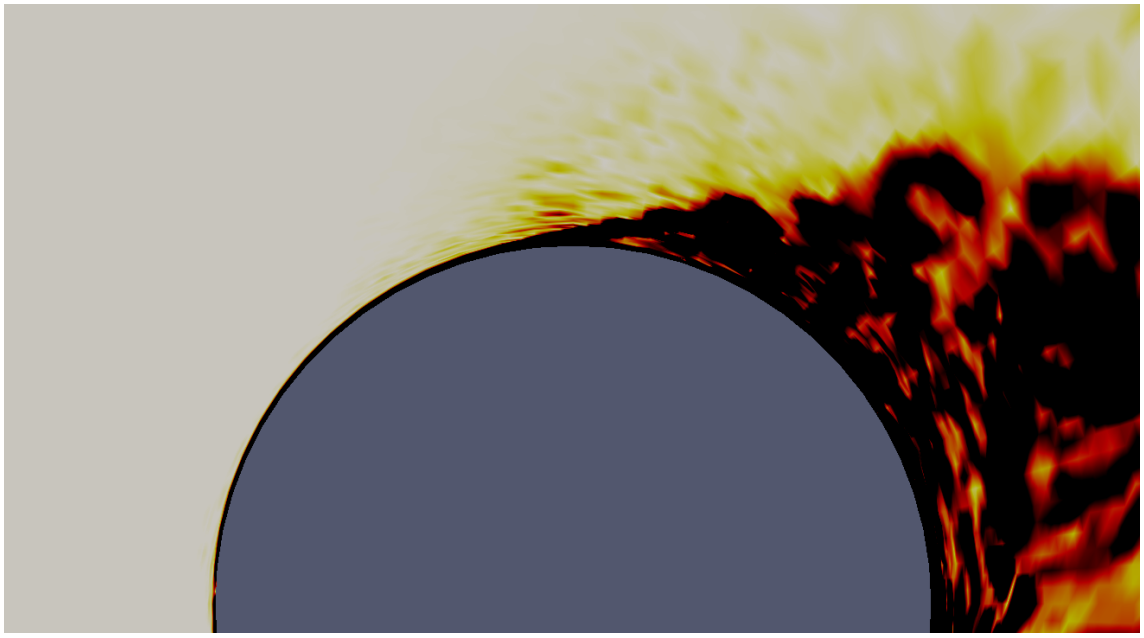


Figure 7.2.3: Zoom-in of image 7.2.3 around the cylinder boundary layer and separation point.

A magnified view of the separation point of the cylinder shown in figure 7.2.3 reveals the production of the flow structures. The thickness of the cylinder boundary layer in the vicinity of the separation point is $\sim 0.01d = 0.001C$. Due to a cylinder-diameter-defined Reynolds number of $Re_d = 48,000$, the relatively small thickness of the boundary layer (relative to the cylinder diameter) underlines the high vorticity expected. However, the onset of Kelvin–Helmholtz instability is not evident in figure 7.2.3. This fact points towards insufficient resolution of the flow dynamics and potentially suggests the development of a cylinder wake that will not agree with experiment.

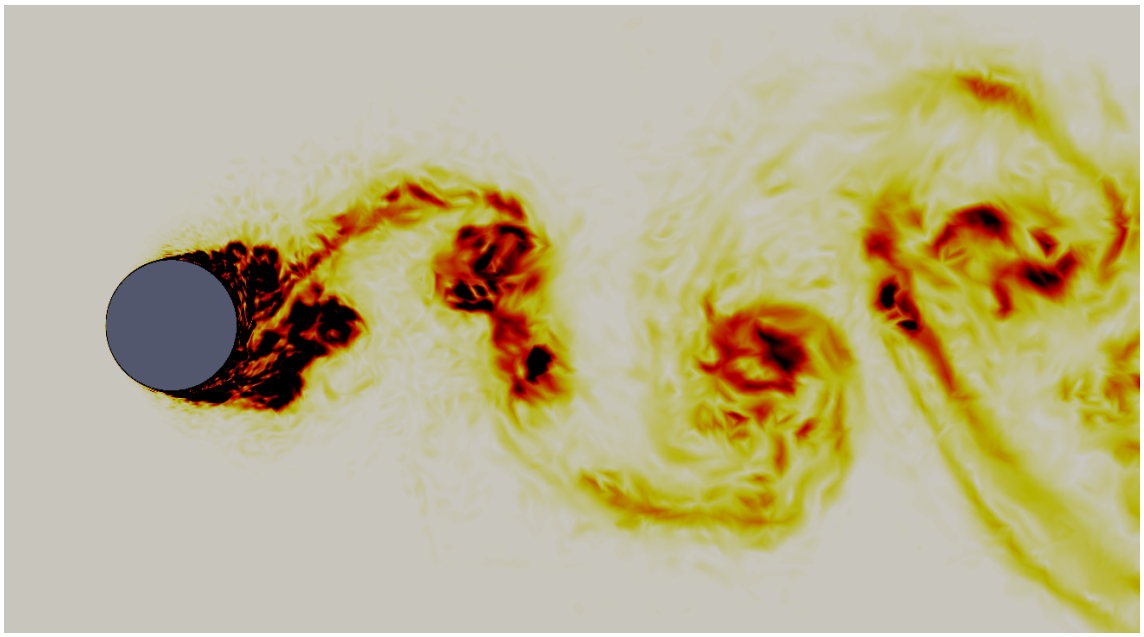


Figure 7.2.4: Vortices generated in the wake of the cylinder. Instantaneous vorticity magnitude, colour scale 1.0 to 100.0 non-dimensional units of vorticity. Simulation time $t_{nd} = 100.0$.

Figure 7.2.5 provides a view of the impingement of the vortices shown in figure 7.2.4 onto the aerofoil leading edge.

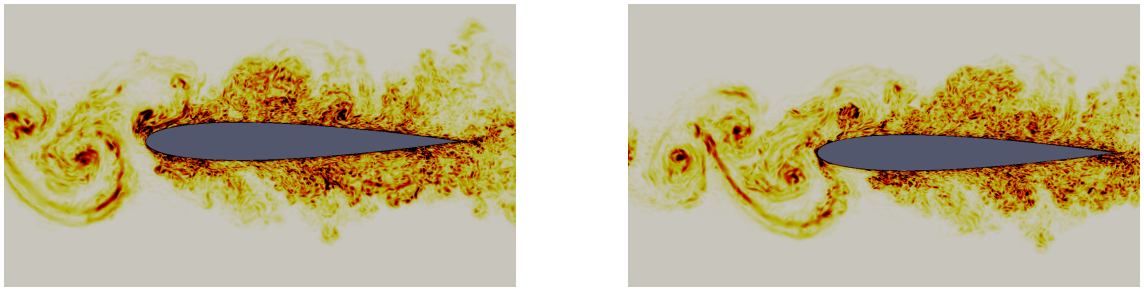


Figure 7.2.5: A vortex impinging on the aerofoil leading edge. Instantaneous vorticity magnitude, colour scale 1.0 to 100.0 non-dimensional units of vorticity. Left snapshot at simulation time $t_{nd} = 100.0$. Right snapshot at simulation time $t_{nd} = 100.165$.

The break-up of the vortices is apparent in the image above. The out-most front of the the vortex depicted in the left part of figure 7.2.5 has impinged on the leading edge of the aerofoil as illustrated in the right part of figure 7.2.5. The vortex will then breakdown and the resultant dynamics give rise to challenges related to the stability and robustness of the simulation. These challenges will now be discussed below.

As mentioned in the previous section, in an effort to optimise the simulation with respect to time step size that would lead to the lowest possible wall-clock times, the adaptive time step controller of PyFR was invoked from $t_{nd} = 8.0$ (phase 3B). Figure 7.2.6 shows a series of resultant time steps using the aforementioned controller.

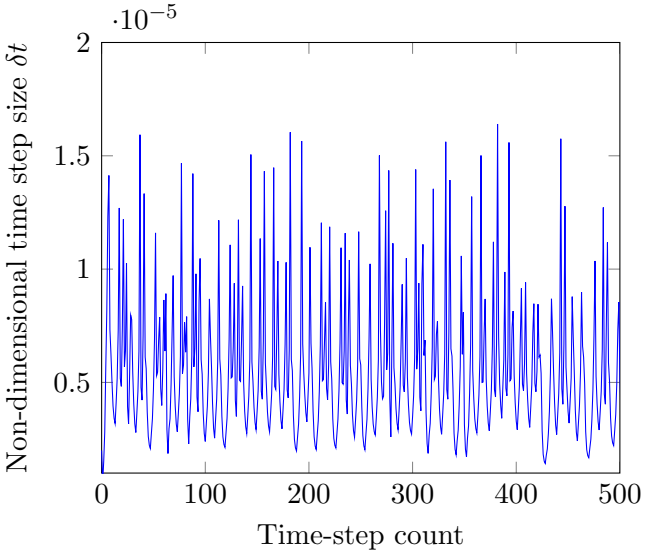


Figure 7.2.6: Time history of time step size δt using the adaptive controller of PyFR. The 500 iterations of the plot cover a non-dimensional simulation time of $t_{nd} \approx 0.0019$.

The behaviour shown in figure 7.2.6 retained the same nature for $O(20)$ flow passes over the cord. It demonstrates the difficulty of adequately modelling the dynamics depicted in figure 7.2.5: The adaptive time step size control procedure constantly adjusts the time step size in response to rapidly developing flow structures. It can be seen that the adjustment can even reach an $\sim 8\times$ change in time step size in the space of a few iterations.

The severity of this type of flow dynamics did however result in situations which resulted in instabilities that neither the used anti-aliasing scheme nor the adaptive time stepping controller were able to alleviate.

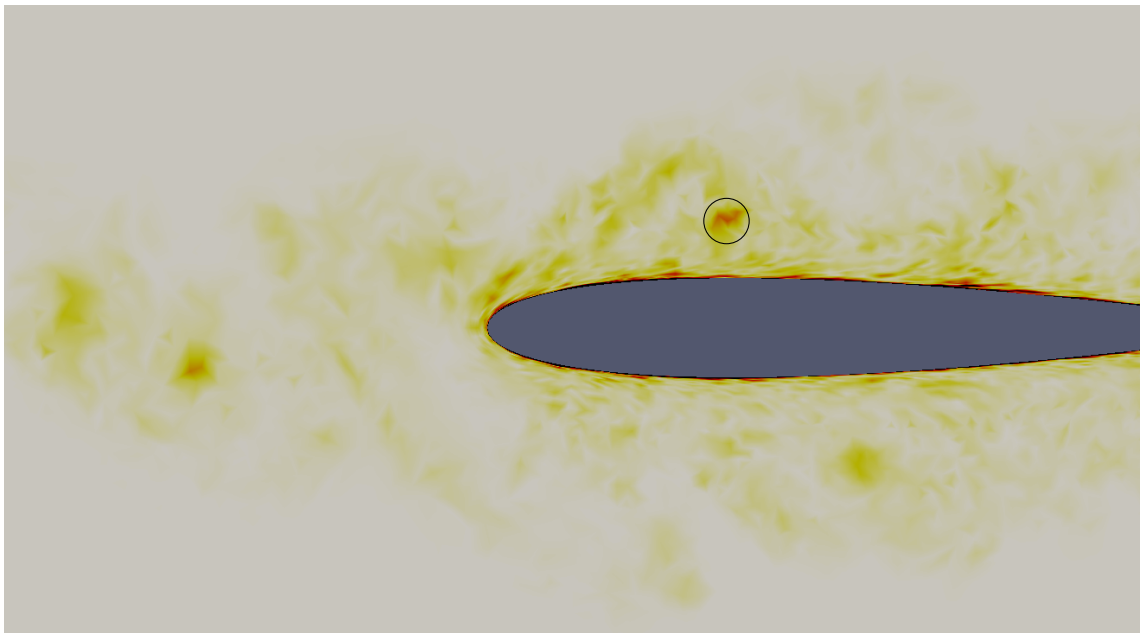


Figure 7.2.7: Instantaneous vorticity magnitude prior to local instability at $t_{nd} \approx 41.3$. Colour scale 1.0 to 300.0 non-dimensional units of vorticity. The circle denotes the problematic area vorticity magnitude above the 300.0 scale.

Figure 7.2.7 shows the instantaneous flow condition prior to a local instability that the time step size controller was not able to prevent. Attention is drawn to the change in the colouring scale compared to the standard $[1.0, 100.0]$ range used, in order for the intensity of the flow dynamics inside the problematic vortical region to be more clearly illustrated.

In order for the simulation to be stabilised, the entropy stability scheme introduced in Chapter 6 was utilised. As mentioned in Chapter 6, this strategy consists of two components:

- the entropy limiter.
- the CFL controller.

The first component limits (or filters) the solution in a manner that would, at least in theory, guarantee a bound in the entropy of the simulation across the entire domain. The second component would attempt to control the maximum CFL number of the simulation in a manner that would, at least in theory, provide additional stability to the explicit time marching scheme. The chosen reference entropy s^0 for the limiter was set as identified by equation 6.2.10. The details of the

scheme are given in section 6.2.

$$s^0 = s_\infty = \ln(p_\infty) - \gamma \ln(\rho_\infty). \tag{7.2.1}$$

with s_∞, p_∞ and ρ_∞ being the reference entropy, pressure and density respectively.

Following the re-starting of the simulation from a point in time located two flow passes prior to the local instability point of $t_{nd} \approx 41.3$ with both the entropy limiter and the CFL controller utilised, it was possible to adequately stabilise the simulation and move through the time level at which the instability would otherwise set in.

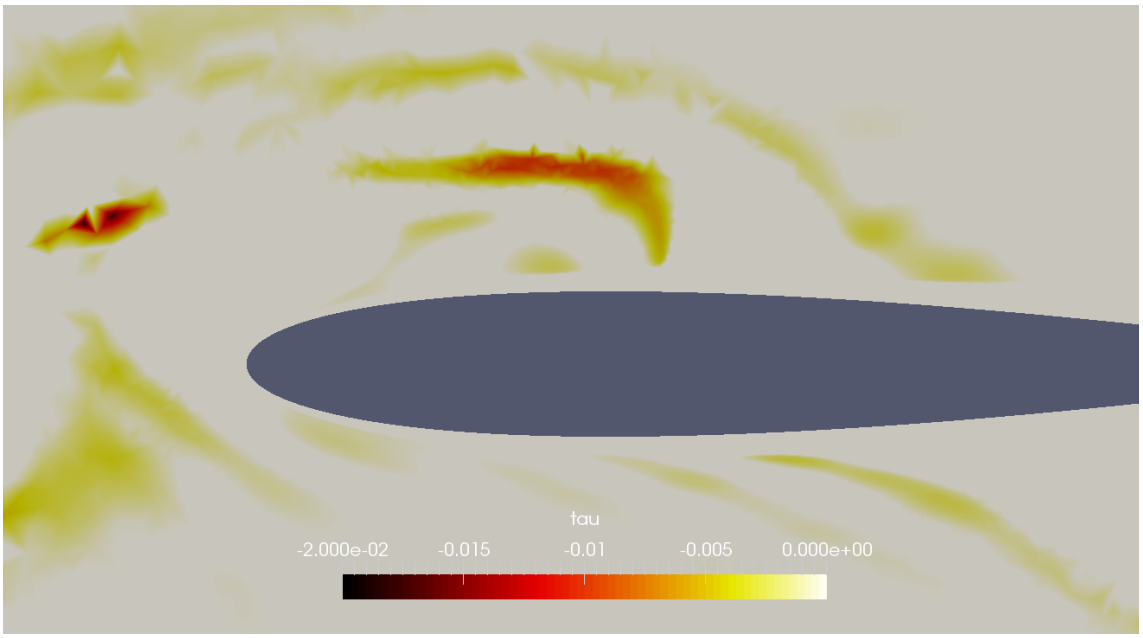


Figure 7.2.8: Entropy stability limiter parameter τ (defined in section 6.2) distribution and intensity in the vicinity of the circled area of figure 7.2.7.

Figure 7.2.8 shows the intensity distribution of parameter τ (defined in section 6.2) which can be used as a measure of the degree at which the entropy limiter acts upon each region, as outlined in section 6.2. The limiter was able to suppress the errors leading to instability in the area circled in figure 7.2.7, as well as potential problems in other areas. The above demonstrate the benefits of the entropy stability implementation into PyFR.

The use of the CFL controller proved less successful however. After approximately nine flow passes following the deployment of both components of the entropy stability scheme, the simulation failed in a similar manner to the one described earlier, due to local instability. It was found however that a combination of the entropy limiter

and a constant time step size of $\delta t = 2 \times 10^{-6}$ was sufficient to ensure a problem-free simulation up to $t_{nd} \approx 100.0$.

7.2.2 Flow Statistics

In line with the procedure outlined in section 5.2.1, the point at which the turbulent statistics were deemed to have settled and the transient phase completed was identified via an identical convergence study. Time $t_n d = 30.0$ was identified as the aforementioned point and hence, all post-processing was applied on data inside the $t_{nd} = [30.0, 100.0]$ range.

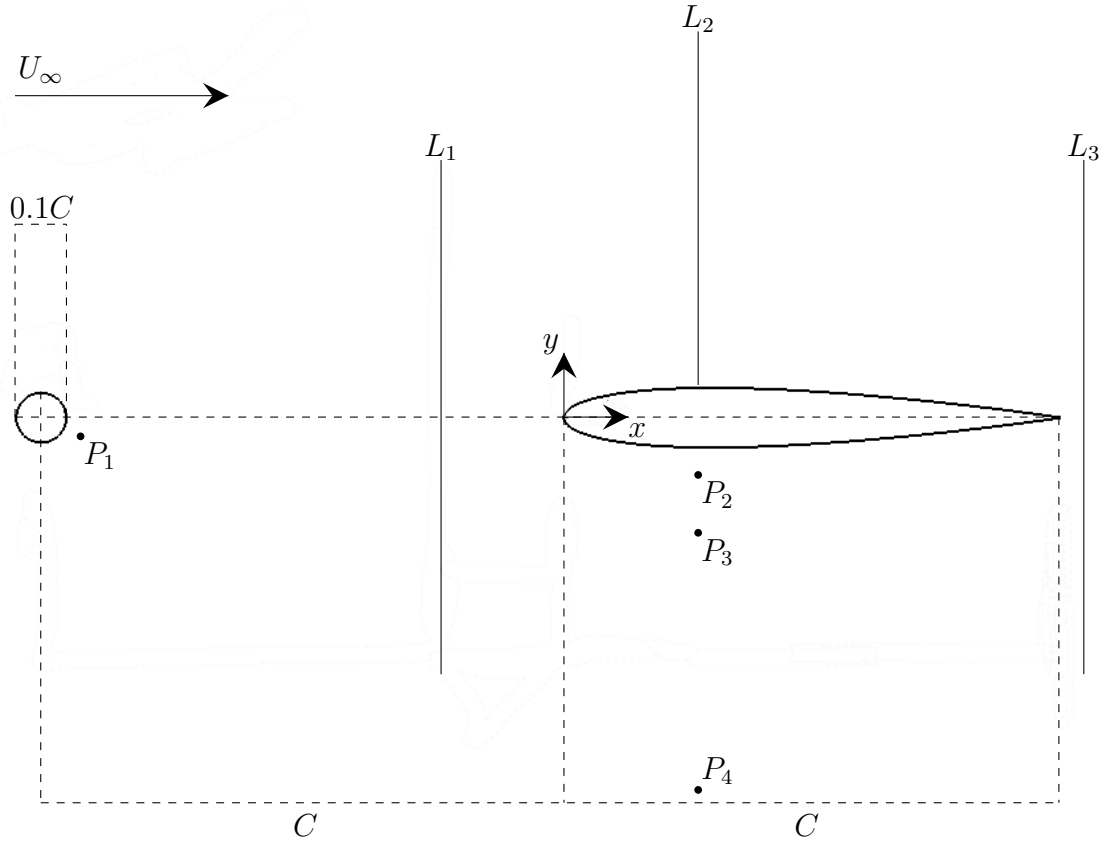


Figure 7.2.9: Definition of data extraction regions. Points P_1, P_2, P_3 and P_4 have coordinates $[-0.87C, -0.05C]$, $[0.25C, -0.16C]$, $[0.25C, -0.30C]$ and $[0.25C, -0.90C]$ respectively. Their spanwise location is the middle of the domain. Lines L_1, L_2 and L_3 are parallel to the y axis and their x coordinates are $-0.255C$, $0.25C$ and $1.1C$ respectively. Their spanwise location is the middle of the domain.

The results presented below were produced via averaging in time all solution fields inside the averaging period covering approximately 70 flow passes. Each

solution field comprised of the five conserved variables $[\rho, \rho u, \rho v, \rho w, \rho E]$ across the entire flow domain, a cross-section of which is depicted in figure 7.1.1. The period between the generation of each solution volume was 0.005 non-dimensional time units. Once the time averaged field was calculated, a series of ten cross-sections normal to the spanwise direction were averaged as well, so as to obtain spatial averaging in the spanwise direction as well. The resultant averaged 'slice' was used in order to obtained velocity profiles along lines L_1, L_2 and L_3 of figure 7.2.9.

Wake of the Cylinder

In this section the focus will be the wake of the cylinder. The time-averaged flow field and the turbulent intensities will be investigated along line L_1 of figure 7.2.9.

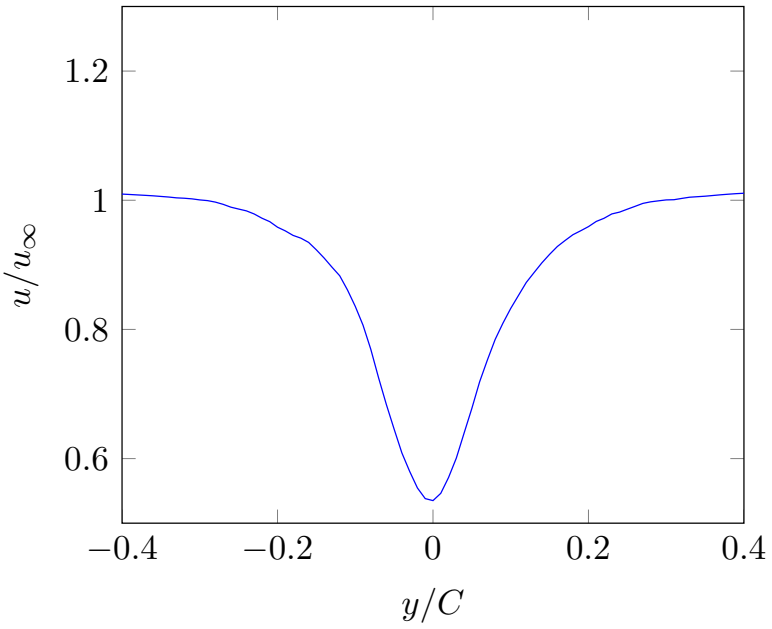


Figure 7.2.10: Time-averaged streamwise velocity u profile predicted via simulation with PyFR, along line L_1 of figure 7.2.9. Velocity u_∞ is the freastream velocity. Averaging period $t_{nd} \approx 30.0$ to $t_{nd} \approx 100.0$.

.

Figure 7.2.10 shows the time-averaged stream-wise velocity component u in the wake of the cylinder. As expected, a deficit in the velocity profile is observed which reflects the momentum loss mainly caused by the pressure difference across the cylinder.

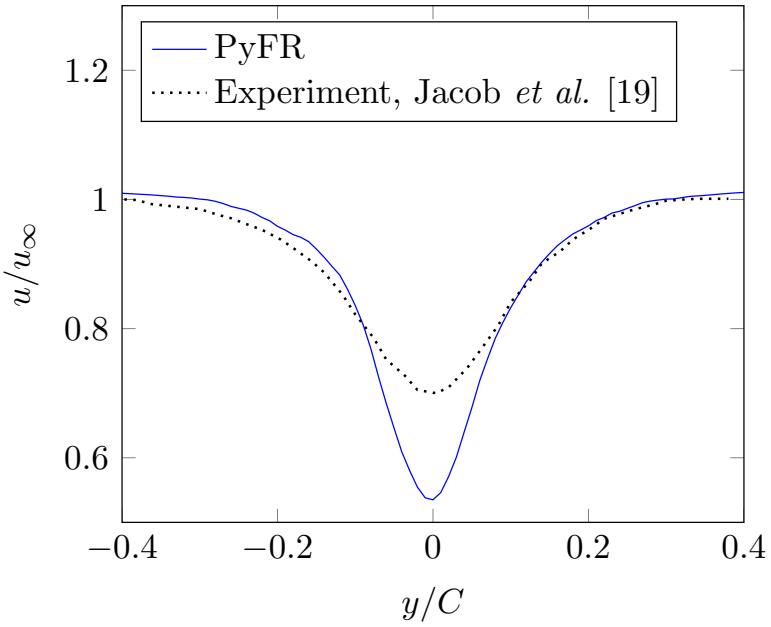


Figure 7.2.11: Time-averaged streamwise velocity u profile along line L_1 of figure 7.2.9. Comparison with experiment.

.

Figure 7.2.11 gives a comparison between the solution of the present study and the experiment of Jacob *et al.* [19]. Whilst the two curves agree well outside the region approximately defined by $\mid y/C \mid < 0.1$, inside that region there is substantial difference, particularly in the peaks of the velocity deficit. More specifically, PyFR predicts a larger velocity deficit. This gives rise to the expectation that the pressure difference across the cylinder is greater than physically measured, with a greater than expected proportion of stream-wise momentum being lost by the pressure difference, and thus a higher drag being resultant.

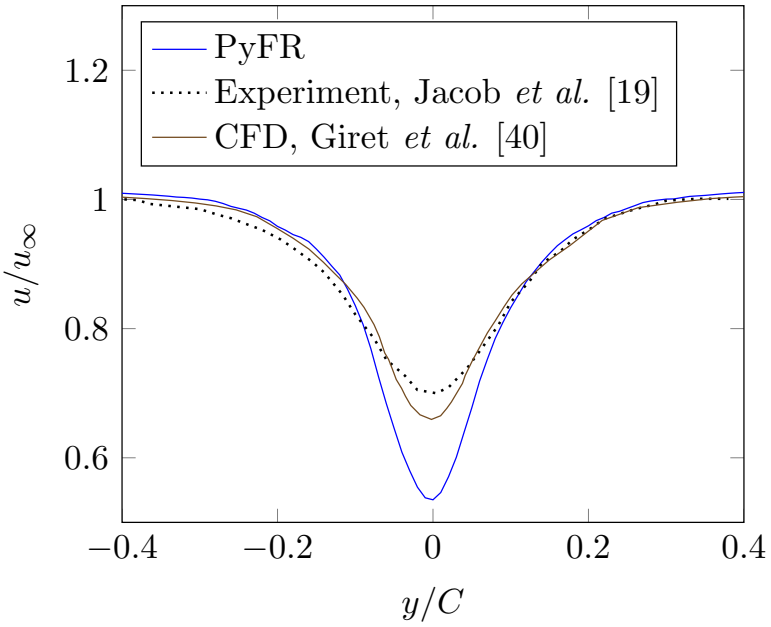


Figure 7.2.12: Time-averaged streamwise velocity u profile along line L_1 of figure 7.2.9. Comparison with experiment and CFD.

.

A comparison with the compressible LES approach of Giret *et al.* [40], reveals further agreement outside of the approximate region $\mid y/C \mid < 0.1$, whilst indicating the possibility that the explicit sub-grid scale modelling of [40] potentially offers higher accuracy in the prediction of the flow around the cylinder. As argued in chapter 5 of the present work, an important parameter that controls this behaviour is arguably the accuracy at which the coherent vortices along with their smaller-scale structures are modelled. Furthermore, the larger velocity deficit predicted results in higher momentum deficit in the x direction, considering that the relatively low Mach number of $M = 0.21$ does not result in significant changes in density. A greater momentum deficit would suggest a greater predicted drag coefficient for the cylinder relative to the one established in previous studies. The expected cylinder drag coefficient is $C_d = 1.05$ based on [55], [56] and [57]. With the contribution of frictional drag ignored, one can write the following expression.

$$C_d = \frac{\int_0^{2\pi} p(\phi) sin(\phi) r_c d\phi}{\rho_\infty u_\infty^2 r_c}. \tag{7.2.2}$$

In order to further investigate the greater momentum deficit observed in the results of the PyFR simulation, the pressure field was integrated around the cylinder

and the resultant cylinder drag coefficient C_d was evaluated.

In equation 7.2.2, ϕ is the angular coordinate with $\phi = 0$ corresponding to the lowest point in the cylinder circumference as per figure 7.2.9. Angle ϕ increases clockwise. Furthermore, $p(\phi)$ is the corresponding time-averaged pressure on the cylinder and r_c is the radius of the cylinder.

The result of equation 7.2.2 performed on the time-averaged pressure field yielded a drag coefficient of $C_d^p = 1.19$. Here, superscript p denotes the drag coefficient as computed via the integration of pressure. The value of the resultant drag is consistent with the higher velocity and momentum deficits observed compared with experiment: The pressure loss prediction across the cylinder appears to be larger leading to greater predicted drag.

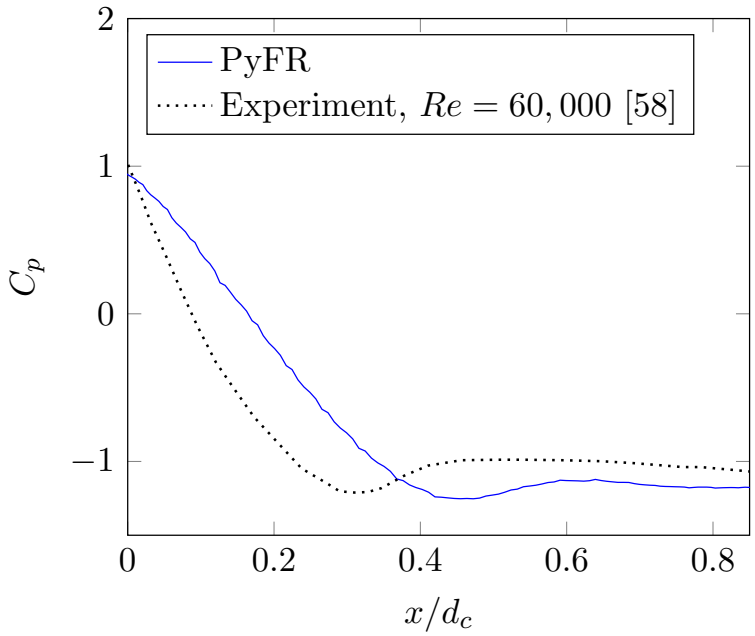


Figure 7.2.13: Pressure coefficient C_p of the time-averaged field around the cylinder. Quantity d_c denotes the diameter of the cylinder. Coordinate x is the streamwise coordinate as per figure 7.2.9. $x = 0$ corresponds to the leading edge of the cylinder

.

Figure 7.2.13 shows a comparison between the pressure coefficient C_p as obtained by experiments from Norberg [58] and the pressure coefficient as obtained by the time-averaged results of PyFR.

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty u_\infty^2}. \tag{7.2.3}$$

It is relevant to note that the experimental result shown in figure 7.2.13 corresponded to a cylinder-diameter defined Reynolds number of $Re = 60,000$ as opposed to $Re_c = 48,000$ of the PyFR simulation. This result provides further evidence that the pressure loss is over-predicted by PyFR, with the PyFR curve reaching lower values than experiment in the downstream region. However, one obvious aspect of figure 7.2.13 is the inability of the present work to agree with experiment in the upstream and effectively inviscid region of the C_p distribution. As the latter observation cannot be explained purely by a possible lack in time-averaging convergence or a lack in resolution and/or sub-grid scale modelling, there is need for investigation of the phenomenon.

Further investigation into the nature of the larger momentum deficit predicted by PyFR in the wake of the cylinder can be made via the comparison of $C_d^p = 1.19$ with the drag coefficient C_d^m computed via a momentum and force balance operation applied on a control volume that encloses the cylinder. The control volume was a rectangle and was symmetrical with respect to axis x of figure 7.2.9. Its upper and lower faces had a distance of $H = 2.4C$ or 24 cylinder diameters between them in order for the flow in their vicinity to have negligible variation from the free-stream. It was calculated that the stremwise component of the momentum transport across the upper and lower sides of the control volume was negligible as an order of magnitude. The trailing face of the control volume was adjacent to line L_1 of figure 7.2.9. The leading face was far enough upstream so that it was entirely located into the uniform free-stream. It was also assumed that the (time-averaged) flow field inside the control volume was steady. Viscous forces were also neglected. Consequently, the stream-wise force-momentum balance of the control volume can be expressed in terms of quantities along only the leading and trailing faces of the control volume. The coordinate along the trailing face is denoted by h.

$$F_c = \rho_\infty u_\infty^2 H + p_\infty H - \int_0^H (\rho u^2 + p) dh. \tag{7.2.4}$$

$$C_d^m = \frac{-F_c}{\rho_\infty u_\infty^2 r_c}. \tag{7.2.5}$$

In equations 7.2.4 and 7.2.5, F_c is the x-component of the force that the cylinder exerts on the fluid, i.e. the opposite of the cylinder's drag.

Equations 7.2.4 and 7.2.5 resulted in $C_d^m = 1.31$. The result is once more consistent with the greater velocity and momentum deficits predicted. However, it is significantly larger than $C_d^p = 1.19$ as computed by the pressure distribution around the cylinder. It is possible that the (omitted) viscous contribution in the cylinder drag can account for part of the observed difference. Furthermore, the accuracy at which the integrations in equations 7.2.2 and 7.2.4 can also potentially account for part of the difference. It is the opinion of the author that a non-negligible proportion of the difference can potentially be attributed to non-sufficient time-averaging, with residual unsteady behaviour present in the averaged field inside the control volume. As will be demonstrated below in this chapter, the indication of insufficient time-averaging arises in other results as well. Further investigation into this behaviour is however needed in order for reliable conclusions to be drawn.

Further insight into the velocity deficit in the cylinder wake can be made via the most recent work of Agrawal *et al.* [41], [42].

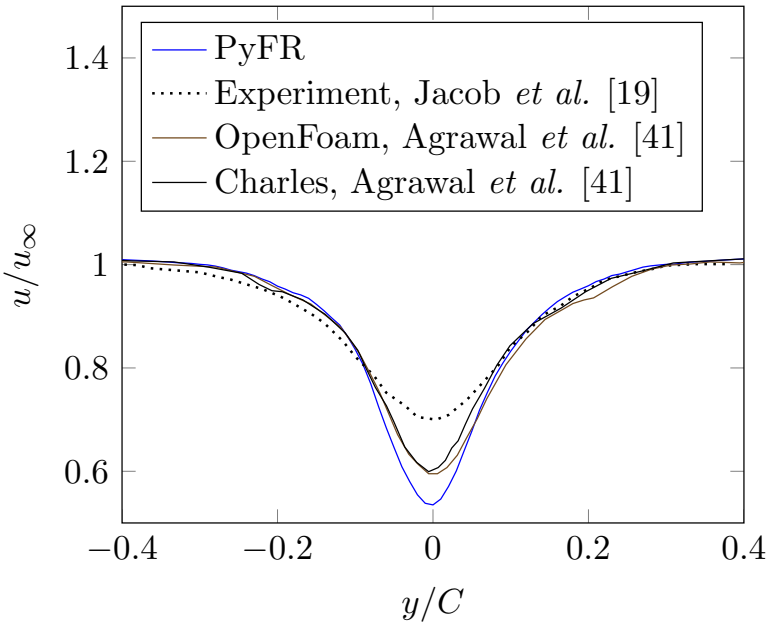


Figure 7.2.14: Time-averaged streamwise velocity u profile along line L_1 of figure 7.2.9. Comparison with experiment and the LES work of Agrawal *et al.* [41].

.

In figure 7.2.14 one can observe that both the compressible LES solver Charles and the incompressible LES solver PisoFoam (OpenFoam) of [41] predict a significantly larger deficit compared to the result obtained by Giret *et al.* [40] shown in figure

7.2.12. The significant difference in the resolution of the two studies as summarised in table 4.2.1 indicates that the exact type and sub-grid scale modelling of an LES approach has significant influence in the quality of the prediction. Whilst the result of PyFR depicts an even greater deficit, one can appreciate the failure of the LES approaches of [41] to match the LES approach of [40] in terms of the time-averaged cylinder wake, despite a significant increase in resolution.

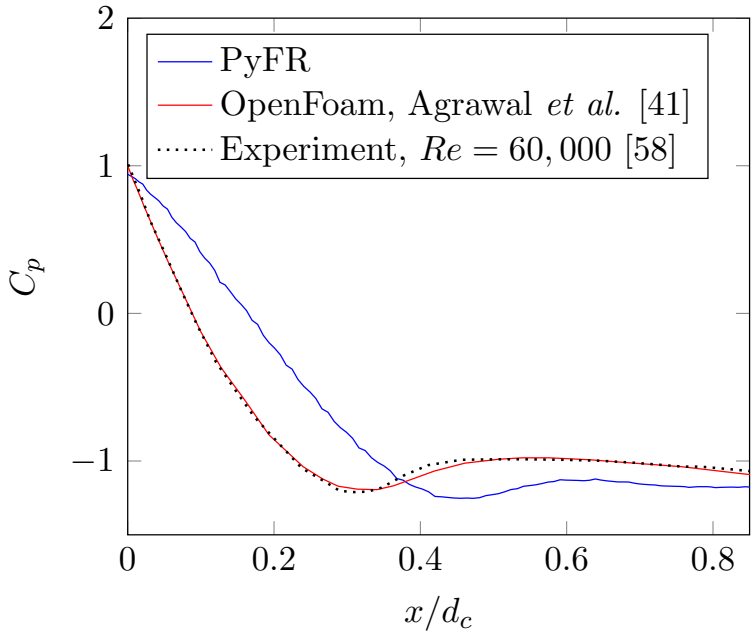


Figure 7.2.15: Pressure coefficient C_p of the time-averaged field around the cylinder. Quantity d_c denotes the diameter of the cylinder. Coordinate x is the streamwise coordinate as per figure 7.2.9. $x = 0$ corresponds to the leading edge of the cylinder

.

However, figure 7.2.15 depicts near-perfect agreement between the LES approach of [41] and experiment, indicating the success of the approach to capture the pressure coefficient distribution. The last two, apparently conflicting, observations indicate that there is a fundamental aspect of the flow that is not captured by PyFR.

The turbulence intensity of the cylinder is shown in the following figure 7.2.16.

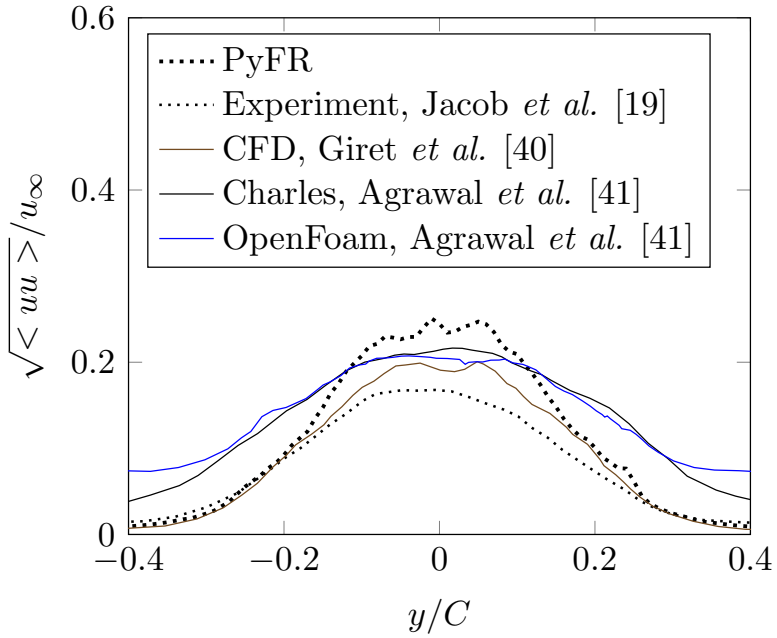


Figure 7.2.16: Profile of turbulence intensity $\sqrt{< uu >}$ along line L_1 of figure 7.2.9. Comparison with experiment and other CFD studies.

.

Inspection figure 7.2.16 re-enforces the observation that the higher velocity deficit of the time-averaged field predicted by PyFR results in higher du/dy gradients, which in turn results in the production of turbulence of higher intensity. Furthermore, in the vicinity of the region approximately defined by $\mid y/C \mid < 0.1$ and mentioned in the discussion of figure 7.2.11, the turbulence intensities predicted by the LES simulations of [41] are higher than those predicted by both experiment and the study of [40], which is consistent with the previous observation regarding the velocity deficit. A further important observation is the apparent lack of symmetry in the result obtained by the present work. As argued earlier, this lack of symmetry points towards insufficient time-averaging.

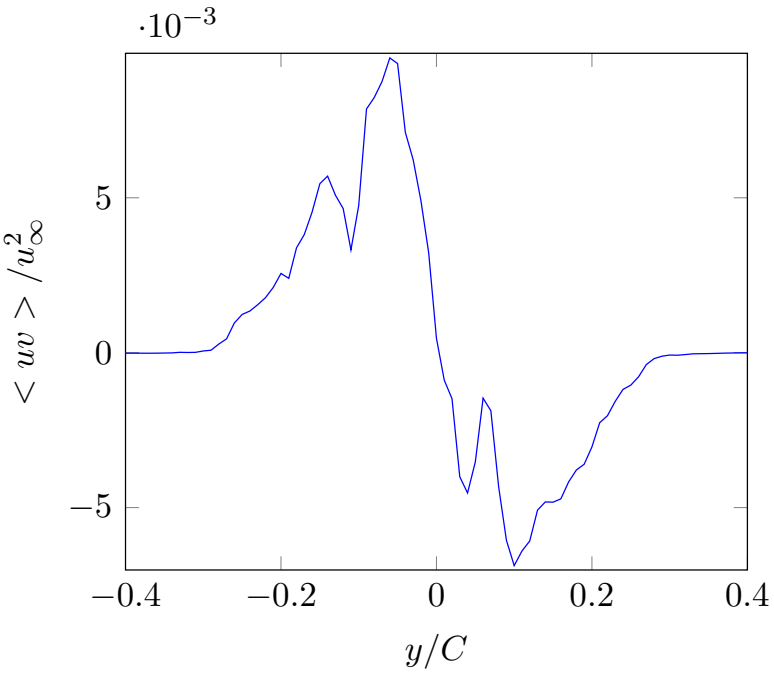


Figure 7.2.17: Time-averaged Reynolds stress $< uv >$ profile along line L_1 of figure 7.2.9 via simulation with PyFR.

The behaviour of the shear Reynolds stress $< uv >$ along line L_1 is demonstrated in figure 7.2.17. However, the existence of two additional antisymmetric extremes on the curve, apart from the two absolute extremes, indicates a possible non-convergence of the statistics of the flow further indication a lack of sufficient time-averaging. As demonstrated in the work of Berajeklian *et al.* [59], such non-smooth behaviour of the curve should not occur.

The above observations lead to the conclusion that the moderate success that the present work has had in resolving the wake of the cylinder will have an impact into the quality of the prediction of flow field quantities further downstream.

Side of the Aerofoil

This section discusses the behaviour of the flow in the vicinity of the aerofoil side.

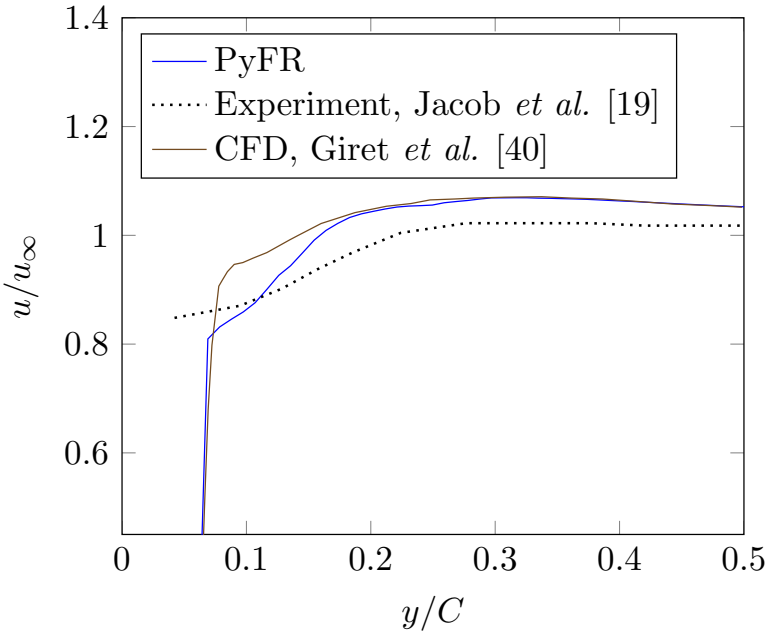


Figure 7.2.18: Time-averaged streamwise velocity u profile along line L_2 of figure 7.2.9. Comparison with experiment and CFD.

.

Figure 7.2.18 reveals closer agreement between PyFR, experiment [19] and the LES simulation of [40], compared to the agreement observed in the wake of the cylinder. Of particular note is the close agreement between PyFR and [40] for $y/C > 0.2$. Differences observed for $y/C < 0.2$ indicate that the type of simulation (high-order FR DNS versus high-order LES) and the boundary layer resolution have considerable influence in the vicinity of the aerofoil surface. Furthermore, the significant velocity deficit over-prediction observed in wake of the cylinder potentially partially explains the increase in the velocity levels compared to experiment observed here: Mass conservation would require the greater velocity deficit observed in the wake of the cylinder to be balanced by accelerated flow downstream along the sides of the aerofoil.

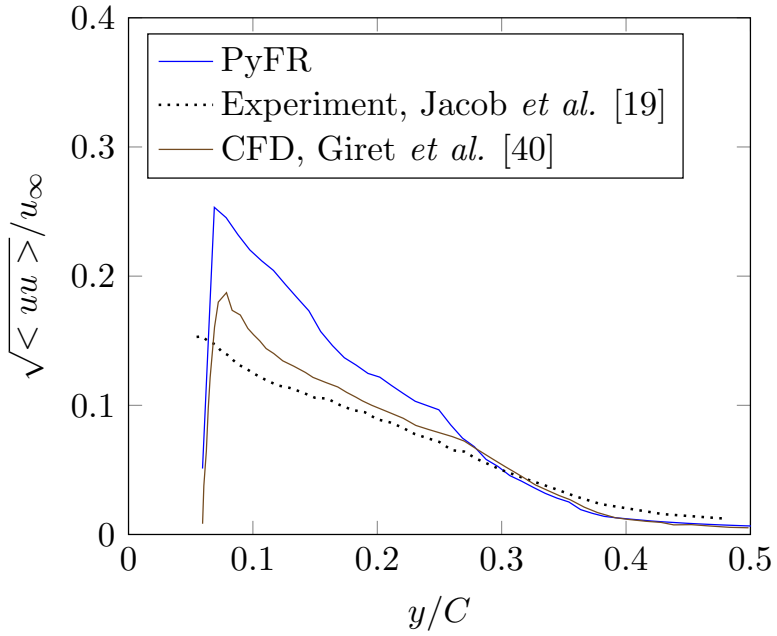


Figure 7.2.19: Turbulence intensity profile $\sqrt{< uu >}$ along line L_2 of figure 7.2.9. Comparison with experiment and CFD.

.

Reinforcing the observation made on figure 7.2.16, the turbulent intensity predicted by PyFR along the side of the aerofoil is higher than both experiment [19] and the LES of [40] with the latter two being in relatively close agreement. The larger $\frac{\partial u}{\partial y}$ velocity gradients of the blue PyFR curve of figure 7.2.18 in the vicinity of the wall is expected to lead to greater turbulence production, an assumption which is consistent with the observations of figure 7.2.19. The higher turbulent intensity further underlines the potentially lower dissipative nature of PyFR compared to an LES approach.

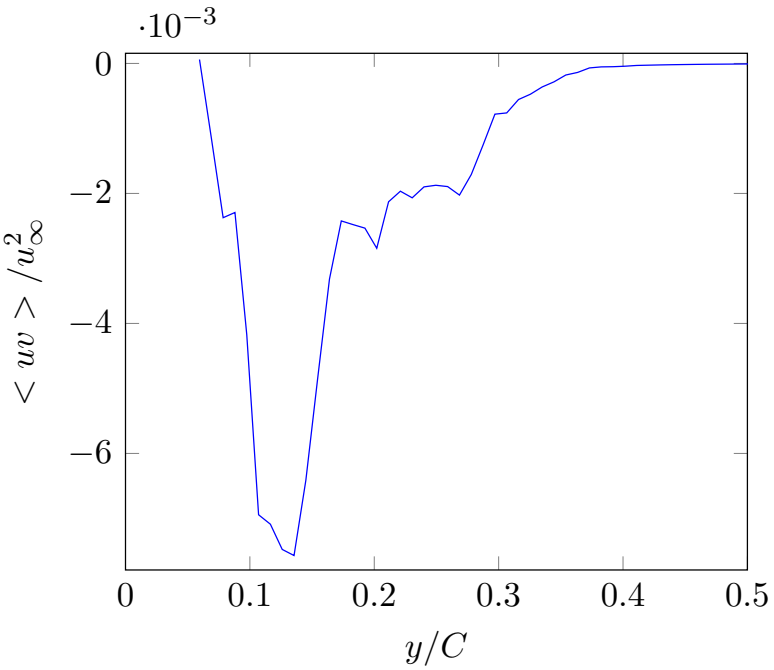


Figure 7.2.20: Time-averaged Reynolds stress $< uv >$ profile along line L_2 of figure 7.2.9 via simulation with PyFR.

In direct analogy to figure 7.2.17, curve non-smoothness approximately located within $0.2 < y/C < 0.3$ indicate a lack of absolute statistical convergence. The non-smooth behaviour is primarily centred on areas across which coherent vortices generated in the wake of the cylinder move. A future simulation and inclusion into the averaging operation of more convective passes could potentially confirm this hypothesis.

Wake of the Aerofoil

This section discusses the behaviour of the flow in the vicinity of the aerofoil wake.

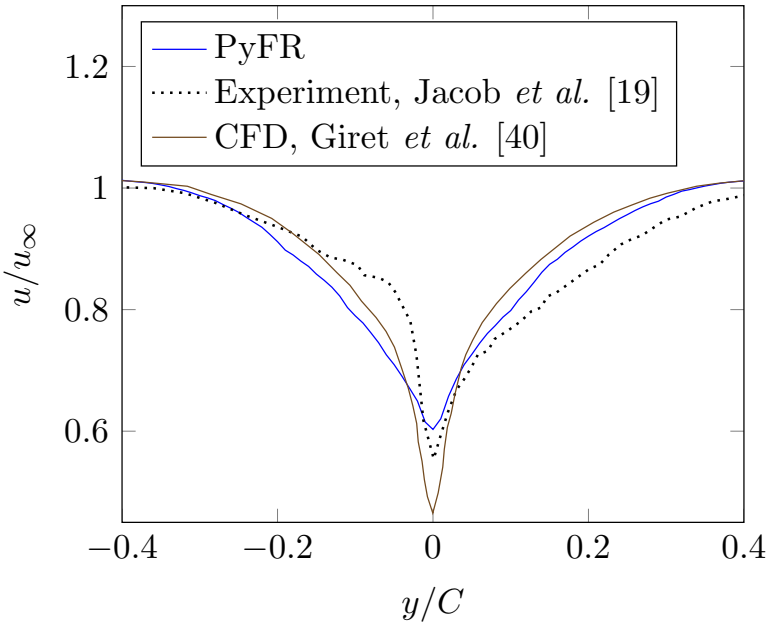


Figure 7.2.21: Time-averaged streamwise velocity u profile along line L_3 of figure 7.2.9. Comparison with experiment and CFD.

.

Arguably closer agreement between PyFR and experiment than is the case with [40] and experiment can be observed for the wake of the aerofoil. The relevant curves are shown in figure 7.2.21. In this instance, the extent of the velocity deficit is predicted more accurately compared to the case of the cylinder wake and the aerofoil side. However, one needs to take into account the fact that the wake of the cylinder is not adequately predicted as mentioned previously. Consequently, an error in the distribution of the mass flow has been introduced since the aerofoil is located downstream of the cylinder. Interestingly, in contrast to the observations made for the wake of the cylinder and the side of the aerofoil, the velocity gradient magnitudes $\frac{\partial u}{\partial y}$ as predicted by PyFR are lower than both experiment and the LES study of [40]. Consequently, one would expect lower turbulent intensities.

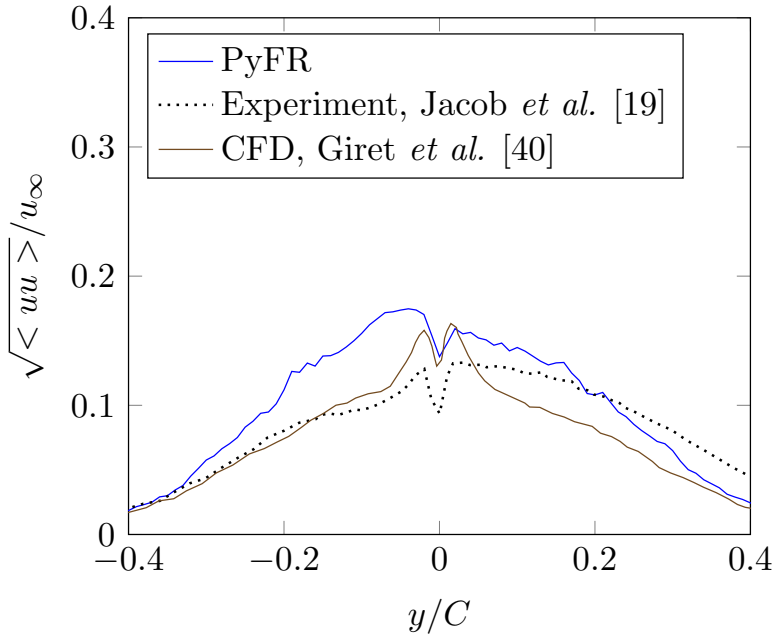


Figure 7.2.22: Turbulence intensity profile $\sqrt{< uu >}$ along line L_3 of figure 7.2.9. Comparison with experiment and CFD.

.

Figure 7.2.22 is not consistent with the last expectation however. Higher turbulent intensity is predicted by PyFR in the wake of the aerofoil, albeit in to a lesser extent than in the two previous instances. Furthermore there is a lack of symmetry with respect to the y axis of the plot which would indicate a lack of absolute statistical convergence. Figure 7.2.23 of Reynolds shear stress $< uv >$ further reinforces the hypothesis of inadequate statistical convergence due to non-smooth behaviour, whilst retaining the expected $< uv >= 0$ at $y = 0$.

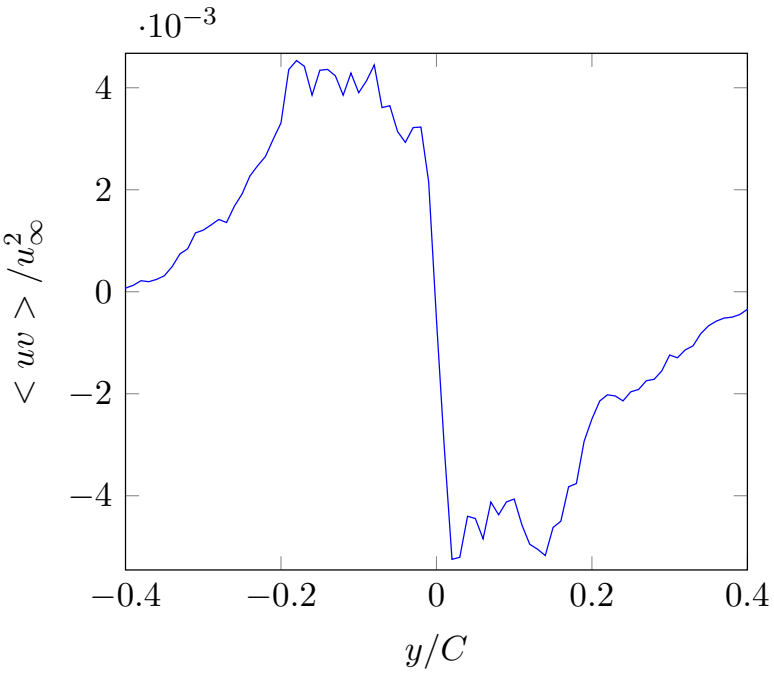


Figure 7.2.23: Time-averaged Reynolds stress $< uv >$ profile along line L_3 of figure 7.2.9 via simulation with PyFR.

.

As a result of the above, a general conclusion can be made about the agreement between experiment and PyFR: As to some extent observed via [40] as well, an increase in the intensity and complexity of the flow field results in an decrease in agreement between experiment and PyFR. As discussed previously, this behaviour is arguably partly due to inadequate dissipative characteristics of PyFR and also a lack of absolute statistical convergence. It is possible that the absence of explicit sub-grid scale modelling in PyFR limits the ability to successfully simulate the flow around the Rod-Aerofoil configuration across all scales.

Velocity Spectra

In this section, the spectra of instantaneous data from points of significance in the flow domain will be presented. All spectra were calculated in the following manner: Time series of instantaneous velocity u sampled every 0.0002 non dimensional time units across ~ 70 flow passes over an aerofoil chord were extracted from points P_1, P_2, P_3 and P_4 of figure 7.2.1. The time series was dimensionalised with respect to the physical values of [19], namely $U_\infty = 72.0m/s$ and $C = 0.1m$. The power spectral density (PSD) of the dimensionalised time series was then estimated via

the *periodogram* and *welch* routines of Python's Scipy library [50].

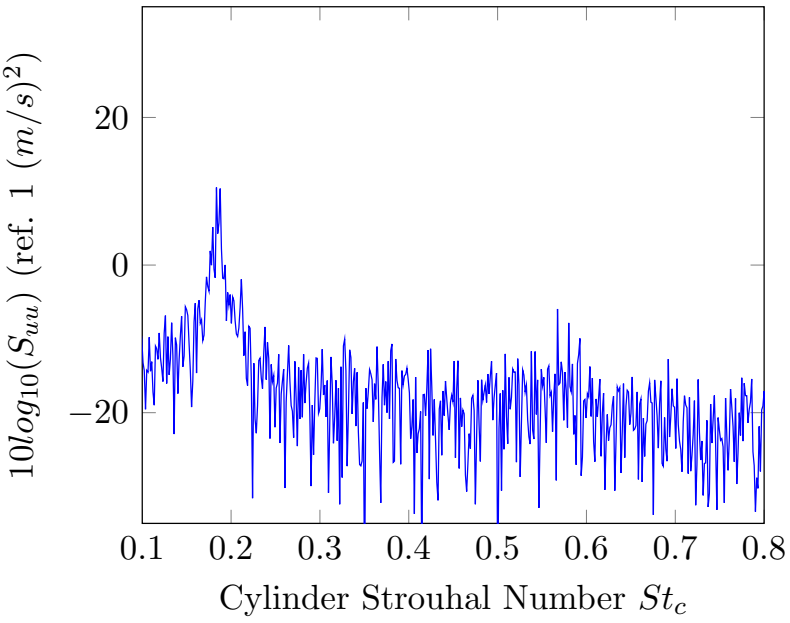


Figure 7.2.24: Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9 via simulation with PyFR.

.

Figure 7.2.24 shows the power spectrum of velocity u on point P_1. S_{uu} refers to the power of the time signal following the application of a Fast Fourier Transform (FFT) on it. An immediate observation is the maximum cylinder Strouhal number (St_c). The predicted number is $St_c \approx 0.19$, in good agreement with experiment [19]. The success of the simulation in estimating St_c in such accuracy can partly be attributed to the work presented in Chapter 5.

Filtering of the spectrum via the a Welch-type estimator results in the smoother curve depicted below.

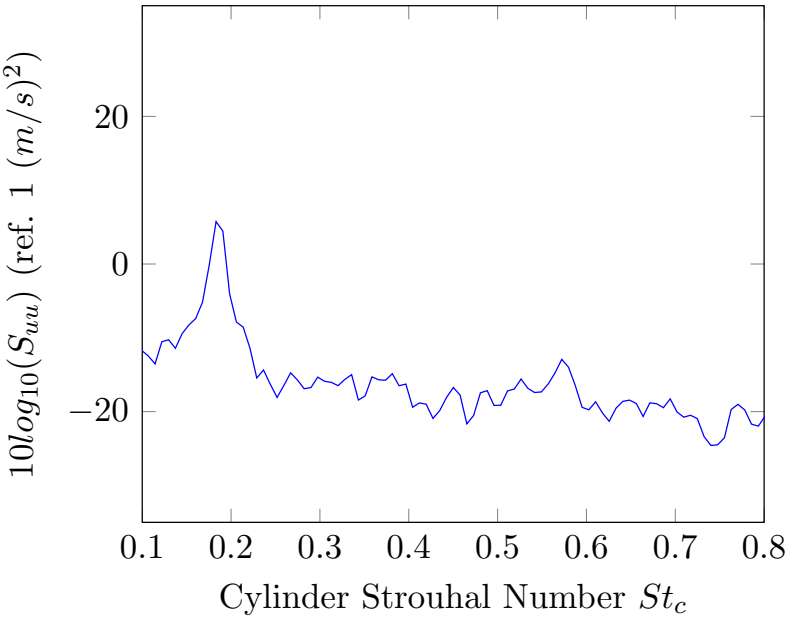


Figure 7.2.25: Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9 via simulation with PyFR. Estimation via Welch method of order 64.

.

Figure 7.2.25 accentuates the dominance of the vortex shedding and suggests that the dynamics of the flow in the vicinity of point P_1 are dominated by the large scale vortical structures, as is to be expected.

As a study into the validity of using a spectrum that is the average of spectra from points distributed along the spanwise direction, a 'rake' of nine equidistant points having the $x - y$ coordinates of point P_1 was used to obtain a spanwise average. The result of the averaging procedure is shown in figure 7.2.26.

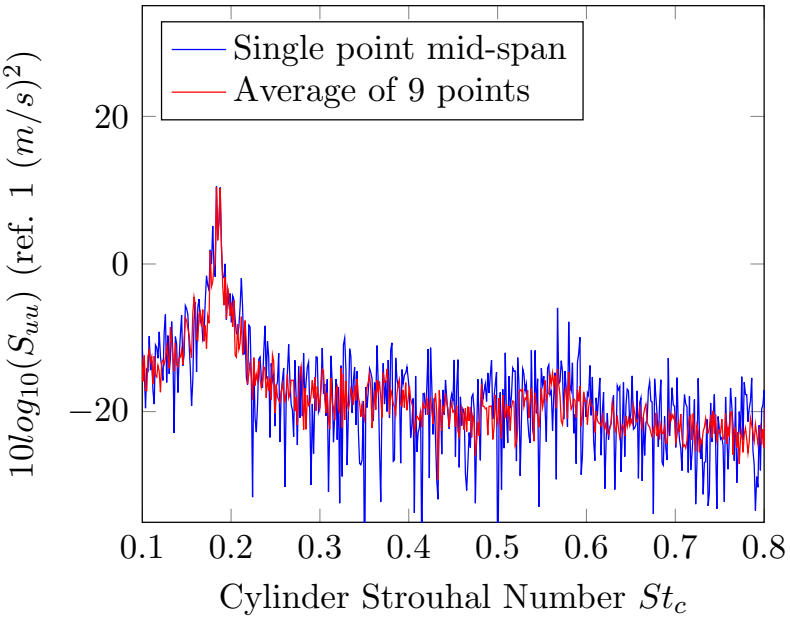


Figure 7.2.26: Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9 via simulation with PyFR. Comparison of one-point spectrum with average spectrum of nine equidistant points distributed along the spanwise direction.

.

Inspection of figure 7.2.26 reveals similar behaviour between the two approaches with respect to the predominant peaks of the spectrum. As a result, the averaging approach could, to some extent, replace the Welch type estimation if one seeks some form of filtering of the single-point, original spectrum.

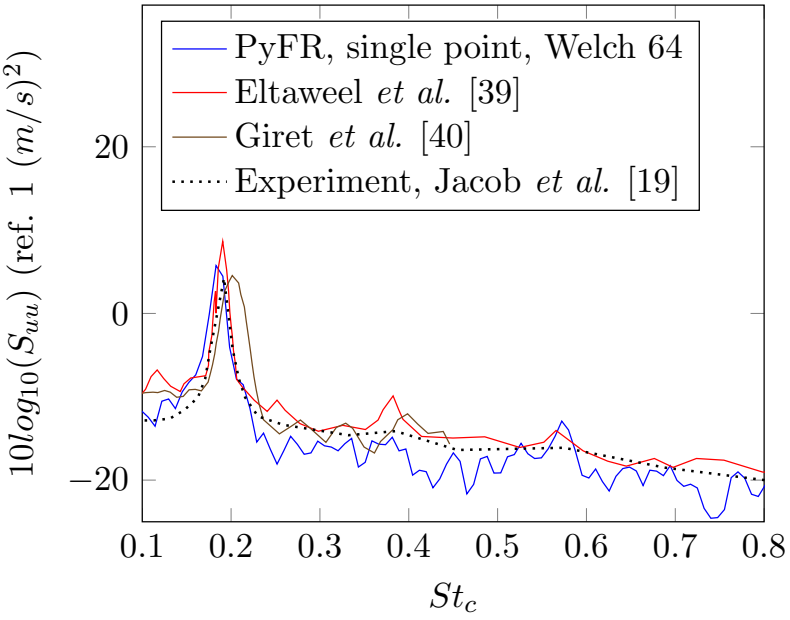


Figure 7.2.27: Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9. Comparison with experiment and other CFD approaches.

.

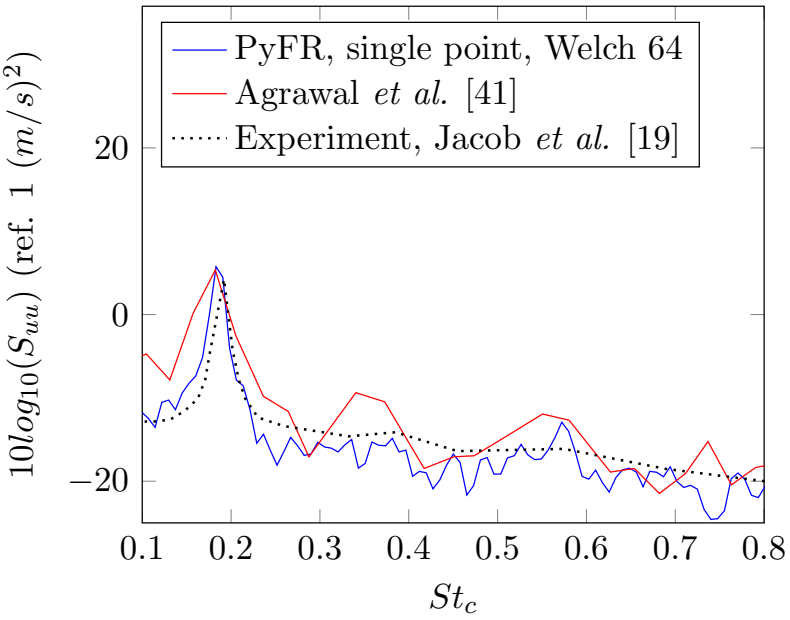


Figure 7.2.28: Power spectral density S_{uu} of velocity u at point P_1 of figure 7.2.9. Comparison with experiment and the compressible LES Charles code of Agrawal *et al.* [41].

.

As demonstrated earlier with the use of raw versus welch spectra, the specific algorithm used for the estimation of the PSD has significant influence on the 'resolution' of the curve, i.e. the number of individual frequencies represented. However, it is evident that the order of magnitude of the most powerful modes around $St_c = 0.2$ of the PyFR prediction remains in relatively good agreement with experiment. The mean of spectral fall-off is predicted well by both PyFR and the other CFD approaches included in figures 7.2.27 and 7.2.28, however there is a greater number of tonal peaks predicted by the CFD data, that are not present in experiment.

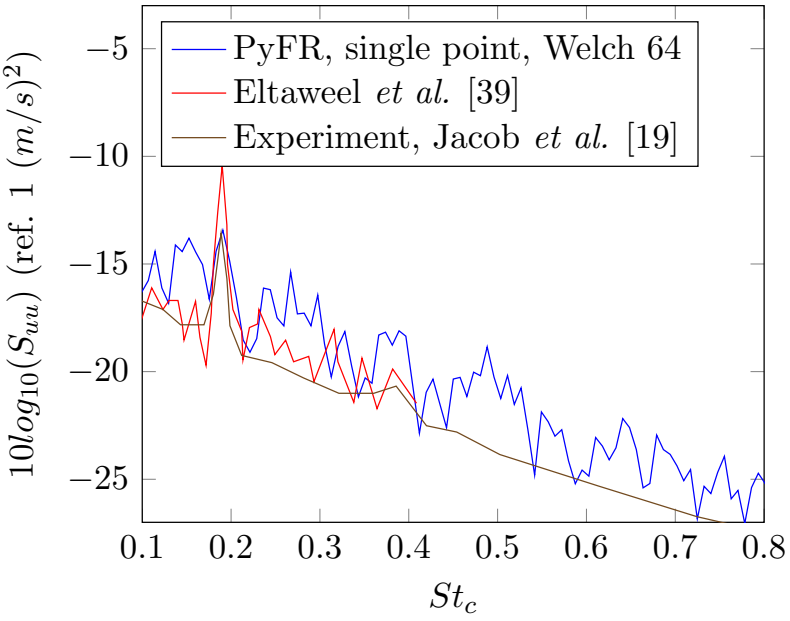


Figure 7.2.29: Power spectral density S_{uu} of velocity u at point P_2 of figure 7.2.9. Comparison with experiment and CFD.

.

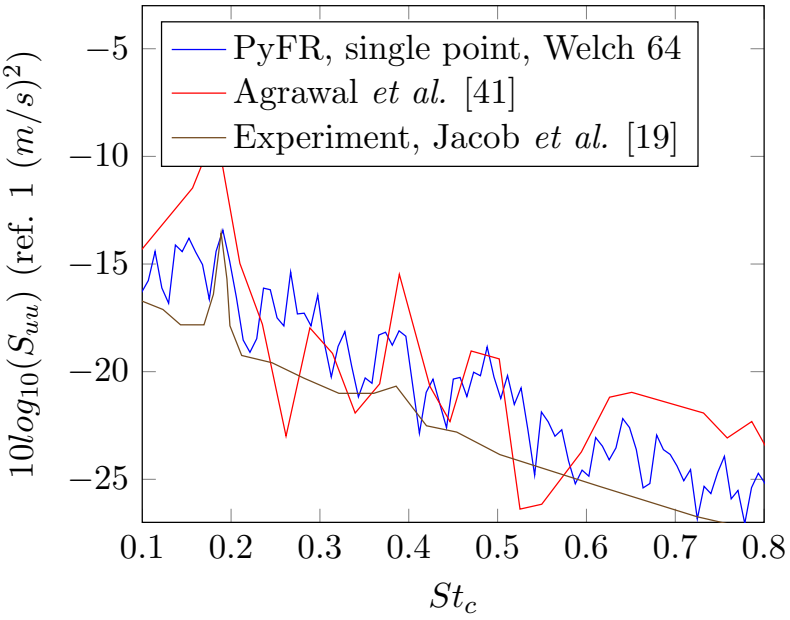


Figure 7.2.30: Power spectral density S_{uu} of velocity u at point P_2 of figure 7.2.9. Comparison with experiment and the compressible LES Charles code of Agrawal *et al.* [41].

.

Figures 7.2.29 and 7.2.30 provide a further comparison of the performance of PyFR against experiment and other CFD approaches. The spectra shown correspond to point P_2 of figure 7.2.9. Agreement with both experiment and the two LES studies is not close, except for the continued predominance of the shedding mode $St_c \approx 0.2$. The energy distributed over a wide range of frequencies is of usually higher magnitude, reinforcing the theory of the the less dissipative nature of PyFR. A notable exception is the case of the Charles compressible LES study of Agrawal *et al.* [41]. Through the significant differences observed between various approaches, one can conclude that the power spectrum of the velocity at point P_2 is highly dependant upon the type of solver used and the resolution of the computational grid.

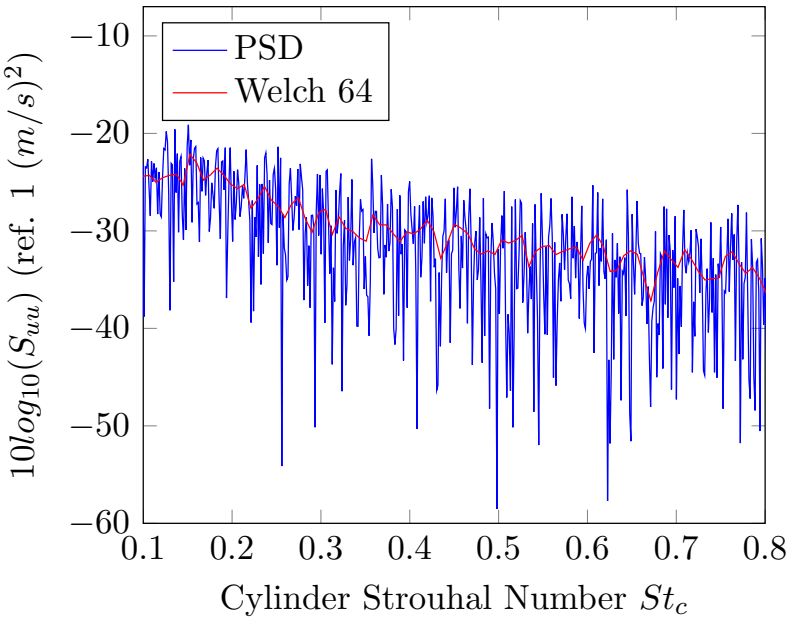


Figure 7.2.31: Power spectral density S_{uu} of velocity u at point P_3 of figure 7.2.9 via simulation with PyFR.

.

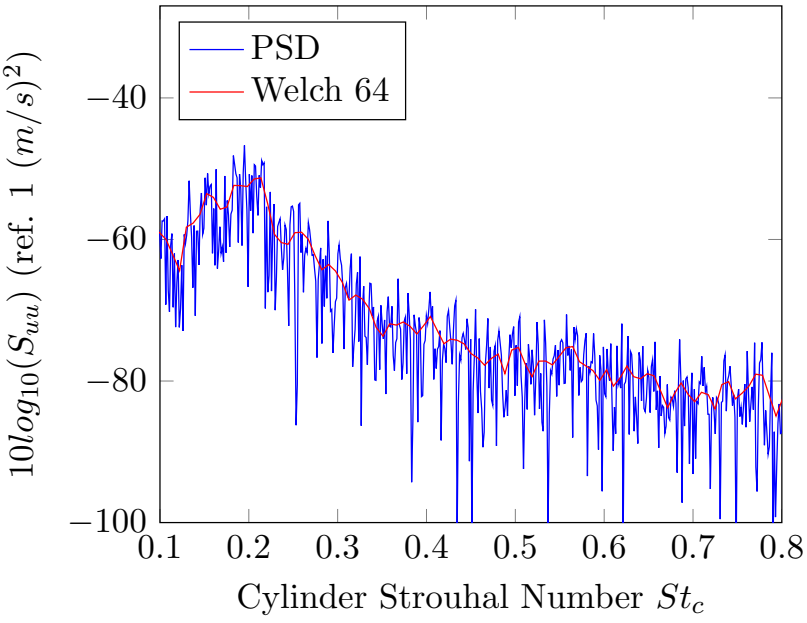


Figure 7.2.32: Power spectral density S_{uu} of velocity u at point P_4 of figure 7.2.9 via simulation with PyFR.

.

Inspection of figures 7.2.26 and 7.2.27 reveals that the power of the dynamics modes diminishes rapidly with increasing distance from the aerofoil. However, except in the case of point P_3 a general power peak can still be identified in the vicinity of $St_c = 0.2$. The last observation indicates that relatively coherent structures originating from of the cylinder-shed vortices are still present, even at a distance approximately equal to a chord length from the side of the aerofoil.

7.3 Conclusions

Based on the results presented in the current chapter, the following conclusions may be drawn.

- Simulating the Rod-Aerofoil problem required access to considerable computational resource, equal to an equivalent of 200 Nvidia P100 GPUs operating for an order of 500 hours.

- Despite the considerable computational resource of the above requirement, the results clearly indicated lack of accuracy in a number of predicted quantities. One can also conclude that a considerable portion of the simulation's failure could have been addressed if a more rigorous and detailed preliminary cylinder study was completed. Such a study should also incorporate an extensive 3D component.

- With regards to the stability of the simulation, it was clearly demonstrated that the combined use of anti-aliasing and entropy limiting was a sufficient and necessary condition. Whilst the effect of higher spatial order in the stability of the simulation was not studied, it was demonstrated that entropy limiting was able to alleviate instabilities related to energy accumulation in highly vortical regions, a situation that requires unfeasible (from a computational resource point of view) quadrature degrees of anti-aliasing if no other stabilisation technique is present. Furthermore, both the CFL-controled variable time-stepping component of the entropy stability implementation and the convergence-based adaptive time-stepping implementation of PyFR proved ineffective in alleviating instabilities. Even more importantly, the use of a constant-sized time step was

proven necessary at some point in the simulation process.

- In regard to the specifics of the flow dynamics, it was shown that vortices shed by the cylinder remain coherent until their impingement on to the aerofoil. However, moderate agreement with experiment was found in the time-averaged flow field and turbulent intensity. The largest differences corresponded to the regions with large-scale vortical dynamics at their highest power, i.e. the cylinder wake. The results indicated that PyFR and the specific computational setup was less dissipative in nature than other LES CFD approaches. A relatively large disagreement with experiment was found in the cylinder C_p distribution. Cylinder drag and momentum loss in the cylinder wake of the time-averaged field were found to be overestimated. Also, lack of statistical convergence for the time averaging was strongly indicated by several observations. On the other hand, good agreement with experiment was found in the prediction of the cylinder Strouhal number. Good agreement with experiment was also found in the velocity power spectra around the $St_c = 0.2$ frequency region, particularly in the vicinity of the cylinder, where the coherence of the shed cylinder vortices was at its highest. Power levels at other frequencies reinforced the theory of the notion of potentially inadequate dissipative characteristics of PyFR.

In conclusion, it has been demonstrated that PyFR is able to simulate the Rod-Aerofoil problem stably without the use of sub-grid scale modelling, provided that suitable stabilising techniques are used and access to the required computational resource is present. It has also been demonstrated that the concept of entropy-bound limiters can provide adequate stability, whilst retaining the solution accuracy of the scheme in flow domain regions where stability is not a concern. However, greater insight into the interaction between solution limiting and CFL controlling is needed in order for a fully optimised setup to be reached. Furthermore, the resolution and meshing study outlined in chapter 5, resulted in a computational grid that was unsuccessful in capturing the flow dynamics across all scales, with the tonal peak position predictions in the frequency spectra of instantaneous velocity proving the only clear success. The size of the simulation along with the necessary use of anti-aliasing resulted in a computational cost that necessitated large-scale

computation. Also, the accuracy at which many averaged results were computed proves that the current setup provides limited success in the full resolution of the problem.

To the author's knowledge, the current chapter is the first application of DNS through high-order FR schemes stabilised via entropy limiting for the study of wake-body interaction. Based on the above remarks, the strengths of PyFR relative to other approaches are centred in the accuracy-preserving nature of the spatial scheme if fully resolved DNS is attempted and the inherent flexibility of the FR method to "numerically" adapt to different requirements. However, whilst arguably providing a suitable basis for further development, the weaknesses of the method relative to other approaches centre in the considerable sensitivity an prone to unstable behaviour and the lack of suitable sub-grid scale model that would potentially be able to capture more dynamics at lower computational cost.

Chapter 8

Conclusions And Future Research

8.1 Conclusions

The current thesis investigated the utility of the FR approach on the simulation of the flow around the rod-aerofoil configuration. The specific high-order scheme recovered, DG, was complemented with a variety of numerical strategies designed to stabilise the simulation. The entire numerical approach was computationally deployed through PyFR on clusters of varying size, comprised of Nvidia GPU accelerators. The main simulation resulted in approximately $140,000,000$ DoFs on 200 Nvidia P100 GPUs.

The conclusions that can be drawn from the research presented in this thesis are summarised below.

- **Resolution requirements.** The task of identifying a suitable grid resolution for the problem was proven of great importance, particularly if one considers the absence of a sub-grid scale model for the modelling of unresolved turbulent scales. It was found that boundary layer resolution influenced the large-scale vortex production of a 2D cylinder simulation in a critical manner. Thus, the identification of a suitable range for the boundary layer resolutions was made via 2D simulations. The effect of spanwise resolution was also found to have considerable influence on the flow dynamics. Furthermore, it was found that suitable adjustment of the spanwise resolution can alleviate a portion of the problems found when under-resolving the boundary layer in the 2D simulations. Consequently we can conclude that the rod-aerofoil required

considerable care to be taken in the setup and fine-tuning of the grid resolution before useful simulations could be considered. It was found that maintaining an aspect ratio of approximately 1.0 for the hexahedral elements in the wake of the cylinder and upstream of the aerofoil was required. However, it has been demonstrated that important work towards identifying the correct resolution requirements should still be carried out, as the final large-scale simulation exhibited clear lack of convergence in numerous cases. A more rigorous and exhaustive three-dimensional preliminary cylinder study has the potential to eliminate a number of inefficiencies

- **Stability of the simulation.** The rod-aerofoil problem required suitable strategies that ensured the stability of the simulation over a time period long enough for useful data to be acquired. It was found that employment of anti-aliasing of quadrature degree $Q_d = 11$ was a necessary but not sufficient condition for the required stability. The latter was achieved via the inclusion of local entropy stability. Furthermore, the employment of adaptive time stepping was found to alleviate portions of problematic behaviour such as violent impinging of vortices on the aerofoil surface but was not the answer in every occasion. It has been demonstrated that, despite the use of all the aforementioned methods, the inclusion of the entropy stability limiter was necessary in order to stabilise the simulation in a definitive manner. The utility of the novel approach of stabilising the solution via entropy limiting was confirmed in the case of an oblique shock but was also successfully extended to turbulent high Reynolds, low Mach number flows such as the flow around the rod-aerofoil problem. The combination of the two primary stabilisation techniques (anti-aliasing and entropy stability) allowed data of sufficient amount to be produced. The behaviour encountered also underlined the challenge of simulating this type of flow without the use of turbulence modelling.

- **Performance of the Simulation.** The utility of using the FR approach and, in particular, PyFR for the simulation of the rod-aerofoil problem was assessed and was found to be considerably lacking in certain areas. It was demonstrated that, in the presence of suitable computational resource, PyFR is successful in capturing a portion of the large-scale dynamics of the flow such as

the Strouhal number. However, as clearly demonstrated by the calculated C_p distribution on the cylinder, the method lacks significant accuracy at aspects of the flow that are very well documented, understood and "routinely" predicted well. It was also demonstrated that the approach exhibited limited success in the modelling of smaller-scale dynamics. Furthermore, greater pressure and momentum deficits than experiment were predicted. Results reflected that energy dissipation across turbulent scales was potentially inadequate compared to other LES approaches and experiment, as also shown in the power levels of the velocity spectra. A highly turbulent behaviour was observed across several positions in the flow domain, with energy levels consistently higher than experiment and other studies. This indicates that PyFR in its current form and the computational setup used, whilst successful in preserving the (qualitative) shape of the plots of various time-averaged quantities, is not as accurate as would be required for certain applications.

The general conclusion that can be drawn from the present work is that further research and fine-tuning of both the computational method and meshing is needed in order to adequately simulate the problem. Through the use of a stabilisation technique that is not *adhoc*, it is possible to stabilise the rod-aerofoil problem via high-order, FR under-resolved DNS however in a manner that does not result in satisfactory prediction quality for all of the flow quantities. The utility of the high-order FR, under-resolved DNS approach compared to other methods including LES simulations has not been proven. However, the unsuccessfully predicted quantities also provided a means of identifying areas for improvement, primarily related to momentum and pressure loss across wakes and turbulent energy dissipation. The sensitivity shown by the FR method in anti-aliasing, entropy stability and adaptive time-step control whilst simulating the rod-aerofoil problem, points towards a potential existence of a broad range of usable setups. As a result, continued research towards the identification of the optimum setup is justified. It is the opinion of the author that the primary strength of current high-order FR capability in terms of computational efficiency is the fact that high accuracy can be achieved for a given computational cost provided that a considerable amount of work is done in order to ensure that the simulation setup is optimum. The main weakness of the method is reduced

robustness compared to, for instance, an LES approach, with the latter providing a platform that can be more easily fine-tuned towards a successful simulation.

8.2 Future Research

The work presented in this thesis demonstrated a series of challenges involved in the simulation of complex, unsteady flow via high-order DNS. Ways of addressing these issues were assessed and deployed. As expected however, there remain a number of problems to be researched in the future in order to achieve results of greater quality. These problems include the following.

- Fine-tuning of the method. The simulations analysed in the present work demonstrated success in a limited number of flow characteristics such as Strouhal number prediction and velocity spectra power estimation, which indicates that a good platform for further research is provided. This research should aim to achieve a complete, industry-relevant, accurate and robust simulation and should be organised into a rigorous and systematic study that builds onto the already proven ability to predict cylinder vortex shedding at high Re. It should involve simple problems, such as the flow around a single cylinder, that gradually become more complicated and demanding.

- An extension of the grid resolution study that includes a full, 3D unstructured approach, different types of elements and a greater range in the boundary layer resolution strategies.

- Sub-grid scale modelling. The development and implementation of a sub-grid turbulence model for PyFR can potentially result in better small-scale prediction quality and energy dissipation characteristics. This strategy can also alleviate portion of the stability issues, reducing the need for resource-consuming stabilisation techniques.

- Near-wall boundary models. The development and implementation of a wall model can potentially result in lower boundary layer resolutions for a given solution quality thus resulting in smaller grids.

- Theoretical research into the dissipation properties of FR. The turbulence results presented in this thesis provided a first insight into the number and power of turbulent scales of the rod-aerofoil problem that are resolved via PyFR. There remains much scope for the relationship between desired turbulent levels and the numerical stability of the scheme to be investigated.

- A FR polynomial degree study. A study of whether the expected accuracy increase for a problem of constant size is observed with increasing polynomial order P. An investigation into the effect of the use of element types other than hexahedra can also be made. Such studies will also provide definitive answers regarding the optimum strong scaling performance that can be achieved by PyFR whilst simulating the rod-aerofoil problem.

It is the opinion of the author that the current thesis demonstrates the importance and potential benefits of continuing the research associated with PyFR in order for a simulation efficiency beneficial to the aerospace industry to be achieved.

Bibliography

[1] D. Kuechemann, *The aerodynamic design of aircraft*, Pergamon Press, 1978

[2] H.T. Huynh, *A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods*, AIAA paper, 4079, 2007

[3] A. Jameson, K. Ou, *50 years of transonic aircraft design*, Progress in Aerospace Sciences, Volume 47, Issue 5, p. 308–318, 2011

[4] M. R. Malik, D. M. Bushnell , *Role of Computational Fluid Dynamics and Wind Tunnels in Aeronautics R&D*, NASA technical paper, Volume 217602, 2012

[5] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D. Mavriplis, *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*, NASA, CR-218178, 2014

[6] N. Kroll, C. Rossow, *Current Status and Challenges in CFD at the DLR Institute of Aerodynamics and Flow Technology*, AIAA Presentation, 2015

[7] P.E. Vincent, A. Jameson, *Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists*, Mathematical Modelling of Natural Phenomena, 6(03), p. 97-140, 2011

[8] A. Harten, B. Engquist, S. Osher, S. Chakravarthy, *Uniformly high order essentially non-oscillatory schemes III*, Journal of Computational Physics, 71:231, 1987

[9] S. Lele, *Compact finite difference schemes with spectral-like resolution*, Journal of Computational Physics, 103, p. 16–42, 1992

[10] X. Liu, S. Osher, T. Chan, *Weighted essentially non-oscillatory schemes*, Journal of Computational Physics, 115, p. 200–212, 1994

[11] T. Barth, P. Frederickson, *High-order solution of the Euler equations on unstructured grids using quadratic reconstruction*, AIAA Paper, 90-0013, 1990

[12] F. Bassi, S. Rebay, *A high–order discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations*, Mathematics of Computation, 52, p. 411-435, 1989

[13] B. Cockburn, C.W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws ii: general framework*, Journal of Computational Physics, 131, p. 267-279, 1997

[14] W.H. Reed, T.R. Hill, *Triangular mesh methods for the neutron transport equation*, Los Alamos Report, LA-UR-73-479, 1973

[15] Y. Liu, M. Vinokur, Z.J. Wang, *Spectral difference method for unstructured grids I: Basic formulation*, Journal of Computational Physics, 216, p. 780-801, 2006

[16] D. Kopriva, J. Kolias, *A conservative staggered-grid Chebyshev multidomain method for compressible flows*, Journal of Computational Physics 125, 244, 1996

[17] F.D. Witherden, A.M. Farrington, P.E. Vincent, *PyFR: An Open Source Framework for Solving Advection-Difusion Type Problems on Streaming Architectures using the Flux Reconstruction Approach*, Computer Physics Communications, Volume 185, p. 3028-3040, 2014

[18] F.D. Witherden, *On the Development and Implementation of High-Order Flux Reconstruction Schemes for Computational Fluid Dynamics*, PhD Thesis, Imperial College London, Department of Aeronautics, 2015

[19] M.C. Jacob, J. Boudet, D. Casalino, M Michard, *A rod-airfoil experiment as benchmark for broadband noise modeling*, Theoretical and Computational Fluid Dynamics, 19, p. 171-196, 2005

[20] P. Castonguay, P.E. Vincent, A. Jameson, *A New Class of High-Order Energy Stable Flux Reconstruction Schemes for Triangular Elements*, Journal of Scientific Computing, Volume 47, p. 50-72, 2011

[21] P.L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, Journal of Computational Physics, Volume 43, 1981

[22] P. Castonguay, P.E. Vincent, A. Jameson, *Application of high-order energy stable flux reconstruction schemes to the Euler equations*, 49th AIAA Aerospace Sciences Meeting, volume 686, 2011

[23] *PyFR Project Web Site*, http://www.pyfr.org/, 2013

[24] M.H. Carpenter and C.A. Kennedy, *Fourth-Order 2N-Storage Runge–Kutta Schemes*, 1994

[25] C.A. Kennedy, M.H. Carpenter, *Low-storage, explicit Runge– Kutta schemes for the compressible Navier–Stokes equations*, Applied numerical mathematics, Volume 35, p. 177-219, 2000

[26] F. Johansson and others, *mpmath: A Python library for arbitrary-precision floating-point arithmetic (version 0.18)*, http://www.mpmath.org/, 2013

[27] M Bayer, *Mako: Templates for Python*, http://www.makotemplates.org/, 2013

[28] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, A. Fasih, *PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation*, Parallel Computing, http://www.makotemplates.org/, Volume 38, p. 157-174, 2012

[29] L Dalcin, *mpi4py: MPI for Python*, https://www.bitbucket.org/mpi4py, 2013

[30] A. Collette, *Python and HDF5*, O'Reilly Media, 2013

[31] A. Jameson, T. Baker, *Solution of the Euler equations for complex configurations*, AIAA Paper, 1929, 1983

[32] A. Henderson, *A parallel visualization application*, ParaView guide, Kitware Inc., 2007

[33] F. Magagnato, E. Sorgueven, M. Gabi, *Far field noise prediction by large eddy simulation and Ffowcs-Williams Hawkings analogy*, In Proceedings of the 9^{th} AIAA/CEAS Aeroacoustics Conference, 2003

[34] J.E. Ffowcs Williams, D.L. Hawkings, *Sound generation by turbulence and surfaces in arbitrary motion*, Phil. Trans. Roy. Soc. (London), volume 264(A), p. 321-342, 1969

[35] S. Peth, J.H. Seo, Y.J. Moon, M.C. Jacob, F. Thiele, *Computation of aerodynamic noise from rod wake-airfoil interactions*, European Conference on Computational Fluid Dynamics, 2006

[36] B. Greschner, F. Thiele, M.C. Jacob, D. Casalino, *Prediction of sound generated by a rod-airfoil configuration using EASM DES and the generalised Lighthill/FW-H analogy*, Computational Fluids, volume 32, 402413, 2008

[37] J. Berland, P. Lafon, F. Crouzet, F. Daude, C. Bailly, *Numerical insight into sound sources of a rod-airfoil flow configuration using direct noise calculation*, 16^{th} AIAA/CEAS Aeroacoustics Conference, 2011

[38] S. Galdeano, S. Barre, N. Reau, *Noise radiated by a rod-airfoil configuration using DES and the Ffowcs-Williams and Hawkings analogy*, 16^{th} AIAA/CEAS Aeroacoustics Conference, 2010

[39] A. Eltaweel, M. Wang, *Numerical simulation of broadband noise from airfoil-wake interaction*, 17^{th} AIAA/CEAS Aeroacoustics Conference, 2011

[40] J.C. Giret, A. Sengissen, S. Moreau, M. Sanjose, J.C. Jouhaud, *Prediction of the sound generated by a rod-airfoil configuration using a compressible unstructured LES solver and a FW-H analogy*, 33^{rd} AIAA Aeroacoustics Conference, 2012

[41] B. Agrawal, A. Sharma, *Aerodynamic Noise Prediction for a Rod-Airfoil Configuration using Large Eddy Simulations*, Conference Paper, AIAA, 2014

[42] B. Agrawal, A. Sharma, *Numerical analysis of aerodynamic noise mitigation via leading edge serrations for a rod–airfoil configuration*, International Journal of Aeroacoustics, volume 15, p. 734-756, 2016

[43] Y. Zhou, Z.J. Wang, *Absorbing boundary conditions for the Euler and Navier–Stokes equations with the spectral difference method*, Journal of Computational Physics, volume 229, p. 8733-8749, 2010

[44] C. Geuzaine, J.F. Remacle, *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*, International Journal for Numerical Methods in Engineering, volume 79, p. 1309-1331, 2009

[45] R.M. Kirby, S.J. Sherwin, *Aliasing errors due to quadratic nonlinearities on triangular spectral/hp element discretisations*, Journal of Engineering Mathematics, volume 56, p. 273-288, 2006

[46] G.J. Gassner, A.D. Beck, *On the accuracy of high-order discretizations for underresolved turbulence simulations*, Theoretical and Computational Fluid Dynamics, volume 27, p. 221-237, 2013

[47] Y. Lv, M. Ihme, *Entropy-bounded discontinuous Galerkin scheme for Euler equations*, Journal of Computational Physics, volume 295, p. 715-739, 2015

[48] Hairer, Ernst, Norsett, Syvert, Warner, Gerhard, *Solving Ordinary Differential Equations II*, Springer, p. 28 - 31, 1996

[49] Butcher, Wiley, *Numerical Methods for Ordinary Differential Equations*, 2^{nd} Revision, p. 309 - 314, 2008

[50] E. Jones, E. Oliphant, P. Peterson, *et al. SciPy: Open Source Scientific Tools for Python*, http://www.scipy.org/, 2001

[51] J.S. Park, M. Yu, C. Kim, Z. J. Wang *Comparative Study of Shock-Capturing Methods for High-Order CPR: MLP and Artificial Viscosity*, Conference Paper, The 8^{th} International Conference on Computational Fluid Dynamics, 2014

[52] J.S. Hesthaven, T. Warburton *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Springer Verlag New York, volume 54, 2008

[53] P. Castonguay, D.M. Williams, P.E. Vincent, A. Jameson, *Energy Stable Flux Reconstruction Schemes for Advection-Diffusion Problems*, Computer Methods in Applied Mechanics and Engineering, 2013

[54] A.L.F. Lima, E. Silva, A. Silveira-Neto, J.J.R. Damasceno, *Numerical simulation of two-dimensional flows over a circular cylinder using the immersed*

boundary method, Journal of Computational Physics, volume 189, p. 351-370, 2003

[55] A. Roshko, *Experiments on the flow past a circular cylinder at very high Reynolds number*, Guggenheim Aeronautical Laboratory, California Institute of Technology, 1960

[56] G. Bruschi, T. Nishioka, K. Tsang, R. Wang, *Drag Coefficient of a Cylinder*, 2003

[57] D. J. Tritton, *Physical Fluid Dynamics*, 2^{nd} Edition, Oxford Science Publications, 1988

[58] C. Norberg, *Pressure Forces on a Circular Cylinder in Cross Flow*, Bluff-Body Wakes, Dynamics and Instabilities, p. 275-278, 1993

[59] A. Berajeklian, L. Mydlarski, *Simultaneous velocity-temperature measurements in the heated wake of a cylinder with implications for the modeling of turbulent passive scalars*, Physics of Fluids, volume 23, 2011

[60] CD-adapco, https://mdx.plm.automation.siemens.com/

[61] Ansys Fluent, https://www.ansys.com/Products/Fluids/ANSYS-Fluent

[62] Ansys CFX, https://www.ansys.com/en-gb/products/fluids/ansys-cfx

[63] OpenFOAM User Guide, 2012, http://openfoam.com/docs/user/index.php

[64] A. Kosik, *The CFD simulation of the flow around the aircraft using OpenFOAM and ANSA*, 5^{th} ANSA and μETA International Conference, 2013

[65] L. P. Franca, S. L. Frey, *Stabilized finite element methods: II. The incompressible Navier-Stokes equations*, Computer Methods in Applied Mechanics and Engineering, volume 99, p. 209-233, 1992

[66] COMSOL CFD, https://www.comsol.com/cfd-module

[67] A. Meister, S. Ortleb, T. Sonar, M. Wirz, *A comparison of the Discontinuous-Galerkin- and Spectral-Difference-Method on triangulations using*

PKD polynomials, Journal of Computational Physics, volume 231, p. 7722-7729, 2012

[68] P.E. Vincent, A.M. Farrington, F.D. Witherden, A. Jameson, *An extended range of stable-symmetric-conservative Flux Reconstruction correction functions*, Comput. Methods Appl. Mech. Engrg, volume 296, p. 248-272, 2015

Appendix A

In this section details of the Rusanov Riemann solver used in the simulations of the present work will be given.

The inviscid interface flux f^{inv} is given by the following expression:

$$f^{inv} = \frac{\hat{n}}{2} \cdot (f_L^{inv} + f_R^{inv}) + \frac{s}{2}(u_L - u_R).$$

Here, s is an estimate of the maximum speed of sound

$$s = \sqrt{\frac{\gamma(p_L + p_R)}{\rho_L + \rho_R}} + \frac{1}{2} \mid \hat{n}_L \cdot (v_L + v_R) \mid .$$

Appendix B

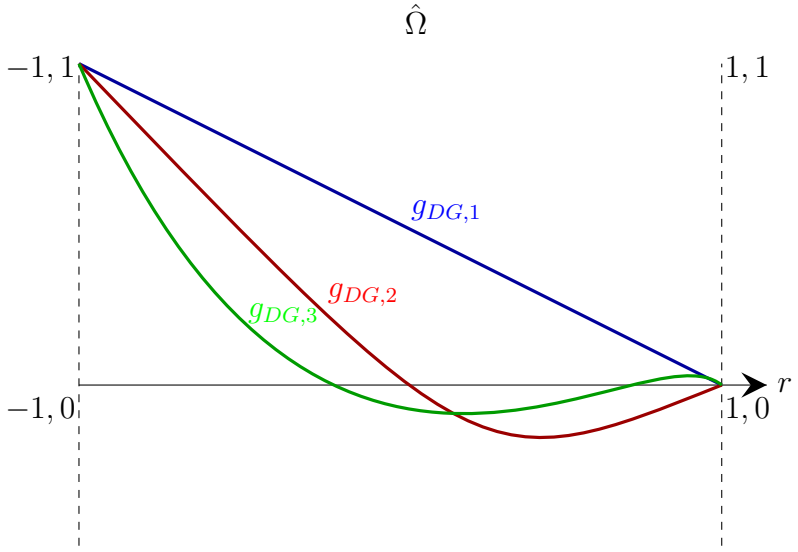
In this section, the one-dimensional form of the correction functions that recover the DG scheme used in the simulations of the present work will be given. The form follows the work of Huynh [2].

We assume that L_k represents a Legendre polynomial of degree k. The general form of the correction function that can recover the DG scheme, g_{DG}, is given below:

$$g_{DG,k} = \frac{(-1)^k}{2}(L_k - L_{k-1}).$$

In the above equation, $g_{DG,k}$ is the correction function of polynomial degree k.

The graphical form of a left-side correction function of the above type for three values of k is given in the figure below. The figure represents a transformed, reference 1D element $\hat{\Omega}$. The transformed axis is denoted as r in line with Chapter 2 of the present work.



1D FR correction functions that recover DG.

Additionally to the above and as a general note, a range of stable correction functions that recover additional schemes can be identified following the work of Vincent *et al.* [68].

We define $\hat{g}_L, \hat{g}_R$ as the left and right vectors of the correction coefficients in modal space:

$$\hat{g}_L = V^{-1} g_L, \hat{g}_R = V^{-1} g_R$$

with V the Vandermonde matrix

$$V[i][j] = L_j(r_i).$$

It has been proven in the work of Vincent *et al.* that a group of correction functions in 1D FR are stable if the following holds. We work in transformed, reference space:

$$\hat{g}_L = -(\hat{M} + \hat{Q})^{-1} \hat{L}.$$

$$\hat{g}_R = (\hat{M} + \hat{Q})^{-1} \hat{R}.$$

If L_i is a Legendre polynomial, then $\hat{M}$ is the modal Legendre mass matrix

$$\hat{M}[i][j] = \int_{-1}^{1} L_i L_j dr.$$

$\hat{Q}$ is a square matrix that satisfies

$$\hat{Q} = \hat{Q}^T,$$

$$\hat{Q}\hat{D} + \hat{D}^T \hat{Q}^T = 0,$$

$$\hat{Q} + \hat{M} > 0,$$

with

$$\hat{D} = V^{-1} D V$$

and

$$D[i][j] = \frac{dl_j}{dr}(r_i).$$

$\hat{L}$ and $\hat{R}$ are defined as

$$\hat{L}[i] = L_i(-1) = (-1)^i,$$

$$\hat{R}[i] = L_i(1) = 1.$$