

Vorticity-Based Polynomial Adaptation for Unsteady Flows

Syedramin Ghoreshilangrodi

A Thesis
In the Department
of
Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Mechanical Engineering) at
Concordia University
Montréal, QC, Canada

June, 2022

© Syedramin Ghoreshilangrodi, 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Syedramin Ghoreshilangrodi**
Entitled: **Vorticity-Based Polynomial Adaptation for Unsteady
Flows**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Mechanical Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____	Chair
<i>Dr. Zhibin Ye</i>	
_____	External Examiner
<i>Dr. Catherine Mavriplis</i>	
_____	Examiner
<i>Dr. Ali Nazemi</i>	
_____	Examiner
<i>Dr. Charles Basenga Kiyanda</i>	
_____	Examiner
<i>Dr. Ali Dolatabadi</i>	
_____	Supervisor
<i>Dr. Brian Vermeire</i>	

Approved by _____
Dr. Ivan Contreras,
Graduate Program Director

2022-08-05 _____
Dr. Mourad Debbabi,
Dean of Faculty

Abstract

Vorticity-Based Polynomial Adaptation for Unsteady Flows

Syedramin Ghoreshilangrodi, Ph.D.

Concordia University, 2022

Aerospace applications, including but not limited to the aerodynamics of slats and flaps, the aeroacoustics of complete landing gear configurations, and the performance prediction of air brakes, spoilers, fans, and turbines, benefit from high-fidelity scale resolving approaches such as direct numerical simulation and large eddy simulation. These approaches are shown to be accurate in complex unsteady flow regimes where current industry-standard Reynolds-averaged Navier-Stokes methods often fail. For example, recent research has demonstrated that direct numerical simulation can be utilized for analysis of complete jet engine low-pressure turbine cascades at their designed operating condition. High-order unstructured spatial discretizations such as flux reconstruction are particularly appealing for large eddy simulation of unsteady turbulent flows as they are capable of resolving the flow more efficiently; however, their computational cost remains relatively high, preventing their industrial use. Therefore, given the inherent computational cost of high-order methods, it is crucial to deploy strategies to minimize the overall computational cost of large eddy simulation while attaining comparable accuracy. A practical approach is to limit the use of high-order elements to regions where higher resolution is required for an accurate discrete approximation of the solution, reducing the total number of degrees of freedom. This can be achieved by locally increasing or decreasing the solution polynomial degree to adjust resolution and order of accuracy to achieve an accurate and cost-efficient simulation, a practice called polynomial adaptation or p -adaptation. Large eddy simulation can also benefit from the computational power of state-of-the-art hardware architectures by taking advantage of

parallel computing, which can achieve a significant speed-up by decomposing the domain and solving the problem concurrently. Parallel computing also provides a larger memory capacity and higher bandwidth, resulting in better performance. However, combining polynomial adaptation with a massively parallel computation can cause overhead because p -adaptation tends to change the degree of solution polynomials locally, leaving the computational domain unbalanced. To maintain parallel efficiency, dynamic load balancing techniques have been exploited. The current work introduces a novel dynamically load-balanced polynomial adaptation method for simulations of unsteady flows in the vicinity of complex geometries using the flux reconstruction scheme. This approach is applied to both two-dimensional and three-dimensional studies, while for the latter, we make use of implicit large eddy simulation, which relies on the dissipation error of the numerical scheme to act as a subgrid-scale model to dissipate the kinetic energy from the smallest turbulent scales in the flow.

We verify the utility of our dynamically load-balanced adaptation approach when applied to the arbitrary Lagrangian–Eulerian form of the compressible Navier–Stokes equations for a range of applications on moving and deforming domains. Specifically, we verify the arbitrary Lagrangian–Eulerian and dynamic load balancing implementation by performing simulations of an Euler vortex, and then illustrate the accuracy and efficiency of the adaptation routine by performing simulations of flow over an oscillating circular cylinder with two different flow settings, dynamic stall of a 2D NACA 0012 airfoil undergoing heaving and pitching motions, and flow over a vertical axis wind turbine composed of two NACA 0012 airfoils. We further illustrate the accuracy and efficiency of the routine when applied to transitional and turbulent flows by performing simulations of a shallow dynamic stall of a 3D SD 7003 airfoil undergoing heaving and pitching motions and transitional flow over a 3D circular cylinder. Results demonstrate that the dynamically load-balanced adaptation algorithm can track regions of interest, such as vortices and boundary layers, and yields a significant speed-up when applied to parallel simulations. We also demonstrate the scalability of the algorithm and the capability of dynamic load balancing technique to distribute uniform computational load among processors.

*"My life lasts but a day or two, and fast
Sweeps by, like torrent stream or desert blast,
Howbeit, of two days I take no heed,—
The day to come, and that already past."*

Omar Khayyam

Dedication

To my wife and partner in life, Sanaz Keshavarzi, whose calm glance blows away my perturbations and helps me find peace in the chaos of life as if time has stopped momentarily. To you, who taught me patience, kindness, forgiveness, and to worry about others even if they are total strangers. To you, whose smile gives me the world and whose tears make it fall apart. Thank you for your unconditional support during this journey.

Acknowledgements

I would like to express my most profound appreciation to my supervisor, Dr. Brian Vermeire, who did not only guide me immensely throughout my studies but also taught me how to be a better person and understand others. His high-order computational fluid dynamics knowledge inspired me to pursue my Ph.D. studies. I would not be standing here today if it was not because of him. I will never stop trying to emulate him. I would also like to acknowledge my fellow members at the Computational Aerodynamics Laboratory at Concordia University, particularly Mohsen Hamed, Carlos Pereira, Siavash Hedayati Nasab, and Hamid Karbasian, for their insights and assistance.

I am deeply indebted to my wife, Sanaz, as I could not have undertaken this journey without her. Thank you for being there the entire way. Words cannot express my gratitude to my aunt, Dr. Razieh Ghoreishi, for her pure love and support throughout every step of my adult life. You taught me how to see the best in people and how to care for others. I would also like to thank Setareh Farsi and Amin Mahmoudi for their emotional support during this time and the time we enjoyed together.

I cannot write this acknowledgement without mentioning my grandmother, Zahra Alipour Sabetray. Your memories are so vivid that sometimes I forget you are gone. I still can see you looking at me over your glasses with your cute smile while trying to hide it by covering your face with your hands. I learned how to overcome difficult moments of life without being folded. I learned it from your night cries at your prayer mat and your grins at the breakfast table the next morning.

This work would not have been possible without the support of the Concordia Graduate Excellence Doctoral Fellowship for my Ph.D. studies. This research was enabled in part by support provided by Calcul Quebec (www.calculquebec.ca), WestGrid (www.westgrid.ca), and Compute Canada (www.computeCanada.ca) via a Resources for Research Groups allocation.

Lastly, I would like to express my deepest gratitude to my parents, Hadi Ghoreishi and Fatemeh Kaheh, and my sister, Delaram Ghoreishi, for their love and encouragement at every step of my life. I still remember my first day of school when you stood behind the school entrance door the whole day to give me a hug in case I needed you. You taught me to enjoy making others happy. You taught me to accept my failures, not give up, and start over.

Contents





List of Figures	xii
List of Tables	xx
Acronyms	xxiii
Nomenclature	xxvi
1 Introduction	1
1.1 Motivation	1
1.2 Turbulence in Fluid Mechanics	4
1.3 Simulation Approaches	6
1.4 High-Order Methods	9
1.5 Parallel Computing	11
1.6 Research Objectives and Contributions	11
1.7 Thesis Outline	12
2 Governing Equations	14
2.1 Conservation of Mass	14
2.2 Conservation of Momentum	15
2.3 Conservation of Energy	17
2.4 General Conservation Law	19
2.5 Euler and Navier-Stokes Equations	19
2.6 Arbitrary Lagrangian-Eulerian Formulation	21

2.7	Advection Equation	22
2.8	Diffusion Equation	23
3	Numerical Schemes	24
3.1	The Flux Reconstruction Scheme	24
3.1.1	Linear Advection	24
3.1.2	Linear Diffusion	34
3.1.3	Multidimensional Extension	37
3.1.4	Arbitrary Lagrangian Eulerian Extension	41
3.1.5	Advantages of Flux Reconstruction	44
3.2	Temporal Schemes	45
4	Dynamically Load Balanced Adaptation Algorithm	50
4.1	Adaptation Techniques	50
4.1.1	Adjoint-Based Adaptation Indicators	51
4.1.2	Truncation-Error-Based Adaptation Indicators	52
4.1.3	Feature-Based Adaptation Indicators	56
4.2	Vorticity-Based Polynomial Adaptation	58
4.3	Load Balancing	59
4.3.1	Introduction	59
4.3.2	ParMETIS	60
4.3.3	Implementation	62
5	Verification and Validation Test Cases	65
5.1	ALE Verification	65
5.2	DLB Verification	67
5.3	Validation: 2D Circular Cylinder	77
5.4	Validation: SD 7003 Airfoil	84
6	Polynomial Adaptation for Moving and Deforming Domains	98
6.1	Oscillating Circular Cylinder	99
6.1.1	Introduction	99

6.1.2	Computational Details	101
6.1.3	Numerical Results	103
6.2	2D Dynamic Stall of a NACA 0012 Airfoil	113
6.2.1	Introduction	113
6.2.2	Computational Details	117
6.2.3	Numerical Results	120
6.3	2D Vertical Axis Wind Turbine	132
6.3.1	Introduction	132
6.3.2	Computational Details	133
6.3.3	Numerical Results	135
6.4	Conclusion	139
7	Dynamically Load Balanced Polynomial Adaptation of Turbulent Flows	143
7.1	3D Dynamic Stall of a SD 7003 Airfoil	144
7.1.1	Computational Details	144
7.1.2	Numerical Results	145
7.2	Turbulent Flow Over a 3D Circular Cylinder	151
7.2.1	Introduction	151
7.2.2	Computational Details	153
7.2.3	Numerical Results	153
7.3	Conclusion	160
8	Conclusions and Future work	165
	Appendix A DLB: Element Weights	183
	Appendix B Experimental Tangential Force Coefficient [1]	184

List of Figures

1.1	Boeing’s sub-scale X-48B blended wing body aircraft without distributed propulsion flew over the edge of Rogers Dry Lake at Edwards Air Force Base during its fifth flight on Aug. 14, 2007 [2].	3
1.2	NASA N3-X blended wing body aircraft concept with turbo-electric distributed propulsion system [3].	4
1.3	The double bubble D8 aircraft concept with engines flush mounted on the top rear of the fuselage, designed based on a concept known as Boundary Layer Ingestion (BLI), claimed to achieve a 36% reduction in fuel burn from its configuration [4]. This aircraft is designed to fly at Mach 0.74 and carry 180 passenger for domestic flights [5].	5
1.4	Diagram of the turbulent kinetic energy cascade. Blue lines denotes modelling and orange lines denotes simulation.	7
3.1	Spatial discretization of a one-dimensional domain.	25
3.2	Schematic representation of solution polynomials along with the discontinuous and continuous flux functions.	27
3.3	One-dimensional nodal basis functions with different polynomial orders using Gauss points (black points).	28
3.4	Right Radau polynomials necessary for the $K = 0$ to $K = 5$ numerical schemes.	31
3.5	Exact and approximate solutions for a one-dimensional linear advection test case with an initial solution distribution $u_0 = \sin(2\pi x)$ at $t = 1$ for different polynomial degrees using a four-element mesh.	33

3.6	An example of a $K = 2$ quadrilateral element and a $K = 3$ triangular element including mapping points (grey), solution nodal basis points (red), flux points (blue), and Riemann points (green) on the interface between the elements. Elements are shown separated for clarity.	39
3.7	Two-dimensional nodal basis functions for a degree $K = 3$ polynomial using Gauss points.	41
3.8	Two-dimensional and three-dimensional schematic view of solution points for a $K = 2$ degree polynomial for different reference element types.	45
4.1	Schematic view of the initial mesh partition for a polynomial adaptive simulation with an unbalanced computational load distribution.	60
4.2	Schematic view of an unbalanced mesh partition (top) with highly  , moderately  , lightly  , and barely  expensive regions along with a balanced mesh (bottom) with equal computational load distribution for a polynomial adaptive simulation. Black lines denote the partition boundaries.	61
4.3	Schematic view of a mesh with 16 elements partitioned over four processors.	62
5.1	Mesh deformation for the isentropic vortex case on the 20×20 element mesh.	68
5.2	Density contours for the isentropic vortex case on the 20×20 element mesh with solution polynomials of degree $K = 5$	69
5.3	Convergence plot of the density error for the deforming and static isentropic vortex cases.	71
5.4	Contours of density, polynomial distribution, and the corresponding repartitioned mesh with different load balancing schemes at several time instants for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over 4 processors.	74
5.5	Plots of averaged weight distribution W_C , averaged maximum computational burden $\bar{W}_{C,max}$, simulation time per iteration T_{itr} , and weight fluctuation ratio W_f for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over 4 and 32 processors.	75

5.6	Plot of the speed-up of the dynamic load-balanced adaptive algorithm for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over different numbers of processors. The dashed line represents the ideal efficiency of one.	76
5.7	20×20 centre-portion of the isentropic vortex element meshes.	78
5.8	Density contours, polynomial degree distributions, and mesh partitions for the isentropic vortex case on the three-dimensional $400 \times 400 \times 3$ element mesh with adaptive solution polynomial of degree $K = 1 - 5$	79
5.9	Plot of the compute time per iteration for the isentropic vortex cases on the three-dimensional 400×400 element mesh.	81
5.10	Circular cylinder quadrilateral structured mesh with 4342 elements (mesh 1), on the top, and circular cylinder triangular structured mesh with 8684 elements (mesh 2), on the bottom.	85
5.11	Solution polynomial distribution for three-level adaptive computation ($K1 - K3$) based on vorticity magnitude indicator corresponding to maximal lift for the circular cylinder with $Re = 150$ and $M = 0.1$ for mesh 1 (on the top) and mesh 2 (on the bottom).	86
5.12	Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations for mesh 1, and three-level adaptive computation $K = 1 - 3$ for mesh 2 based on the vorticity magnitude indicator for the circular cylinder with $Re = 150$ and $M = 0.1$	87
5.13	Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations for mesh 1, and three-level adaptive computation $K = 1 - 3$ for mesh 2 based on the vorticity magnitude indicator for the circular cylinder with $Re = 150$ and $M = 0.1$	89
5.14	Drag and lift coefficient profiles of uniform and adaptive simulations for the circular cylinder with $Re = 150$, $M = 0.1$, and quadrilateral elements.	90
5.15	Drag and lift coefficient profiles of mesh 1 and mesh 2 adaptive simulations for the circular cylinder with $Re = 150$ and $M = 0.1$	91

5.16	Drag coefficient profile of adaptive simulations of mesh 1 with different Reynolds numbers for the circular cylinder with $M = 0.1$	92
5.17	SD 7003 airfoil quadrilateral structured mesh with 8190 elements.	93
5.18	Solution polynomial distribution for three-level adaptive computation ($K = 1 - 3$) based on vorticity magnitude indicator corresponding to maximal lift for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$	94
5.19	Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations based on the vorticity magnitude indicator for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$	95
5.20	Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations based on the vorticity magnitude indicator for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$	96
5.21	Drag and lift coefficient profiles of uniform and adaptive simulations for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$	97
6.1	Map of vortex synchronization patterns near the fundamental lock-in region [6]. S represents a single vortex, P denotes a vortex pair, and P + S indicates combination of a vortex pair and a single vortex shed at each oscillation cycle.	101
6.2	Schematic of major vortex patterns observed near the fundamental lock-in region adapted from Williamson and Roshko [6]. The dashed circle shows the vortices shed in one complete oscillation cycle.	102
6.3	Circular cylinder quadrilateral structured mesh with 2668 elements.	104
6.4	Polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the time corresponding to maximal lift for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$.	105
6.5	Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$	106

6.6	Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$	107
6.7	Drag and lift coefficient profiles of the adaptive and reference simulations performed by Guilmineau et al. for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$	110
6.8	Drag and lift coefficient profiles of uniform and adaptive simulations for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$	111
6.9	Solution polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the time of maximal lift for the oscillating circular cylinder with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$	113
6.10	Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the oscillating circular cylinder with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$	114
6.11	Non-dimensional z-component of the vorticity for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on vorticity magnitude indicator for the oscillating circular cylinder with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$	115
6.12	Degrees of freedom profiles for the uniform and adaptive simulations for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, $f_e/f_s = 1.1$ on the top, and with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$ on the bottom.	116
6.13	Plots of vertical location $y(t)$, vertical velocity $y'(t)$, oscillation pitch angle $\theta(t)$, and effective angle of attack $\alpha_e(t)$ for the pitching and heaving NACA 0012 airfoil with $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$. Dashed lines show the location of $t = 1T_e/4$, $t = 2T_e/4$, and $t = 3T_e/4$	118
6.14	The NACA 0012 unstructured mesh with 2722 quadrilateral and triangular elements. Dimensions are given in multiples of the airfoil chord length. . . .	119

6.15	Polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the time corresponding to maximal lift for the pitching and heaving NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	119
6.16	Contours of spanwise vorticity of the B090 case from Moriche et al. [7] (a, d, g, and j) [reproduced with permission from Cambridge University Press], of the $K = 5$ deforming mesh NACA 0012 case (b, e, h, and k), along with the mesh deformation (c, f, i, and l) at four time instants.	122
6.17	Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the heaving and pitching NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	123
6.18	Non-dimensional z-component of the vorticity for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on vorticity magnitude indicator for the heaving and pitching NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	124
6.19	Drag and lift coefficient profiles of the uniform $K = 1 - 5$ and the B090 case performed by Moriche et al. [7] for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$. Dashed lines correspond to $t = 1T_e/4$, $t = 2T_e/4$, and $t = 3T_e/4$	126
6.20	Drag and lift coefficient profiles of the uniform and adaptive simulations for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	127
6.21	Contours of spanwise vorticity (left), lift coefficient (middle), effective angle of attack (top-right), and airfoil vertical velocity (bottom-right) at several time instants for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	131
6.22	Degrees of freedom profile of the uniform and adaptive simulations for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	132

6.23	Straight-bladed, NACA0012, Vertical-axis wind turbine unstructured mesh with 16081 quadrilateral elements and $\alpha_0 = 0^\circ$ captured at its first-quarter turn at the time corresponding to $\theta = 90^\circ$	134
6.24	Polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator in the 4^{th} revolution at the time corresponding to $\theta = 1620^\circ$ for the vertical-axis wind turbine composed of two NACA 0012 blades.	137
6.25	Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator in the 4^{th} revolution at the time corresponding to $\theta = 1620^\circ$ for the vertical-axis wind turbine composed of two NACA 0012 blades.	139
6.26	Tangential force coefficient profiles corresponding to the 1^{st} cycle of the uniform $K = 5$ simulation and the previous numerical study performed by Kanner et al. [8], for the second blade of the vertical-axis wind turbine with $\alpha_0 = -2$. .	140
6.27	Tangential force coefficient profiles corresponding to the 4^{th} cycle of the uniform $K = 5$ simulation and the previous experimental study performed by Strickland et al. [1, 9], for a blade of the vertical-axis wind turbine with $\alpha_0 = 0$	140
6.28	Tangential force coefficient profiles of the 4^{th} cycle of the uniform and adaptive simulations for the vertical-axis wind turbine composed of two NACA 0012 blades.	141
6.29	Degrees of freedom profile over four cycles of the uniform and adaptive simulations for the vertical-axis wind turbine composed of two NACA 0012 blades.	141
7.1	The SD 7003 mesh with 56000 hexahedral elements. Dimensions are given in multiples of the airfoil chord length.	146
7.2	Non-dimensional velocity magnitude at different time instants for the adaptive dynamic load-balanced heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$	148

7.3	Contours of Q-criterion colored by velocity magnitude at the time instant of $1T_e/4$ for the adaptive dynamic load-balanced heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.	148
7.4	Contours of Q-criterion colored by velocity magnitude, and polynomial distribution for the adaptive dynamic load-balanced heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.	150
7.5	Phase averaged x-component of the velocity profile at different chord locations and time instants (solid lines correspond to this study, and the crosses correspond to the experiment [10]) for the heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.	152
7.6	Three-dimensional circular cylinder structured mesh with 78176 hexahedral elements.	154
7.7	Contours of Q-criterion, polynomial distribution, and dynamically balanced mesh partition for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the times corresponding to maximal and minimal lift for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.	156
7.8	Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive load balanced $K = 1 - 5$ computations based on the vorticity magnitude indicator at the times corresponding to maximal lift for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.	157
7.9	Contours of Q-criterion colored by velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive load-balanced $K = 1 - 5$ computations based on the vorticity magnitude indicator at the times corresponding to maximal lift for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.	158
7.10	Drag and lift coefficient profiles of the uniform and adaptive simulations for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.	163
B.1	Tangential force coefficient profiles corresponding to the 4 th cycle of the experimental study performed by Strickland et al. [1] for a blade of a two-bladed wind turbine at a tip speed ratio of 0.75 with $\alpha_0 = 0$.	185

List of Tables

1.1	Comparison of spatial discretization methods, adapted from Hesthaven and Warburton [11], where ✓ represents the capability of the methods, ✗ indicates that the method is incapable of solving such problems, and (✗) reflects inefficiency or limitations.	10
3.1	Legendre polynomials used in the current work.	31
3.2	L_2 norm of the solution error for a one-dimensional linear advection test case with an initial solution distribution $u_0 = \sin(2\pi x)$ at $t = 1$ for different polynomial degrees and meshes.	34
3.3	Number of degrees of freedom for different element types for degree polynomial K	42
3.4	Butcher tableau for a general Runge-Kutta scheme used in the current study.	47
3.5	Butcher tableaus for the second two-stage and three-stage third order singly diagonally implicit Runge-Kutta methods.	48
3.6	Butcher tableaus for Runge-Kutta methods.	49
4.1	Arrays of <code>xadj</code> , <code>adjncy</code> , and <code>vtxdist</code> for the schematic mesh with 16 elements shown in Figure 4.3.	63
5.1	Density error for the deforming and static isentropic vortex cases.	70
5.2	Load balancing schemes employed for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$	73

5.3	Numerical values of $\bar{W}_{C,max}$, W_f , and T_{itr} for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over 4 and 32 processors.	73
5.4	Scalability metrics of the LB2 and LB3 schemes for the isentropic vortex case on the two-dimensional 40×40 element mesh.	73
5.5	Density error, mean number of degrees of freedom, simulation time per iteration, and speed-up factor for the isentropic vortex case on the three-dimensional $400 \times 400 \times 3$ element mesh.	80
5.6	Computational meshes employed for the circular cylinder the circular cylinder with $Re = 150$ and $M = 0.1$	84
5.7	Numerical values of ΔC_L , ΔC_D , \bar{C}_D , St , T_w , and error percent for the circular cylinder with $Re = 150$ and $M = 0.1$	84
5.8	Numerical values of ΔC_L , ΔC_D , \bar{C}_D , St , T_w , and error percent for the circular cylinder with $Re = 150$ and $M = 0.1$	94
6.1	Numerical values of \bar{C}_D , C_{Lrms} , and St for a fixed circular cylinder at $Re = 185$.103	
6.2	Numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , and error percent for the oscillating circular cylinder at $Re = 185$ with $A/D = 0.2$ and $f_e/f_s = 1.1$. . .	109
6.3	Numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , and error percent for the oscillating circular cylinder at $Re = 200$ with $A/D = 1$ and $f_e = 0.0193$. . .	112
6.4	Numerical values of \bar{C}_D , C_{Drms} , \bar{C}_L , C_{Lrms} , St , \overline{DOF} , and error percent for the oscillating NACA 0012 at $Re = 1000$ with $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$	121
6.5	Numerical values of C_T , C_P , and \overline{DOF} averaged over the fourth cycle with related error percent for the vertical-axis wind turbine composed of two NACA 0012 blades and $\alpha_0 = 0$	137
7.1	Computation time of the dynamic load-balanced adaptive algorithm for the heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$	150

7.2	Scalability metrics of the dynamic load-balanced adaptive algorithm for the heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$	150
7.3	Numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , T_{itr} and S_A averaged over the 100 convective times for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$	160
A.1	Element weight W_E defined for different element types and degree polynomials, non-dimensionalized based on the weight of a $K = 1$ triangular element. . . .	183

Acronyms

ACI	Airports Council International
ALE	Arbitrary Lagrangian–Eulerian
BLI	Boundary Layer Ingestion
BR1	First Method of Bassi and Rebay
BR2	Second Method of Bassi and Rebay
CANSO	Civil Air Navigation Services Organisation
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewey
CPI	Cycles Per Instruction
CPU	Central Processing Unit
CSF	Compressed Storage Format
DG	Discontinuous Galerkin
DLB	Dynamic Load Balancing
DNS	Direct Numerical Simulation
DOF	Degrees of Freedom
DRAM	Dynamic Random Access Memory
ENO	Essentially Non-Oscillatory
ERA	Environmentally Responsible Aviation
ERK	Explicit Runge-Kutta
EU	European Union
EV	Euler Vortex
FDM	Finite Difference Methods

FEM	Finite Element Methods
FLOP/s	Floating Point Operations per second
FR	Flux Reconstruction
FVM	Finite Volume Methods
GARDN	Green Aviation Research & Development Network
GMRES	Generalized Minimal Residual Method
HORUS	High-ORder Unstructured Solver
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
ICCAIA	International Coordinating Council of Aerospace Industries Associations
ILES	Implicit Large Eddy Simulation
LCP	Lifting Collocation Penalty
LDG	Local Discontinuous Galerkin
LES	Large Eddy Simulation
LEV	Leading-Edge Vortex
MAV	Micro Air Vehicle
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
PDE	Partial Differential Equation
PETSc	Portable, Extensible Toolkit for Scientific Computing
PIV	Particle Image Velocimetry
RAM	Random-Access Memory
RANS	Reynolds-Averaged Navier-Stokes
RK _{4,4}	Four-Stage Fourth Order Runge-Kutta Scheme
RMS	Root Mean Square
RPKs	Revenue Passenger-Kilometres
SD	Spectral Difference
SDIRK	Singly Diagonally Implicit Runge-Kutta
SG	Staggered Grid
SGS	Subgrid-Scale

SV	Spectral Volume
TSR	Tip Speed Ratios
US	United States
VAWT	Vertical Axis Wind Turbine
VIV	Vortex-Induced Vibration
WENO	Weighted Essentially Non-Oscillatory

Nomenclature

English Letters

a	Characteristic wave speed
\mathbf{a}	Acceleration vector
A	Oscillation amplitude
\mathbf{A}	Advection velocity
A_f	Frontal area
\mathbf{b}	Vector of temporal scheme weight coefficients
c	Chord length of the airfoil
\mathbf{c}	Vector of temporal scheme intermediate time steps
c_p	Specific heat coefficient at constant pressure
c_v	Specific heat coefficient at constant volume
C	Processor core
C_D	Drag coefficient
C_L	Lift coefficient
C_P	Power coefficient
C_T	Tangential force coefficient
D	Diameter
D_f	Diffusion coefficient
e	Total energy per unit mass
e_i	Internal energy per unit mass
e_k	Kinetic energy per unit mass
E	Total energy

E_k	Kinetic energy
E_s	Simulation efficiency
f	Element face
\hat{f}	Corrected continuous flux
f_b	Blending function
f^c	Common Riemann flux
f_e	Oscillation frequency
f_s	Vortex shedding frequency
f_K	Reduced frequency
\mathbf{F}	Vector of fluxes
\mathbf{F}_b	Vector of body forces
F_D	Drag force
\mathbf{F}_e	Vector of inviscid fluxes
\mathbf{F}_g	Vector of ALE fluxes
F_L	Lift force
\mathbf{F}_v	Vector of viscous fluxes
\mathbf{F}_Ω	Vector of forces acting on the control volume
g	Correction function
h_j	Element width in physical space
\mathbf{I}	Identity matrix
I_P^K	Interpolation operator from fine mesh to a coarse mesh
\bar{I}_P^K	Residual transfer operator from order P to K
J	Jacobian matrix
k	Thermal conduction coefficient
K	Polynomial degree
K_m	Maximum polynomial degree used in a simulation
K_{min}	Minimum polynomial degree used in a simulation
L	Characteristic length
L_c	Edge length of a cubic domain
L_R	Reference domain interval

m	System mass
M	Mach number
$\hat{\mathbf{n}}$	Outward normal vector to the surface
N	Conserved property within a system
$N_{d,C}$	Number of local element edges
N_{DOF}	Number of degrees of freedom
N_E	Number of elements
$N_{E,C}$	Number of local elements
N_g	Number of mapping points
N_{itr}	Number of iterations
N_n	Number of nodes
N_p	Number of processors
p	Pressure
P_K	Legendre polynomials of degree K
Pr	Prandtl number
q	Dynamic pressure
\mathbf{q}	Vector of surface heat flux per unit area
Q	Q-criterion
\dot{Q}	Net rate of heat added to the system
r	Specific gas constant
r_{max}	Upper limit of the blending function
r_{min}	Lower limit of the blending function
$r_{\mathbf{x}}$	Distance of the point \mathbf{x} from the mesh origin
R	Residual
R_b	Wind turbine rotor radius
R_c	Characteristic vortex radius
R_{ent}	Entropy residual
R_L	Left Radau polynomials
R_R	Right Radau polynomials
Re	Reynolds number

Re_b	Blade Reynolds number
s	Number or temporal scheme stage
S	Control surface
$\overline{\overline{\mathbf{S}}}$	Strain rate tensor
S_A	Adaptive simulation speed-up factor
S_f	Element face area
S_N	Speed-up factor based on a one node simulation
S_P	Parallel speed-up
S_t	Strouhal number
S_v	Vortex strength
St	Strouhal number
t	Time
t_c	Convective time
T	Temperature
T_A	Compute time of an adaptive simulation
T_C	Compute time of a parallel simulation on C cores
T_e	Oscillation period
T_{int}	Time interval length
T_{itr}	Simulation time per iteration
T_N	Compute time of a parallel simulation on one node
T_q	Torque
T_s	Vortex shedding period
T_S	Compute time of a serial simulation
T_U	Compute time of a uniform simulation
T_w	Wall clock time
u	Solution/conserved variable
\mathbf{u}	Vector of solution/conserved variables
\hat{u}	Corrected continuous solution
\tilde{u}	Non fully time-converged solution
u_0	Initial solution

\mathbf{u}_0	Vector of initial solution
u^c	Common solution
u_e	Exact solution
U_b	Blade velocity
U_∞	Free-stream velocity
v	Velocity
v'	Fluctuating components of the velocity
\bar{v}	Time-averaged velocity
\mathbf{v}	Velocity vector
\mathbf{v}_f	Velocity induced by the oscillating function
\mathbf{v}_g	Vector of grid velocity
\dot{W}	Net rate of work done on the system
W_C	Weight distribution on a processor
$\bar{W}_{C,max}$	Maximum computational burden on a processor
W_E	Element weight
W_f	Weight fluctuation ratio
x	Cartesian coordinate in physical space
\mathbf{x}	Coordinate vector in physical space
x_{cp}	Location of center of pressure
\mathbf{x}_g	Coordinate vector of mapping points
x_L, x_R	One-dimensional element interfaces
\mathcal{D}	Derivative matrix
\mathcal{S}	Entropy

Greek Letters

α	Angle of attack
$\hat{\alpha}$	Constant lifting coefficient
α_0	Blade offset pitch angle
α_e	Airfoil effective angle of attack
γ	Specific heat ratio

δ_{ij}	Kronecker delta
δ	Element correction field
ΔC_D	Drag peak fluctuation
ΔC_L	Lift peak fluctuation
Δt	Time step size
Δx	Grid spacing
Δx_{max}	Maximum mesh dimension of the element
ϵ	Adaptation constant resolution threshold
ϵ_h	Discretization error
ϵ_{itr}	Iteration error
ε	Kinetic energy dissipation rate
$\varepsilon_{\bar{C}_D}$	Mean drag coefficient error
$\varepsilon_{\bar{C}_L}$	Mean lift coefficient error
ε_{C_P}	Power coefficient error
ε_{C_T}	Sectional tangential force coefficient error
η	Amount of conserved property per unit mass
$\hat{\eta}$	Kolmogorov length scale
θ	Oscillation pitching angle
θ_0	Mean pitch angle value
θ_e	Pitch amplitude
θ_r	Ramp-up angular displacement
ϑ	Adaptation constant coefficient
κ	Maximum non-dimensional vorticity magnitude
$\bar{\kappa}$	Wavenumber
λ	Wavelength
λ_b	Tip speed ratio
μ	Dynamic viscosity
ξ	Cartesian coordinate in reference space
ξ	Coordinate vector in reference space
ρ	Density

ρ_∞	Free-stream density
σ	Density error
$\boldsymbol{\sigma}$	Stress tensor
σ_{ii}	Normal stress acting on a plane perpendicular to the i -axis
τ	Truncation error
$\boldsymbol{\tau}$	Viscous shear stress
τ_{ij}	Viscous stress acting on a plane perpendicular to the i - in direction of the j -axis
τ_{max}	Desired maximum threshold
ϕ	Nodal basis function
ϕ_e	Oscillation phase shift
$\bar{\omega}$	Non-dimensionalized counterpart of vorticity magnitude
$\boldsymbol{\omega}$	Vector of vorticity
$\hat{\omega}$	Wind turbine angular velocity
ω_r	Ramp-up angular velocity
Ω	Control volume
$\overline{\overline{\Omega}}$	Vorticity tensor

Operators

\cdot	Dot product
\circ	Hadamard product
\times	Vector product
\otimes	Tensor product
\oplus	Direct sum
\prod	Product operator
∇	Divergence operator
\mathcal{R}	Partial differential operator
tr	Trace
$\bar{\bullet}$	Averaged value
$\tilde{\bullet}$	Phase averaged value
\bullet^T	Transpose

$ \bullet $	Magnitude
$\ \bullet\ '$	First derivative
$\ \bullet\ _2$	L_2 norm
$\ \bullet\ _\infty$	Infinite norm

Superscripts

h	Discrete approximation
m	Temporal scheme intermediate stage index
t	Current time level
$t + 1$	Next time level
$*$	Non-dimensional

Subscripts

j	Element index
k	Solution point index
l	Flux point index
L	Left boundary of the element
m	Temporal scheme intermediate stage index
rms	Root mean square value
R	Right boundary of the element
x, y, z	Cartesian directions
δ	Final numerical solution

Chapter 1

Introduction

This work introduces a novel dynamically load-balanced polynomial adaptation method for simulations of unsteady flows in the vicinity of complex geometries. We start by giving motivation for this study and explaining the need for improvement in the field of Computational Fluid Dynamics (CFD) in the aerospace industry. We then outline the general characteristics of turbulence. Furthermore, we present common simulation approaches and explain why Large Eddy Simulation (LES) is particularly appealing for aerodynamics analysis. Then we will discuss high-order methods used for LES, particularly the Flux Reconstruction (FR) scheme, which is the numerical method used in the current work. Finally, we discuss the necessity of parallel computing for CFD.

1.1 Motivation

The environmental impacts of aviation have been rising since the adoption of jet engines on commercial aircraft. Global demand for air transportation is proliferating. Although the COVID-19 outbreak caused a 47 percent decline in industry-wide Revenue Passenger-Kilometres (RPKs) within two years [12], it is anticipated to be a temporary downturn. Based on the International Civil Aviation Organization (ICAO) annual report, 4.5 billion passengers were carried on 38.3 million flights in 2019 [13], and by 2040, the International Air Transport Association (IATA) predicts a 200 percent growth in the yearly number of flights [12]. This demand growth can potentially raise the CO₂ and other greenhouse gases emitted by aviation

industry by up to 400 percent [14]. To prevent this drastic emission increase and to move forward to complete decarbonisation, a collective commitment was announced by the Airports Council International (ACI), Civil Air Navigation Services Organisation (CANSO), IATA, and International Coordinating Council of Aerospace Industries Associations (ICCAIA) on behalf of the international air transport industry to decrease carbon emissions to 50 percent of 2005 levels by 2050 [15]. In line with this decision, the European Union (EU), the United States (US), and Canada have embarked on multi-billion dollar research programs to reduce the environmental impacts of aviation. These include the Clean Sky program [16], the Environmentally Responsible Aviation (ERA) program [17], and the Green Aviation Research & Development Network (GARDN) [18], respectively. In order to meet the targeted fuel efficiency goals outlined by ACI, CANSO, IATA, and ICCAIA, critical aviation infrastructure and technology advancement is required [19], beyond the capabilities of current tube-and-wing commercial aircraft.

The National Aeronautics and Space Administration (NASA) announced the $N + 3$ quest in 2008 to develop enabling technologies to provide step change improvements in fuel efficiency over current state-of-the-art commercial aircraft, targeting aircraft three generations ahead of its current fleet. Some proposed designs include, but are not limited to, blended wing bodies configurations [20, 21] with and without distributed propulsion system [22, 23], shown in Figures 1.1 and 1.2, and the double bubble D8 concept [24, 25], shown in Figure 1.3. Designing these aircraft will require a modern engineering approach, including simulation.

With the advent of digital computers, CFD became a vital instrument for engineers to understand, design and analyse not only aerospace products but also any product interacting with a fluid, including automotive, marine, heating and cooling, architecture, power generation, and pharmaceutical. Although CFD is being widely used in the design of commercial aircraft, there are still many emerging areas where CFD is expected to provide step improvements in the future. The primary role of CFD is to approximate the flow field on a computational grid by discretizing the Partial Differential Equations (PDEs) governing fluid flow, namely the inviscid Euler equations [26] or the viscous Navier-Stokes equations [27, 28]. CFD allows for comprehensive access compared to an experimental study which is limited to a finite number of locations where measuring instruments are placed. It also significantly reduces design



Figure 1.1. Boeing’s sub-scale X-48B blended wing body aircraft without distributed propulsion flew over the edge of Rogers Dry Lake at Edwards Air Force Base during its fifth flight on Aug. 14, 2007 [2].

costs as it does not require a physical model nor prototype and experimental apparatus. Furthermore, with the recent development of Micro Air Vehicles (MAVs) and more flexible airframes, wind tunnel set-ups are becoming less capable of providing the required conditions to study an object experimentally. These have turned CFD into an indispensable component in aerospace design. However, based on the NASA Vision 2030 [29], in order to study the conceptual designs mentioned earlier, we need radical improvement in the CFD prediction of unsteady flow in the vicinity of complex geometries. This is because there are always some degrees of uncertainty and error associated with CFD simulations, primarily introduced by physical approximation and discretization errors. In fact, the governing equations and physical models used in CFD are usually simplified either because the physics is not completely known or for the sake of computational efficiency. However, one of the primary sources of error is



Figure 1.2. NASA N3-X blended wing body aircraft concept with turbo-electric distributed propulsion system [3].

due to spatial discretization, which is sensitive to the number of Degrees of Freedom (DOF) used for the computation. CFD simulations require large numbers of DOF to capture the complex physics governed by the Navier-Stokes equations with a high level of accuracy, which with the current hardware architectures is inherently expensive. This challenge motivates the current work, aiming to investigate methods to make CFD more accurate and computationally efficient.

1.2 Turbulence in Fluid Mechanics

Understanding turbulence is a critical prerequisite in the process of aircraft design using CFD. Although there is no definitive definition for turbulence, we can describe it as a chaotic distribution of vorticity in time and space, with a wide range of length and time scales, ranging from large high-energy vortices to small dissipative eddies [30, 31]. Turbulent flows are characterized by three-dimensional chaotic velocity fluctuations, where a small perturbation in the initial condition can exponentially grow with time and significantly affect the flow



Figure 1.3. The double bubble D8 aircraft concept with engines flush mounted on the top rear of the fuselage, designed based on a concept known as Boundary Layer Ingestion (BLI), claimed to achieve a 36% reduction in fuel burn from its configuration [4]. This aircraft is designed to fly at Mach 0.74 and carry 180 passenger for domestic flights [5].

structure at a later time. Turbulence is a natural state of most flows, from the liquid core of our planet, oceanic and atmospheric flows on the surface of the Earth, solar flares on the sun to galaxy clusters billions of light-years away [30]. This makes turbulence an integral part of practical engineering applications. Turbulent flows consist of a wide range of length scales, where the largest turbulent length scale is proportional to the characteristic length scale [30] and the smallest is dictated by the fluid viscosity μ and the dissipation rate ε through the dissipation length scale, known as the Kolmogorov length scale $\hat{\eta}$ [32]

$$\hat{\eta} = \left(\frac{\mu^3}{\varepsilon} \right)^{\frac{1}{4}}. \quad (1.1)$$

The large turbulent length scales within a flow are inertially unstable tending to break into smaller-scale eddies. In other words, energy cascades from large scale eddies to smaller ones until the Reynolds number reaches approximately unity. At this point, viscous effects become dominant, and kinetic energy starts to dissipate. A diagram of the turbulent kinetic energy

cascade with a distribution of different scales is shown in Figure 1.4 where the kinetic energy is plotted against the wavenumber.

The relationship between the largest eddy and the smallest turbulent length scales can be expressed as a function of the Reynolds number according to [30]

$$\frac{L}{\hat{\eta}} = Re^{\frac{3}{4}}, \quad (1.2)$$

where L is the characteristic length scale. Hence the larger the Reynolds number, the greater the ratio of the largest to smallest turbulent scales, necessitating a finer resolution in space to capture the smallest eddies. In fact, the fidelity of a CFD simulation of turbulent flow is determined based on the range of length scales resolved by the simulation approach. The following subsections provide further details regarding the simulation approaches in CFD.

1.3 Simulation Approaches

Common simulation approaches include Reynolds-Averaged Navier-Stokes (RANS) methods [33], Direct Numerical Simulation (DNS), and LES [34]. As mentioned earlier, turbulent flows consist of a wide range of length scales. DNS numerically resolves the entire range of the turbulent length and time scales. The number of degrees of freedom required to resolve all eddies in a cube can be estimated as [30]

$$DOF = \left(\frac{L_c}{\Delta x} \right)^3 \approx \left(\frac{L_c}{L} \right) Re^{\frac{9}{4}}, \quad (1.3)$$

where L_c is the edge length of a cubic domain of equal sides, and Δx is the numerical grid spacing required to resolve the smallest length scales. Hence, the total number of degrees of freedom increases rapidly as Re increases. Furthermore, from Equation 1.2, a larger Reynolds number requires a finer grid resolution, which dictates a smaller time step size. All these make DNS computationally expensive. Consequently, although DNS is the most accurate approach, its application is limited by the Reynolds number, and may not be feasible for flows with moderate to high Reynolds numbers in terms of computational cost with current computing hardware.

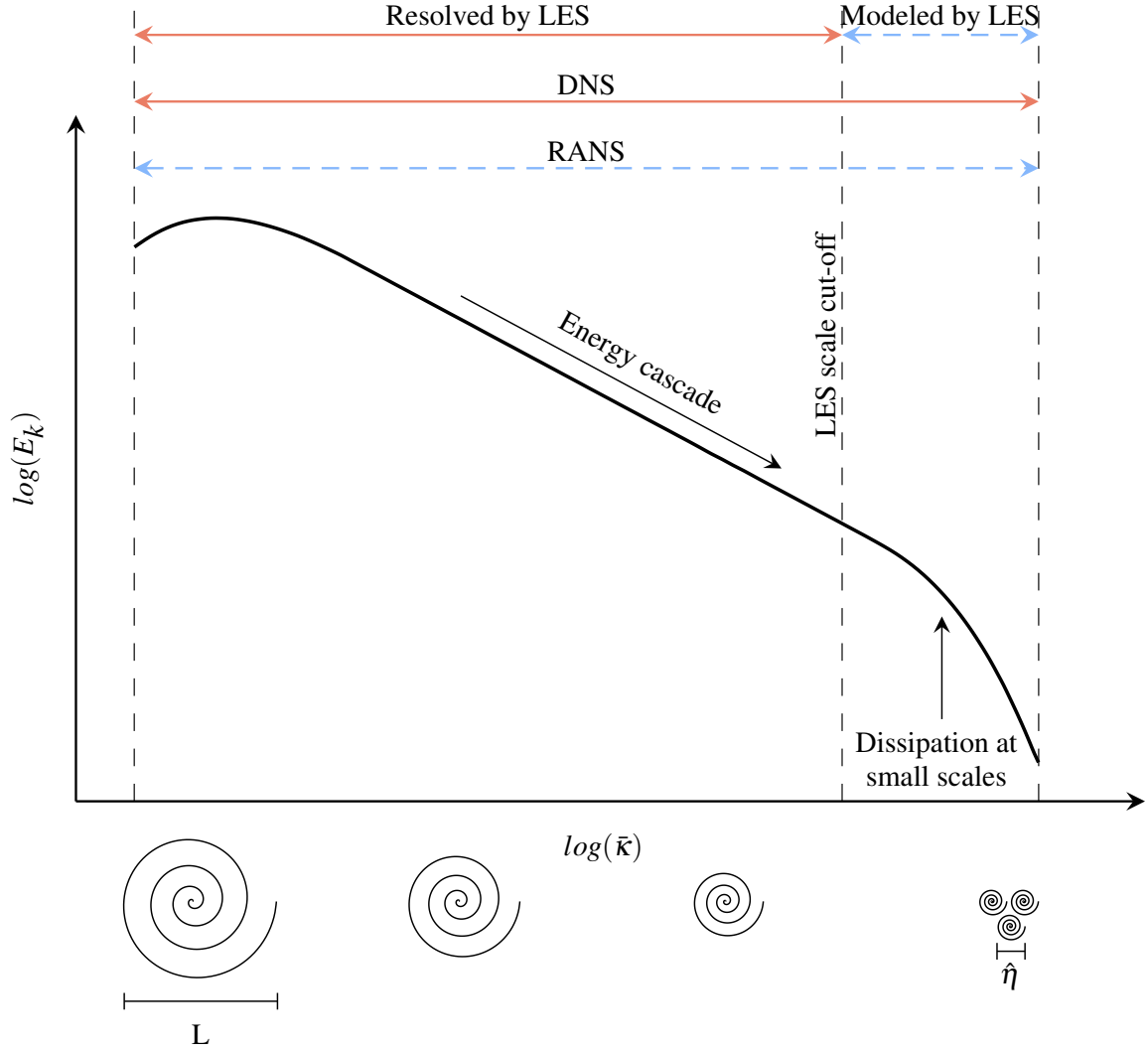


Figure 1.4. Diagram of the turbulent kinetic energy cascade. Blue lines denotes modelling and orange lines denotes simulation.

In contrast, RANS methods model the entire range of the turbulent length scale by utilizing the concept of Reynolds decomposition in which the instantaneous velocity field is decomposed into time-averaged and fluctuating components

$$v(\mathbf{x}, t) = \bar{v}(\mathbf{x}) + v'(\mathbf{x}, t), \quad (1.4)$$

where $\bar{v}(\mathbf{x})$ and $v'(\mathbf{x}, t)$ are the time-averaged and fluctuating components of the velocity field $v(\mathbf{x}, t)$ respectively. Following the Reynolds decomposition, the conservation of momentum

equations for incompressible flow can be expressed as [35]

$$\rho \frac{D(v_j)}{Dt} = \frac{\partial}{\partial x_i} \left[\mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) - \bar{p} \delta_{ij} - \overline{\rho v'_i v'_j} \right], \quad (1.5)$$

where δ_{ij} is the Kronecker delta and $\overline{\rho v'_i v'_j}$, arising from the fluctuating velocity field, represents the Reynolds stresses, which is computed by eddy-viscosity models [35]. This results in a time-averaged expression of Navier-Stokes equations. The turbulence models used by RANS methods are formulated for specific types of applications and are not accurate for all classes of flows [35]. Furthermore, they only predict the largest length scales in the flow and neglect the unsteady nature of the flow. These make RANS inaccurate when approximating unsteady separated flows and narrow their use to a relatively small region of the operation design space [30, 35].

To reduce the computational burden of DNS, but benefit from a higher numerical accuracy than RANS, high-fidelity scale resolving approaches such as LES can be used. LES resolves the larger length scales, while modelling the effect of smaller eddies. Recent research has shown that scale-resolving techniques such as LES are able to obtain accurate results in complex unsteady flow regimes where current industry-standard RANS methods often fail [36, 37]. Based on accuracy considerations, industrial deployment of LES will be critical in achieving the step-change CFD technology requirements of Clear Sky, ERA, and GARDN as outlined by NASA [29]. Figure 1.4 indicates the resolution limits of different simulation approaches. The cut-off wavenumber for LES is usually indicated by the grid size, which indicates the scale after which the small-scale information is filtered from the solution. The effects of these filtered smaller eddies are then modelled using a Subgrid-Scale (SGS) model. SGS models primarily account for the under-resolution of LES by dissipating the energy from the smallest resolved scales. This is typically done by introducing an artificial viscosity to the numerical simulation, which mimics the dissipative nature of the smaller-scale eddies, a practice known as explicit LES. As mentioned earlier, there is also another approach known as Implicit LES (ILES). In this approach the dissipation error of the numerical scheme acts as an SGS model when coupled with appropriate high-order temporal schemes [38, 39]; hence, no extra terms are included in the Navier-Stokes equations which alleviates the potentially expensive

implementation costs of SGS models and avoids complexity in linearization of the system of equations. The accuracy of the ILES approach has shown consistency with experimental and DNS studies [40, 41, 42]. Further studies on stabilization of high-order ILES have been performed by means of filtering the solution polynomials [43]. In the current work, we are particularly interested in using ILES for three-dimensional studies.

1.4 High-Order Methods

As discussed earlier, aerospace applications benefit from high-fidelity unsteady solution techniques, such as LES and DNS. The traditional Finite Volume Methods (FVM) represent a continuous PDE using a discrete number of volumes where inside each volume the solution is represented by a constant cell average. FVM are proven to be robust for non-linear problems such as simulations of detonation and shock waves, especially when coupled with the Essentially Non-Oscillatory (ENO) [44] and Weighted Essentially Non-Oscillatory (WENO) [44, 45, 46] procedures. While FVM are suitable for unstructured meshes, higher-order accuracy can only be achieved by increasing the computational stencil representing the continuous PDE, which poses significant computational cost. Furthermore, although FVM has an explicit form, it is dominated by point-wise indirect operations, which requires interactions with adjacent elements. Hence FVM are bounded by the memory bandwidth and are capable of achieving only about 3% of peak Floating Point Operations per second (FLOP/s) [47].

On the other hand, the Finite Difference Methods (FDM) approximate a PDE using a discrete number of points. Although FDM are computationally efficient and can be extended to high orders of accuracy, they are generally not conservative since they represent the solution with a divergence form. Furthermore, FDM requires curvilinear meshes. This makes them less suitable for complex geometries.

The classical Finite Element Methods (FEM) combine the appealing features of being an unstructured method and achieving high-order accuracy. The solution is represented using higher-degree polynomials on an element-wise basis. The solution is globally continuous in FEM, with the same solution value on both sides of element interfaces. This requires inversion of a global mass matrix which is computationally expensive and limits the FEM to

implicit solvers only with direct global coupling.

Recent research has led to the development of new high-order unstructured spatial discretizations by combining the element-wise high-order accuracy of the FEM with the localized solution representation of the FVM, such as the Discontinuous Galerkin (DG) [48, 49], Spectral Volume (SV) [50], Spectral Difference (SD) [51], and Staggered Grid (SG) [52]. These schemes benefit from the high-order accuracy of the FEM, as they use an element-wise high-order representation of the solution, yet allow for the solution to be discontinuous on element interfaces, reducing their computational cost by not requiring inversion of a global mass matrix. Furthermore, they are suitable for unstructured meshes and have an explicit form. It has been demonstrated previously that these methods are particularly appealing for scale-resolving LES/DNS simulations of unsteady turbulent flows, and that they are particularly well-suited for modern many-core hardware architectures [47]. Table 1.1 lists pros and cons of each spatial discretization method.

Recently an unstructured scheme was introduced by Huynh [53] which operates on the differential form of the conservation equations. This scheme is particularly appealing as it can recover different high-order methods including the DG, SD, and SV depending on the choice of a correction function. FR is capable of achieving about 55% of the peak FLOP/s [47] since almost all operations are matrix multiplies when the scheme uses higher polynomial orders. This study utilizes the FR scheme which is explained thoroughly in Chapter 3.

Table 1.1. Comparison of spatial discretization methods, adapted from Hesthaven and Warburton [11], where ✓ represents the capability of the methods, ✗ indicates that the method is incapable of solving such problems, and (✗) reflects inefficiency or limitations.

Scheme	Unstructured Meshes	High-order accuracy	Explicit Form	Conservative	Cost Efficient	p -adaptivity
FDM	✗	✓	✓	✗	✓	✗
FVM	✓	(✗)	✓	✓	✗	✗
FEM	✓	✓	✗	✓	(✗)	✓
DG/FR	✓	✓	✓	✓	✓	✓

1.5 Parallel Computing

Microprocessor technology has been developing exponentially over the past decade. Processor clock rates have increased by a factor of 200 from 40 MHz in 1988 (MIPS R3000) [54] to 8.79 GHz in 2012 (AMD FX-8350). Furthermore, processors are now capable of executing multiple tasks simultaneously. The average number of Cycles Per Instruction (CPI) has improved by an order of magnitude [54]. These result in a higher peak FLOP/s of several orders of magnitude. However, the ability of the Dynamic Random Access Memory (DRAM) to feed the Central Processing Unit (CPU) data has lagged behind, causing a performance bottleneck. In fact, DRAM access time has improved only at a quarter the rate of clock speeds in the past decade.

Concurrency plays an important role to accelerate computation. Decomposition of a computation to smaller tasks and assigning them to different processors to be executed in parallel is called parallel computing. The simultaneous execution of smaller tasks yields a significant speed-up. Furthermore, parallel computing provides larger memory capacity and higher bandwidth, resulting in better memory system performance. These have made parallel computing a vital component of engineering and scientific numerical studies including aerodynamic optimization, charge distribution optimization of combustion engines, design of high-speed circuits, structural integrity optimization, bioinformatics, and astrophysics among others [54]. In the current work, we use the Message Passing Interface (MPI) for parallel computation, which is a standard library created to unify different message passing operations into a single global library.

1.6 Research Objectives and Contributions

This study aims to investigate methods to make simulation of unsteady flows computationally efficient, specifically by introducing a novel dynamically load-balanced polynomial adaptation algorithm suitable for unsteady flows. This is performed using an in-house High-Order Unstructured Solver (HORUS) developed at the Concordia Computational Aerodynamics Laboratory, which is based on the flux reconstruction scheme. The accuracy and efficiency

of our adaptation approach are illustrated for a range of applications on fixed and moving domains. To do this, we start by introducing a novel non-dimensional vorticity-based polynomial adaptation indicator and explain how we alleviate the computational burden caused by adaptation by deploying a Dynamic Load Balancing (DLB) algorithm. We then verify the utility of this approach when applied to the arbitrary Lagrangian–Eulerian form of the compressible Navier–Stokes equations, and the DLB implementation by performing simulations of an Euler Vortex (EV). To illustrate the accuracy, speed up factor, and efficiency of the adaptation approach, we perform simulations of flow over an oscillating circular cylinder with two different flow settings, dynamic stall of a two-dimensional NACA 0012 airfoil undergoing heaving and pitching motions, shallow dynamic stall of a three-dimensional SD 7003 airfoil undergoing heaving and pitching motions, flow over a Vertical Axis Wind Turbine (VAWT) composed of two NACA 0012 airfoils, turbulent flow over a three-dimensional circular cylinder, and turbulent flow over a three-dimensional SD 7003 airfoil at a fixed angle of attack.

The primary contribution of this work is the introduction, verification, and validation of a novel dynamically load-balanced polynomial adaptation algorithm suitable for unsteady turbulent flows using a high-order unstructured spatial discretization, which represents significant speed up for high-order accurate LES of turbulent flows using the FR scheme. We expect this will reduce the daunting simulation cost of LES and allow us to apply CFD to a greater range of practical studies, specifically optimization and redesign of conventional aircraft to reduce the carbon footprint of aviation.

1.7 Thesis Outline

Chapter 1 presents an introduction to this work and motivations. It then explains fundamentals, including turbulence, simulation approaches, and parallel computing. Furthermore, it outlines the objectives and contributions of the current work.

Chapter 2 presents the governing equations for fluid flow, including conservation of mass, momentum, and energy. It then derives the Euler and Navier-Stokes equations for static as well as moving and deforming domains.

Chapter 3 introduces the FR scheme as the primary spatial discretization scheme used in this study, with a thorough numerical derivation for one-dimension along with the multi-dimensional and Arbitrary Lagrangian–Eulerian (ALE) extensions. It also discusses the advantages of the FR scheme compared to classical discretizations.

Chapter 4 introduces different adaptation strategies and discusses their pros and cons. It then introduces a novel polynomial adaptation algorithm. Furthermore, it outlines the load balancing technique employed in this study to improve the efficiency of adaptive simulations.

Chapter 5 provides a set of verification and validation test cases for the FR scheme and the in-house solver.

Chapter 6 explores the utility of the novel non-dimensional vorticity-based indicator for p -adaptation for a range of applications on moving and deforming domains.

Chapter 7 explores the performance of the p -adaptation algorithm coupled with dynamic load balancing for turbulent flows over three-dimensional bodies.

Chapter 8 finalizes this work by summarizing the conclusions and presenting recommendations for future work.

Chapter 2

Governing Equations

In this chapter, we present the conservation of mass, momentum and energy in the differential form to suit the FR scheme, presented in Section 3.1. We then conclude by deriving the Euler and Navier-Stokes equations, which are the governing equations in this study, from the conservation equations.

2.1 Conservation of Mass

Conservation of mass states that the mass of a closed system is constant. Using a control volume approach for a finite control volume $\Omega(t)$ bounded by a control surface $S(t)$, we can derive a general form of conservation law using the Reynolds transport theorem as [55]

$$\frac{dN_{system}}{dt} = \frac{\partial}{\partial t} \int_{\Omega(t)} \rho \eta d\Omega + \int_{S(t)} \rho \eta \mathbf{v} \cdot \hat{\mathbf{n}} dS, \quad (2.1)$$

where N is the property to be conserved within the system, η is the amount of the conserved property per unit mass, ρ is density, \mathbf{v} is the velocity vector, $\hat{\mathbf{n}}$ is the outward pointing normal vector on the control surface, and t is time. The first term on the right-hand side of Equation 2.1 is the rate at which the conserved property is generated or used in the control volume, while the second term represents the rate of transport of property in or out of the control volume. Equation 2.1 can be expressed in terms of volume integral using the divergence

theorem

$$\frac{dN_{system}}{dt} = \int_{\Omega(t)} \left[\frac{\partial \rho \eta}{\partial t} + \nabla \cdot (\rho \eta \mathbf{v}) \right] d\Omega. \quad (2.2)$$

Conservation of mass requires the rate of change of the system mass to be zero

$$\frac{dm}{dt} = 0. \quad (2.3)$$

We do not consider any source term in the current work, hence there must be no transfer of conserved variable through boundaries of the domain as it moves through the flow field. Combining Equations 2.2 and 2.3 with the conserved property to be the system mass $N = m$ and $\eta = 1$

$$\frac{dm}{dt} = \int_{\Omega(t)} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) \right] d\Omega = 0. \quad (2.4)$$

Considering the Equation 2.4, the integrand must be zero since the integral holds true for any domain size. Hence, conservation of mass can be expressed as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \quad (2.5)$$

2.2 Conservation of Momentum

Newton's second law of motion for a system states that the sum of forces acting on a system is equal to the time rate of change of momentum of that system as

$$\sum \mathbf{F}_\Omega = m\mathbf{a} = \frac{d}{dt}(m\mathbf{v}), \quad (2.6)$$

where \mathbf{F}_Ω is the sum of forces acting on the control volume, and \mathbf{a} is the acceleration vector. We can apply the Reynolds transport theorem to calculate the total momentum within the system, with the conserved property to be the system momentum $N = m\mathbf{v}$, and the amount of the conserved property per unit mass to be $\eta = \mathbf{v}$

$$\frac{d}{dt}m\mathbf{v} = \int_{\Omega(t)} \left[\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v}) \right] d\Omega. \quad (2.7)$$

The sum of the forces acting on the system can be expressed as a composition of external or body forces acting on the control volume and internal or surface forces acting on the control surface

$$\sum \mathbf{F}_\Omega = \int_{\Omega(t)} \mathbf{F}_b d\Omega + \int_{S(t)} \boldsymbol{\sigma} \cdot \hat{\mathbf{n}} dS, \quad (2.8)$$

where \mathbf{F}_b is the vector of body forces and $\boldsymbol{\sigma}$ is the stress tensor defined as

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix}, \quad (2.9)$$

where σ_{ii} is the normal stress acting on a plane perpendicular to the i -axis, and τ_{ij} is the shear or viscous stress acting on a plane perpendicular to the i -axis in direction of the j -axis. We can express Equation 2.8 in terms of volume integral using the divergence theorem as

$$\sum \mathbf{F}_\Omega = \int_{\Omega(t)} \mathbf{F}_b d\Omega + \int_{\Omega(t)} \nabla \cdot \boldsymbol{\sigma} d\Omega. \quad (2.10)$$

In Cartesian coordinates for a Newtonian fluid, normal and shear stress terms are defined as

$$\begin{aligned} \sigma_{ii} &= -p - \frac{2}{3}\mu(\nabla \cdot \mathbf{v}) + 2\mu \frac{\partial v_i}{\partial x_i} \\ \tau_{ij} &= \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right), \end{aligned} \quad (2.11)$$

where p is the pressure and μ is the dynamic viscosity. The stress tensor can be broken into pressure and viscous terms [27, 28] in a compact form as

$$\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\tau}, \quad (2.12)$$

where \mathbf{I} is the identity matrix. Viscous terms $\boldsymbol{\tau}$ can be expressed as a summation of a full matrix and a diagonal matrix as

$$\boldsymbol{\tau} = \bar{\lambda}(\nabla \cdot \mathbf{v})\mathbf{I} + 2\mu\mathbf{T}, \quad (2.13)$$

where $\bar{\lambda} = -2\mu/3$, and T_{ij} is defined as

$$T_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right). \quad (2.14)$$

Considering Equations 2.6, 2.7, 2.10, and 2.12, we can express conservation of momentum as

$$\int_{\Omega(t)} \left[\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I} - \boldsymbol{\tau}) \right] d\Omega = \int_{\Omega(t)} \mathbf{F}_b d\Omega. \quad (2.15)$$

In the current work there are no external forces acting on the body, therefore, $\mathbf{F}_b = 0$. Furthermore, since the integral holds true for any domain size with smooth solution, the conservation of momentum can be written in differential form as

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I} - \boldsymbol{\tau}) = 0. \quad (2.16)$$

2.3 Conservation of Energy

The first law of thermodynamics for a system states that the time rate of change of a closed system's energy is equal to the net rate of heat transferred into the system plus the net rate of work done on the system as

$$\frac{dE}{dt} = \dot{Q} + \dot{W}, \quad (2.17)$$

where E is the total energy of the system, \dot{Q} is the net rate of heat added to the system, and \dot{W} is the net rate of work done on the system. \dot{Q} can be computed across the control surface as

$$\dot{Q} = \int_{S(t)} -\mathbf{q} \cdot \hat{\mathbf{n}} dS, \quad (2.18)$$

where \mathbf{q} is the surface heat flux per unit area, which can be written as $\mathbf{q} = -k \nabla T$ using Fourier's Law of Heat Conduction, where k is the thermal conduction coefficient and T is the temperature. Other sources of heat such as radiation, or heat released by chemical reactions are not considered in the current work. Furthermore, \dot{W} is net rate of work done on the system by body forces as well as normal and shear stresses acting on a fluid particle, where the former is equal to the dot product of body forces and the particle's velocity \mathbf{v} integrated

on the control volume, and the latter is equal to the dot product of stress tensor $\boldsymbol{\sigma}$ and the particle's velocity integrated across the control surface

$$\dot{W} = \int_{\Omega(t)} \mathbf{F}_b \mathbf{v} d\Omega + \int_{S(t)} \boldsymbol{\sigma} \mathbf{v} \cdot \hat{\mathbf{n}} dS. \quad (2.19)$$

Hence, by applying the divergence theorem, and using Equations 2.18 and 2.19, Equation 2.17 can be rewritten in terms of a volume integral as

$$\frac{dE}{dt} = \int_{\Omega(t)} \nabla \cdot (\boldsymbol{\sigma} \mathbf{v}) d\Omega + \int_{\Omega(t)} \mathbf{F}_b \mathbf{v} d\Omega + \int_{\Omega(t)} \nabla \cdot (k \nabla T) d\Omega. \quad (2.20)$$

We apply the Reynolds transport theorem with the conserved property to be the system's total energy $N = E$, and the intensive property to be the total energy per unit mass $\eta = e$

$$\frac{dE}{dt} = \int_{\Omega(t)} \left[\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \mathbf{v}) \right] d\Omega. \quad (2.21)$$

From Equation 2.20 and 2.21, we can express the general form of the conservation of energy as

$$\int_{\Omega(t)} \left[\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \mathbf{v}) \right] d\Omega = \int_{\Omega(t)} \nabla \cdot (\boldsymbol{\sigma} \mathbf{v} + k \nabla T) d\Omega + \int_{\Omega(t)} \mathbf{F}_b \mathbf{v} d\Omega. \quad (2.22)$$

Breaking the stress tensor into pressure and viscous terms using Equation 2.12, we can express conservation of energy as

$$\int_{\Omega(t)} \left[\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \mathbf{v} + p \mathbf{v} - \boldsymbol{\tau} \mathbf{v} - k \nabla T) \right] d\Omega = \int_{\Omega(t)} \mathbf{F}_b \mathbf{v} d\Omega. \quad (2.23)$$

Body forces are included in equations for generality; however, as mentioned in the previous section, external forces are not considered in the current work, so $\mathbf{F}_b = 0$. Since the integral holds true for any domain size, the final form of conservation of energy can be written in differential form as

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \mathbf{v} + p \mathbf{v} - \boldsymbol{\tau} \mathbf{v} - k \nabla T) = 0. \quad (2.24)$$

2.4 General Conservation Law

Considering the conservation Equations 2.5, 2.16, and 2.24 discussed in previous sections, we can derive a general conservation law in differential form as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F} = 0, \quad (2.25)$$

where \mathbf{u} is the vector of conserved variables taken as ρ , $\rho \mathbf{v}$, and ρe , and \mathbf{F} is a tensor of fluxes of the conserved variables taken as $\rho \mathbf{v}$, $(\rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I} - \boldsymbol{\tau})$, and $(\rho e \mathbf{v} + p \mathbf{v} - \boldsymbol{\tau} \mathbf{v} - k \nabla T)$ for conservation of mass, momentum, and energy respectively.

2.5 Euler and Navier-Stokes Equations

The Navier-Stokes equations [27, 28] are a set of coupled equations that describe the relation between density, velocity, temperature, and pressure of a moving fluid. They can be obtained by rewriting conservation of mass, momentum, and energy in a compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{F}_e(\mathbf{u}) - \mathbf{F}_v(\mathbf{u}, \nabla \mathbf{u})) = 0 \quad (2.26)$$

where \mathbf{u} is a vector of conserved variables, $\mathbf{F}_e(\mathbf{u})$ is the inviscid flux, and $\mathbf{F}_v(\mathbf{u}, \nabla \mathbf{u})$ is the viscous flux, defined as

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ \rho e \end{bmatrix}, \quad (2.27)$$

$$\mathbf{F}_{e,j}(\mathbf{u}) = \begin{bmatrix} \rho v_j \\ \rho v_j v_x + \delta_{jx} p \\ \rho v_j v_y + \delta_{jy} p \\ \rho v_j v_z + \delta_{jz} p \\ (\rho e + p) v_j \end{bmatrix}, \quad (2.28)$$

$$\mathbf{F}_{v,j}(\mathbf{u}, \nabla \mathbf{u}) = \begin{bmatrix} 0 \\ \tau_{jx} \\ \tau_{jy} \\ \tau_{jz} \\ v_x \tau_{jx} + v_y \tau_{jy} + v_z \tau_{jz} + k \frac{\partial T}{\partial x_j} \end{bmatrix}, \quad (2.29)$$

where $j = [x, y, z]$, and δ_{ij} is the Kronecker delta, and the thermal conduction coefficient is determined as

$$k = \frac{\mu c_p}{Pr}, \quad (2.30)$$

where Pr is the Prandtl number taken as 0.71 for air in the current work. We notice that we only have five equations for six unknowns. An equation of state for a perfect gas, which relates the pressure, density and temperature of the gas, is required to complete the set of equations

$$p = \rho r T, \quad (2.31)$$

where r is the specific gas constant, which is related to the specific heat coefficient at constant volume c_v , and the specific heat coefficient at constant pressure c_p as

$$r = c_p - c_v. \quad (2.32)$$

The total energy of a system E is composed of internal energy and kinetic energy, which per unit mass can be expressed as

$$e = e_i + e_k = e_i + \frac{\mathbf{v} \cdot \mathbf{v}}{2}, \quad (2.33)$$

where e_i is the internal energy per unit mass, which is related to the gas temperature as

$$e_i = c_v T. \quad (2.34)$$

Considering Equations 2.32, 2.33, and 2.34, we can rewrite the equation of state for a perfect gas 2.31 as

$$p = \rho(\gamma - 1) \left(e - \frac{\mathbf{v} \cdot \mathbf{v}}{2} \right), \quad (2.35)$$

where $\gamma = c_p/c_v$ is the ratio of specific heat coefficients which is taken as 1.4 for air in the current work. The surface heat flux \mathbf{q} used in the viscous flux $\mathbf{F}_v(\mathbf{u}, \nabla \mathbf{u})$ can be expressed in terms of pressure, density, and the total energy by using Equations 2.33, 2.34, and 2.35 as

$$\mathbf{q} = -\frac{\mu c_p}{Pr} \nabla T = -\frac{\mu}{Pr} \nabla \left(e + \frac{p}{\rho} - \frac{\mathbf{v} \cdot \mathbf{v}}{2} \right). \quad (2.36)$$

Equation 2.26 reduces to the Euler equations by neglecting the effect of viscosity and thermal conduction. The Euler equations describe the relation between density, velocity, and pressure of a moving fluid [26] as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}_e(\mathbf{u}) = 0, \quad (2.37)$$

where \mathbf{u} is a vector of conserved variables and $\mathbf{F}_e(\mathbf{u})$ is the inviscid flux taken from Equations 2.27 and 2.28 respectively.

2.6 Arbitrary Lagrangian-Eulerian Formulation

The ALE form of the Navier–Stokes equations can be derived for a moving domain $\Omega(t)$ by adding ALE fluxes and the local mesh velocity to Equation 2.26 as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{F}_e(\mathbf{u}) - \mathbf{F}_v(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{F}_g(\mathbf{u}, \mathbf{v}_g)) + \mathbf{u}(\nabla \cdot \mathbf{v}_g) = 0, \quad (2.38)$$

where \mathbf{u} is a vector of conserved variables defined in Equation 2.27, \mathbf{v}_g is the vector of grid velocity defined as $\mathbf{v}_g = d\mathbf{x}_g/dt$, \mathbf{x}_g is coordinates of a mapping point used to define the element, $\mathbf{F}_e(\mathbf{u})$ is the inviscid fluxes defined in Equation 2.28, $\mathbf{F}_v(\mathbf{u}, \nabla \mathbf{u})$ is the viscous flux

defined in Equation 2.29, and $\mathbf{F}_g(\mathbf{u}, \mathbf{v}_g)$ is the ALE fluxes defined as

$$\mathbf{F}_g(\mathbf{u}, \mathbf{v}_g) = -\mathbf{v}_g \mathbf{u} = \begin{bmatrix} -\rho \mathbf{v}_g \\ -\rho v_x \mathbf{v}_g \\ -\rho v_y \mathbf{v}_g \\ -\rho v_z \mathbf{v}_g \\ -\rho e \mathbf{v}_g \end{bmatrix}, \quad (2.39)$$

where \mathbf{v} is the vector of fluid velocity. The additional source term $\mathbf{u}(\nabla \cdot \mathbf{v}_g)$ is added to Equation 2.38 whenever the divergence of the velocity field is not zero, which is generally the case for mesh deformation other than solid body translation, and accounts for local volume change within an element [56].

2.7 Advection Equation

Linear advection describes the motion of a scalar quantity carried by a flow at constant velocity. It can be defined by the general form of conservation of mass as

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F} = 0, \quad (2.40)$$

where $u = u(\mathbf{x}, t)$ is the scalar variable and $\mathbf{F} = \mathbf{F}(u) = \mathbf{A}u$ is its flux. We can also express the advection equation using the gradient of the scalar quantity as

$$\frac{\partial u}{\partial t} + \mathbf{A} \nabla u = 0, \quad (2.41)$$

where the Jacobian matrix $\mathbf{A} = \partial \mathbf{F} / \partial u$ is the advection velocity.

2.8 Diffusion Equation

Linear diffusion describes the changes in concentration of a scalar variable towards equilibrium with zero velocity. It can be defined by the general form of conservation law as

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F} = 0, \quad (2.42)$$

where $u = u(\mathbf{x}, t)$ is the scalar variable and $\mathbf{F} = \mathbf{F}(u, \nabla u) = -D_f \nabla u$ is its flux, and D_f is the diffusion coefficient. We can also express the diffusion equation using the second gradient of the scalar quantity as

$$\frac{\partial u}{\partial t} - D_f \nabla \cdot (\nabla u) = 0, \quad (2.43)$$

The following chapter presents the numerical discretization used in this study. We start by introducing the flux reconstruction scheme, followed by its arbitrary Lagrangian-Eulerian extension. We then conclude the chapter by presenting temporal schemes used in this study.

Chapter 3

Numerical Schemes

In this chapter, we present the FR scheme for linear advection and diffusion, and extend it to multidimensional and ALE forms. We then discuss the temporal schemes used to advanced the fully-discrete scheme in time.

3.1 The Flux Reconstruction Scheme

This study is based on the FR scheme, originally introduced by Huynh [53] in 2007 as a new approach for high-order accuracy for the numerical solution of conservation laws when solving them in differential form. This scheme can recover other high-order methods including the DG, SD, and SV schemes. In this section, we describe the original one-dimensional formulation of the FR scheme for the linear advection and diffusion equations.

3.1.1 Linear Advection

Consider a general one-dimensional conservation law of the form

$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} = 0, \quad (3.1)$$

where $u = u(x, t)$ is the solution with a given initial distribution $u(x, 0) = u_0$, the flux depends on u as $F = F(u)$. We start by partitioning the computational domain into N_E non-overlapping elements E_j , where $j = 1, \dots, N_E$, with left and right element boundaries of

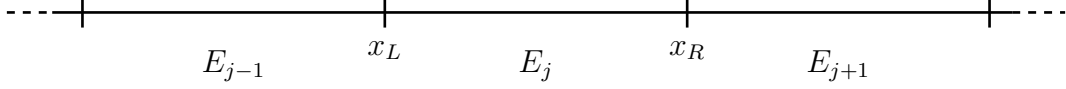


Figure 3.1. Spatial discretization of a one-dimensional domain.

x_L and x_R , as shown in Figure 3.1. On each element, the solution is then approximated at $K + 1$ points $x_{j,k}$, where $k = 1, \dots, K + 1$ and $u_{j,k}$ is the approximated solution at point k of element j . These points are known as solution points $x_{j,k}$. There is no limitation in specifying their locations, but they are typically the Gauss or Lobatto points. However, for the sake of efficiency, it is wise to define the first and last points at the element boundaries x_L and x_R . In this case, no extrapolation is needed to compute values at the boundaries of elements.

Following the flux reconstruction approach, the solution is represented by a discrete approximation on each element such that [53, 57]

$$u(x, t) \approx u^h(x, t) = \bigoplus_{j=1}^{N_E} u_j(x, t), \quad (3.2)$$

where $u^h(x, t)$ is the global piecewise approximation of the solution and $u_j(x, t)$ is a continuous representation of the solution on one of N_E elements in the domain. We take the approximate solution on each element to be a polynomial nodal basis representation of degree K polynomial such that

$$u_j(x, t) = \sum_{k=1}^{K+1} u_{j,k}(t) \phi_{j,k}(x), \quad (3.3)$$

where $u_{j,k}(t)$ is the value of the solution at one of $K + 1$ solution nodal basis points on a given element, as shown by the red points in Figure 3.6, and $\phi_{j,k}(x)$ is its corresponding nodal basis function of degree K . Figure 3.2 shows solution nodal basis points and solution polynomials for a one-dimensional case with degree $K = 2$. This approach ensures the solution is continuous on each element but allows the solution to be discontinuous on the interfaces between elements [53], meaning that the value of the solution on the left boundary of element E_j can differ from the value of the solution on the right boundary of element E_{j-1} . The Lagrange polynomials are commonly used as the nodal basis functions, due to their convenient property of taking on the value of one at the solution point $x_{j,k}$, and value

of zero at other solution points. A Lagrange basis function is generated according to

$$\phi_{j,k}(x) = \prod_{l=1, l \neq k}^{K+1} \frac{x - x_{j,l}}{x_{j,k} - x_{j,l}}. \quad (3.4)$$

Since the solution polynomial is approximated using $K + 1$ solution points, the degree of the solution polynomial will be K . Following the same approach, the flux is also represented by a discrete approximation on each element as

$$f(x, t) \approx f^h(x, t) = \bigoplus_{j=1}^{N_E} f_j(x, t), \quad (3.5)$$

where $f^h(x, t)$ is the global piecewise approximation of the flux and $f_j(x, t)$ is a continuous representation of the flux on one of N_E elements in the domain. The approximate flux of the degree K polynomial on each element is constructed using the same Lagrange basis function as

$$f_j(x, t) = \sum_{k=1}^{K+1} f_{j,k}(t) \phi_{j,k}(x), \quad (3.6)$$

where $f_{j,k}(t)$ is the value of the flux evaluated at one of $K + 1$ solution nodal basis points on a given element, as shown by the blue points in Figure 3.6. It is often more convenient to deal with a reference domain with an interval of $L_R = [-1, 1]$ rather than a physical domain, hence, as a standard practice in finite element methods, the physical element E_j is mapped into a reference space with ξ varying on L_R and x on E_j . The transformation is done by a point-to-point mapping function as

$$\begin{aligned} x(\xi) &= \frac{1}{2} (x_L + x_R + h_j \xi), \\ \xi(x) &= \frac{2}{h_j} \left(x - \frac{x_L + x_R}{2} \right), \end{aligned} \quad (3.7)$$

where h_j is the element width in the physical domain. The mapping metric is then computed

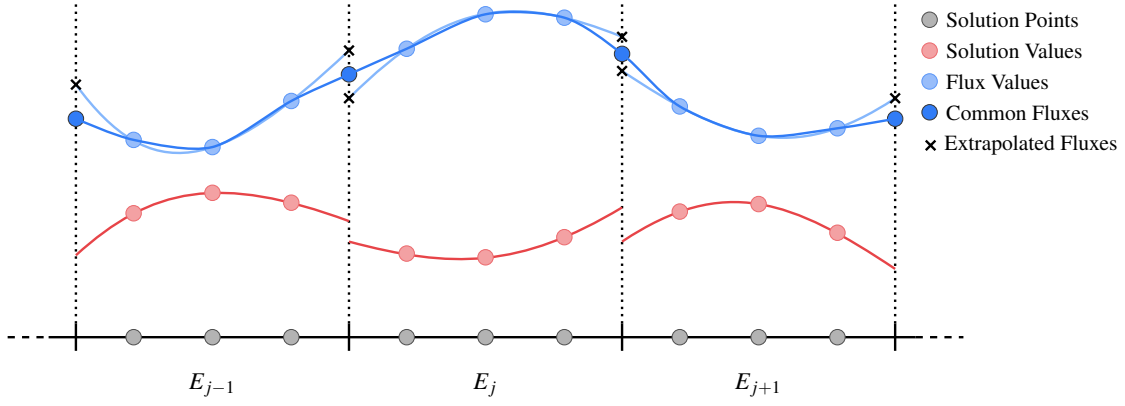


Figure 3.2. Schematic representation of solution polynomials along with the discontinuous and continuous flux functions.

as

$$\begin{aligned}\frac{\partial x}{\partial \xi} &= \frac{h_j}{2}, \\ \frac{\partial \xi}{\partial x} &= \frac{2}{h_j}.\end{aligned}\tag{3.8}$$

Mapped into the reference space, we can define the solution and flux functions as

$$u_j(\xi, t) = \sum_{k=1}^{K+1} u_{j,k}(t) \phi_k(\xi),\tag{3.9}$$

$$f_j(\xi, t) = \sum_{k=1}^{K+1} f_{j,k}(t) \phi_k(\xi),\tag{3.10}$$

where

$$\phi_k(\xi) = \prod_{l=1, l \neq k}^{K+1} \frac{\xi - \xi_l}{\xi_k - \xi_l}.\tag{3.11}$$

As an advantage of mapping to the reference space, the Lagrange basis functions are defined only once when computing a solution of uniform polynomial degree. Figure 3.3 shows the Lagrange polynomials for different polynomial degrees generated using Gauss points.

In order to solve the conservation law, we need to calculate the derivative of the flux. Note that although the flux function has the same polynomial degree as the solution, its derivative will be one order lower and lives in the polynomial space of $K - 1$. Furthermore, as shown in

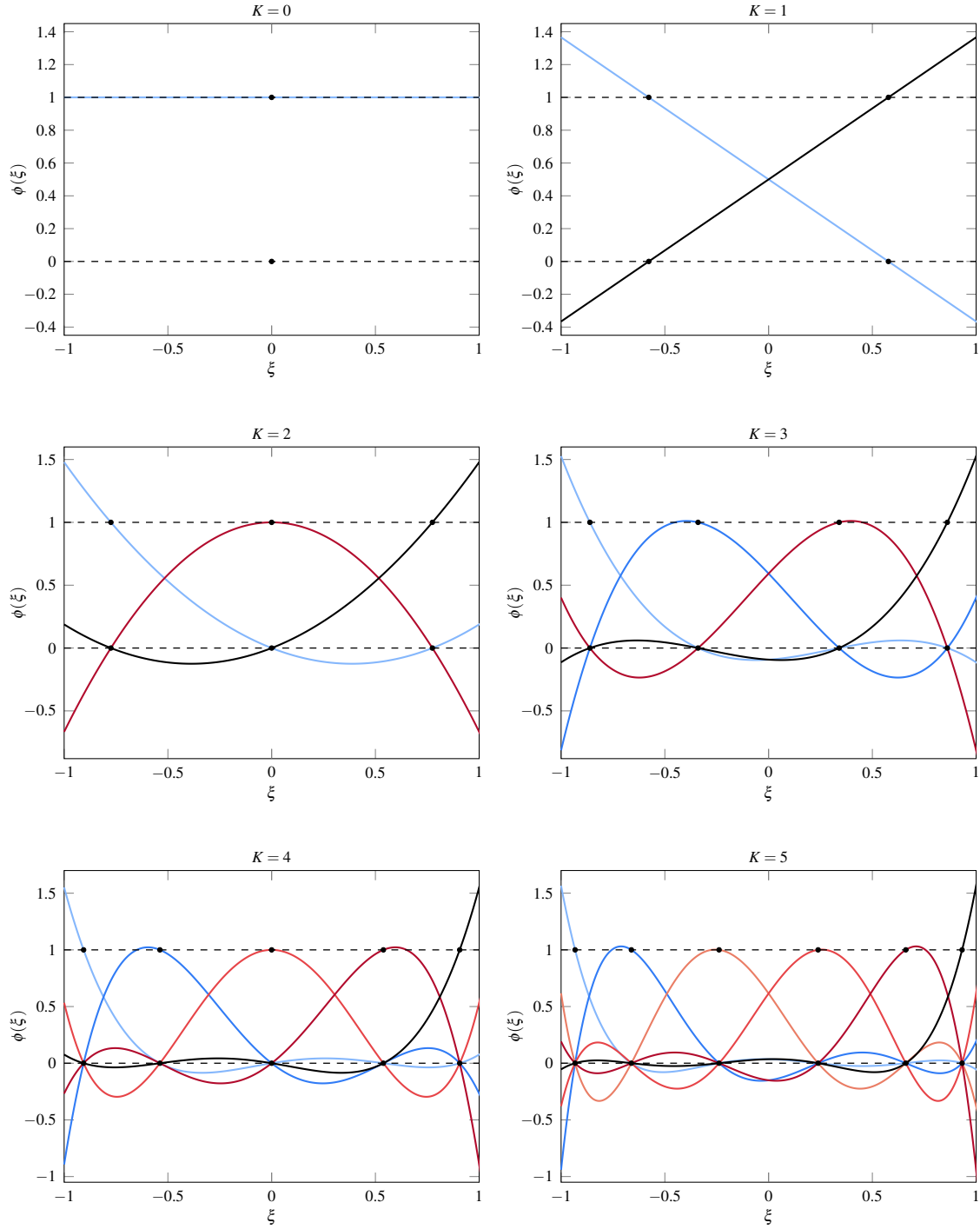


Figure 3.3. One-dimensional nodal basis functions with different polynomial orders using Gauss points (black points).

Figure 3.2, the flux polynomials form a continuous function within each element, yet they can have different values on the element interfaces making a discontinuous global piecewise approximation of the flux. The discontinuity of the piecewise flux leads to erroneous results when calculating its derivative. This is due to the fact that the discontinuous flux is not capable of evaluating the flux when applied to the conservation law, as it does not include the interactions between neighbouring elements. To circumvent this, Huynh [53] proposed a continuous flux function which approximate the flux function on each element $f_j(\xi, t)$, yet forcing continuity on element interfaces. The continuous flux function is constructed by adding a correction term to the flux function $f_j(\xi, t)$ as

$$\hat{f}_j(\xi, t) = f_j(\xi, t) + [f_L^c - f_L] g_L(\xi) + [f_R^c - f_R] g_R(\xi), \quad (3.12)$$

where $\hat{f}_j(\xi, t)$ is the corrected continuous flux function, f_L^c , f_R^c are the common Riemann fluxes on the left and right boundaries of element, f_L , f_R are the values of the flux on the left and right boundaries, and $g_L(\xi)$, $g_R(\xi)$ are the correction functions for the left and right boundaries. Figure 3.2 shows the discontinuous and corrected continuous flux functions for a one-dimensional case with degree $K = 2$. Common Riemann fluxes, shown by the green points in Figure 3.6, are calculated by applying a Riemann solver such as upwind or central schemes. The choice of Riemann solver dictates the dominant error of the numerical method. For instance, an upwind Riemann solver introduces predominantly numerical dissipation, while a central Riemann solver introduces numerical dispersion error. In this study, we used a Rusanov Riemann solver to compute the common inviscid interface fluxes. Note that $[f_L^c - f_L]$ and $[f_R^c - f_R]$ are simply calculating the jumps on the left and right element boundaries.

To construct the correction term, we also need to define the correction functions with two important characteristics. First, in order for the corrected global flux function to be continuous on the element interfaces, they need to take on value of one at their respective

interface and value of zero on the other interface, such that

$$\begin{aligned} g_L(-1) &= 1, & g_L(1) &= 0, \\ g_R(-1) &= 0, & g_R(1) &= 1. \end{aligned} \tag{3.13}$$

Furthermore, they must be defined in a way to correct the jumps on the left and right interfaces while minimizing changes to the values of the flux within an element; hence, they must approximate zero at the solution points within the element. Second, they must belong to the polynomial space of $K+1$ so that the derivative of the continuous flux function would have the same polynomial degree K as the solution function. The distribution of the flux within an element depends on the choice of the correction function. As mentioned earlier, FR unifies several high-order schemes including the DG, SD, and SV schemes. This is done through the choice of the correction function. For example, the correction functions constructed based on the Radau polynomials reduce the scheme to the DG method. Huynh [53] shows that Radau polynomials provide the most accurate solution when used to construct the correction functions. Hence, in this study, the right Radau polynomials R_R are used to define the left correction function $g_L(\xi)$, and left Radau polynomials R_L are used to define the right correction function $g_R(\xi)$. They can be defined as a function of Legendre polynomials $P_K(\xi)$ as

$$\begin{aligned} g_L(\xi) &= R_{R,K+1}(\xi) = \frac{(-1)^{K+1}}{2} (P_{K+1}(\xi) - P_K(\xi)), \\ g_R(\xi) &= R_{L,K+1}(\xi) = \frac{(-1)^{K+1}}{2} (P_{K+1}(-\xi) - P_K(-\xi)), \end{aligned} \tag{3.14}$$

where the set of Legendre polynomials used in this study to construct the Radau polynomials are shown in Table 3.1. The right Radau polynomials used in this study are shown in Figure 3.4.

Now that we have the continuous flux function, we can compute its derivative as

$$\frac{\partial \hat{f}_j(\xi, t)}{\partial \xi} = \frac{\partial f_j(\xi, t)}{\partial \xi} + [f_L^c - f_L] \frac{\partial g_L(\xi)}{\partial \xi} + [f_R^c - f_R] \frac{\partial g_R(\xi)}{\partial \xi}, \tag{3.15}$$

Table 3.1. Legendre polynomials used in the current work.

Degree K	$P_K(\xi)$
0	1
1	ξ
2	$\frac{1}{2}(3\xi^2 - 1)$
3	$\frac{1}{2}(5\xi^3 - 3\xi)$
4	$\frac{1}{8}(35\xi^4 - 30\xi^2 + 3)$
5	$\frac{1}{8}(63\xi^5 - 70\xi^3 + 15\xi)$
6	$\frac{1}{16}(231\xi^6 - 315\xi^4 + 105\xi^2 - 5)$

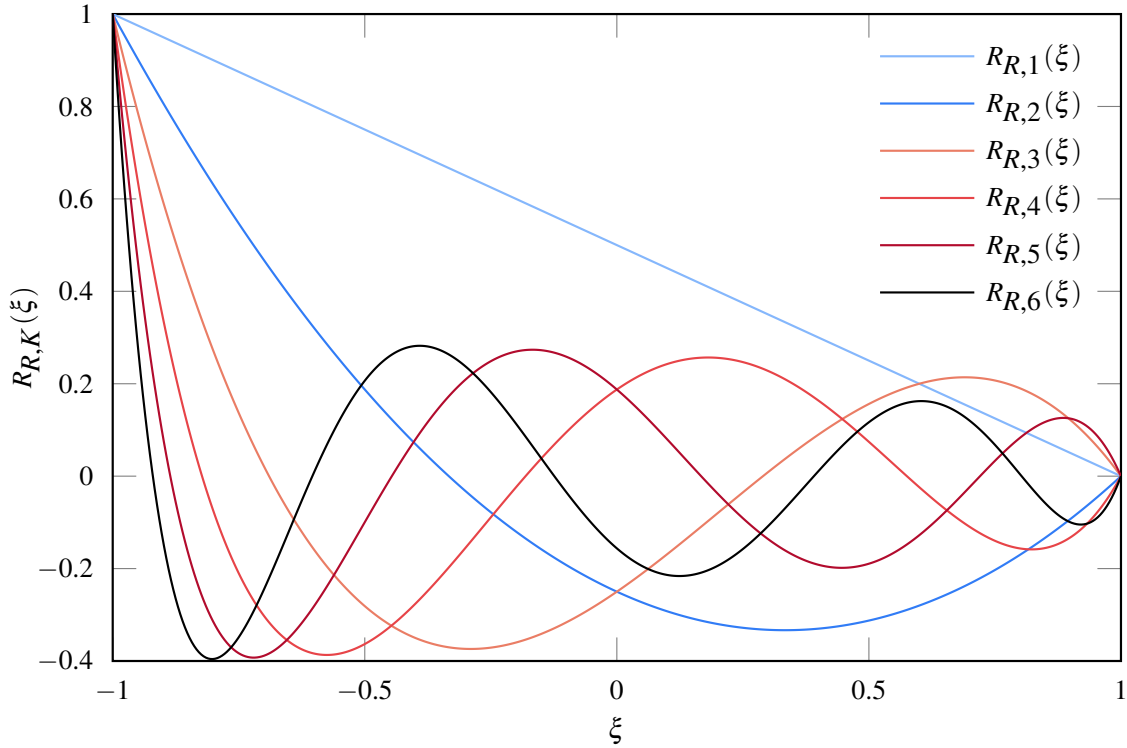


Figure 3.4. Right Radau polynomials necessary for the $K = 0$ to $K = 5$ numerical schemes.

where

$$\frac{\partial f_j(\xi, t)}{\partial \xi} = \mathcal{D} f_j(\xi, t) \quad (3.16)$$

where \mathcal{D} is the derivative matrix of size $K + 1 \times K + 1$ computed for the reference element as

$$\mathcal{D} = \begin{bmatrix} \frac{\partial \phi_1(\xi_1)}{\partial \xi} & \cdots & \frac{\partial \phi_{K+1}(\xi_1)}{\partial \xi} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi_1(\xi_{K+1})}{\partial \xi} & \cdots & \frac{\partial \phi_{K+1}(\xi_{K+1})}{\partial \xi} \end{bmatrix}.$$

Finally, the conservation equation at each solution point in physical space becomes

$$\frac{\partial u_j(x, t)}{\partial t} + \frac{\partial \hat{f}_j(\xi, t)}{\partial \xi} \frac{\partial \xi}{\partial x} = 0, \quad (3.17)$$

where $\partial \xi / \partial x$ is the mapping metric from the reference to physical space defined in Equation 3.8. The solution $u_j(x, t)$ can then be updated in time using a suitable temporal scheme.

To validate the FR scheme, one-dimensional linear advection of a sine wave through a periodic domain with a wave speed of one is considered. The domain is defined on the interval $x \in [0, 1]$, with initial distribution $u(x, 0) = \sin(2\pi x)$. Time integration is carried out with the classical $RK_{4,4}$, a fourth-order four-stage explicit Runge-Kutta scheme, with the time step size $\Delta t = 1 \times 10^{-4}$, and the total advection time of $t = 1$. Radau polynomials are used to construct the correction functions. The common fluxes on the element intervals are calculated with an upwind Riemann solver. The exact solution of the function is $u(x, t) = \sin(2\pi(x - t))$, which will be identical to the initial solution when $t = 1$. Four levels of mesh refinement with 4, 8, 16, and 32 elements, with five polynomial degrees of $K = 1$ to $K = 5$ are considered. Note that $K + 1$ solution points are required to generate a polynomial of degree K . Figure 3.5 shows the exact and numerical solutions of the four-element mesh for different polynomial degrees. It is evident that the higher polynomial degrees are capable of approximating the solution more accurately. Additionally, we notice the dissipation error, inherent in the upwind Riemann solver, for lower-order schemes. Table 3.2 shows the L_2 error norm for each

polynomial degree and mesh, where the L_2 error norm is defined as

$$\| (u_e - u^h) \|_2 = \sqrt{\frac{\sum_{k=1}^{K+1} (u_{e,k} - u_k^h)^2}{(K+1)N_E}}, \quad (3.18)$$

where u_e is the exact solution. These results show that, for a K degree polynomial, the FR scheme is $K+1$ order accurate for advection. Furthermore, they show that, for the same mesh, higher-order schemes are capable of approximating the solution orders of magnitude more accurately than lower-order schemes.

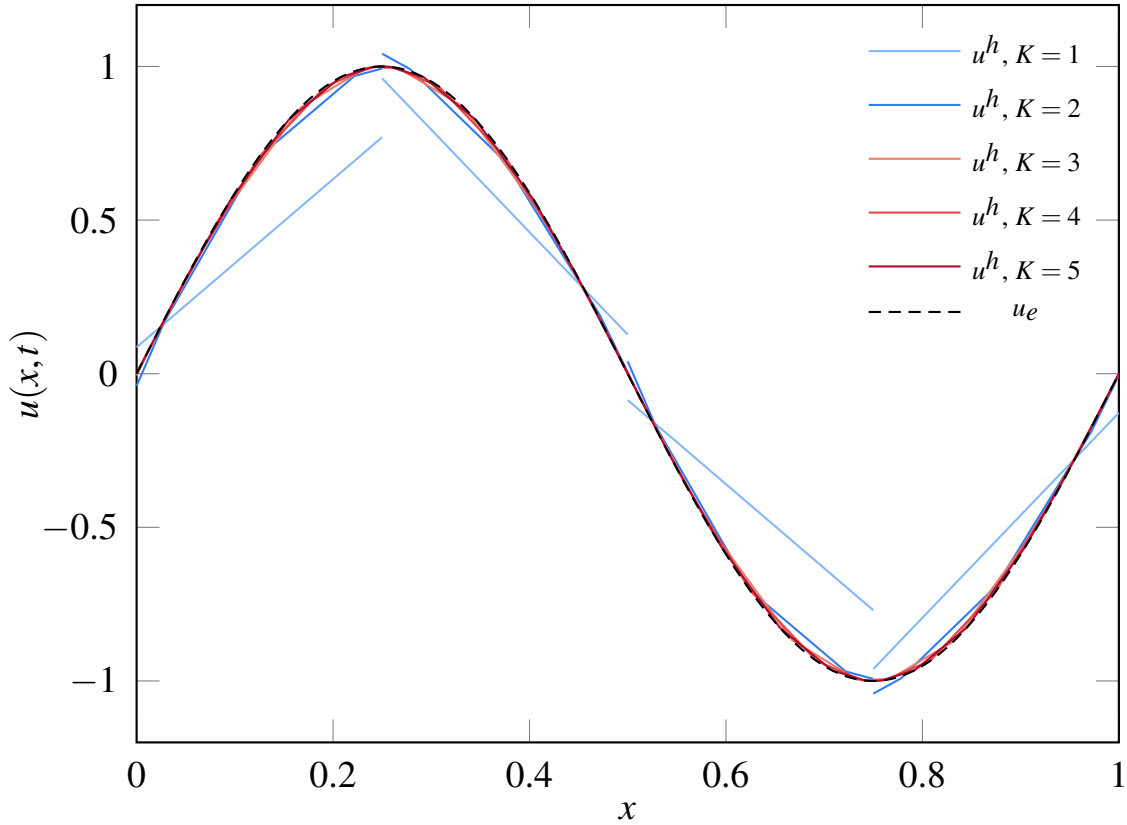


Figure 3.5. Exact and approximate solutions for a one-dimensional linear advection test case with an initial solution distribution $u_0 = \sin(2\pi x)$ at $t = 1$ for different polynomial degrees using a four-element mesh.

Table 3.2. L_2 norm of the solution error for a one-dimensional linear advection test case with an initial solution distribution $u_0 = \sin(2\pi x)$ at $t = 1$ for different polynomial degrees and meshes.

Polynomial Degree	Mesh Size	L_2 Error	L_2 Order of Accuracy
$K = 1$	4	1.856×10^{-1}	-
	8	3.331×10^{-2}	2.48
	16	6.279×10^{-3}	2.41
	32	1.384×10^{-3}	2.18
$K = 2$	4	1.132×10^{-2}	-
	8	1.277×10^{-3}	3.15
	16	1.564×10^{-4}	3.03
	32	1.947×10^{-5}	3.01
$K = 3$	4	9.561×10^{-4}	-
	8	5.791×10^{-5}	4.05
	16	3.627×10^{-6}	4.00
	32	2.266×10^{-7}	4.00
$K = 4$	4	6.980×10^{-5}	-
	8	2.232×10^{-6}	4.97
	16	6.959×10^{-8}	5.00
	32	2.156×10^{-9}	5.01
$K = 5$	4	4.540×10^{-6}	-
	8	7.097×10^{-8}	6.00
	16	1.108×10^{-9}	6.00
	32	1.730×10^{-11}	6.00

3.1.2 Linear Diffusion

Consider a general conservation law of the form

$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} = 0, \quad (3.19)$$

where $u = u(x, t)$ is the scalar solution variable with a given initial distribution $u(x, 0) = u_0(x)$, and $F = F(u, u_x)$ is the flux which is a function of the solution and its gradient. We can also express the diffusion equation using the second gradient of the scalar quantity as

$$\frac{\partial u}{\partial t} + D_f \frac{\partial^2 u}{\partial x^2}, \quad (3.20)$$

where D_f is the diffusion coefficient. Hence, we need to approximate both the first and second derivatives of the solution. Following Huynh [58], as in the linear advection problem, the

computational domain is divided into N_E non-overlapping elements with physical coordinates $x \in [x_L, x_R]$ and width h_j within each element, and reference coordinates $\xi \in [-1, 1]$. Following the same procedure outlined in the previous section, the K degree Lagrange polynomial representation of the solution on each element, which is continuous within the element, yet discontinuous on element intervals, is defined in physical space as

$$u_j(x, t) = \sum_{k=1}^{K+1} u_{j,k}(t) \phi_{j,k}(x), \quad (3.21)$$

or in the reference space as

$$u_j(\xi, t) = \sum_{k=1}^{K+1} u_{j,k}(t) \phi_k(\xi), \quad (3.22)$$

where $\phi_k(\xi)$ is defined as in Equation 3.11. We then define the continuous solution function by applying the correction procedure as

$$\hat{u}_j(\xi, t) = u_j(\xi, t) + [u_L^c - u_L] g_L(\xi) + [u_R^c - u_R] g_R(\xi), \quad (3.23)$$

where $\hat{u}_j(\xi, t)$ is the corrected continuous solution function with K degree polynomial, u_L^c , u_R^c are the common solution values on the left and right element boundaries, and $g_L(\xi)$, $g_R(\xi)$ are the correction functions constructed using the Radau polynomials. The common Riemann solutions can be evaluated using several techniques such as the Local Discontinuous Galerkin (LDG) approach [59], and Bassi and Rebay's first order (BR1) and second order (BR2) methods [60]. The drawback of BR1 schemes is that they achieve the expected order of accuracy of $K + 1$ only for odd values of K when applied to FR. Hence the BR2 method is used to evaluate the common viscous fluxes in the current work. The common interface solution for the BR2 scheme is defined as the average of the solution values on the left and right sides of an interface. Therefore, the common solution values on the interfaces of element E_j are defined as

$$u_L^c = \frac{1}{2}(u_{R,j-1} + u_{L,j}), \quad u_R^c = \frac{1}{2}(u_{R,j} + u_{L,j+1}). \quad (3.24)$$

Having all the necessary values to construct the correction term in Equation 3.23, we can proceed to calculate the first derivative of $\hat{u}_j(\xi, t)$ as

$$\frac{\partial \hat{u}_j(\xi, t)}{\partial \xi} = \frac{\partial u_j(\xi, t)}{\partial \xi} + [u_L^c - u_L] \frac{\partial g_L(\xi)}{\partial \xi} + [u_R^c - u_R] \frac{\partial g_R(\xi)}{\partial \xi}, \quad (3.25)$$

where $\partial \hat{u}_{j,k}(t)/\partial x$ is the corrected continuous first derivative function, which then can be mapped to the physical space to compute the value of the first derivative at each solution point as

$$u'_{j,k}(t) = \frac{\partial u_{j,k}(t)}{\partial x} = \frac{2}{h_j} \left(\frac{\partial \hat{u}_j(\xi_k, t)}{\partial \xi} \right), \quad (3.26)$$

where $u'_{j,k}(t)$ is the value of the first derivative of the solution evaluated at solution point $x_{j,k}$. These $K + 1$ derivatives can form a polynomial of degree K , which, similar to the polynomial representation of the solution, is continuous within an element yet discontinuous on element boundaries. Polynomial representation of the solution's first derivative can be defined as

$$u'_j(\xi, t) = \sum_{k=1}^{K+1} u'_{j,k}(t) \phi_k(\xi). \quad (3.27)$$

To define a second derivative function of the solution, we first need to construct a first derivative function that is continuous through the domain by applying the reconstruction procedure as

$$\hat{u}'_j(\xi, t) = u'_j(\xi, t) + [u_L^c - u'_L] g_L(\xi) + [u_R^c - u'_R] g_R(\xi), \quad (3.28)$$

where $\hat{u}'_j(\xi, t)$ is a continuous function within the entire domain which approximate the discontinuous first derivative of the solution $u'_j(\xi, t)$, and u_L^c , u_R^c are the common gradients that are specified using the BR2 method by averaging the partially corrected gradients on the left and right interfaces of element E_j as

$$\begin{aligned} u_L^c &= \frac{1}{2} \left(u'_{R,j-1} + [u_L^c - u_{R,j-1}] \frac{\partial g_L(\xi)}{\partial \xi} + u'_{L,j} + [u_L^c - u_{L,j}] \frac{\partial g_R(\xi)}{\partial \xi} \right), \\ u_R^c &= \frac{1}{2} \left(u'_{R,j} + [u_R^c - u_{R,j}] \frac{\partial g_L(\xi)}{\partial \xi} + u'_{L,j+1} + [u_R^c - u_{L,j+1}] \frac{\partial g_R(\xi)}{\partial \xi} \right). \end{aligned} \quad (3.29)$$

Additionally, the second derivative can be calculated as

$$\frac{\partial \hat{u}'_j(\xi, t)}{\partial \xi} = \frac{\partial u'_j(\xi, t)}{\partial \xi} + [u'_L - u'_L] \frac{\partial g_L(\xi)}{\partial \xi} + [u'_R - u'_R] \frac{\partial g_R(\xi)}{\partial \xi}, \quad (3.30)$$

where $\partial \hat{u}'_j(\xi, t)/\partial \xi$ is the corrected continuous second derivative function, which can be mapped to the physical space to compute the value of the second derivative on each solution point as

$$\frac{\partial u'_{j,k}(t)}{\partial x} = \frac{2}{h_j} \left(\frac{\partial \hat{u}'_j(\xi, t)}{\partial \xi} \right). \quad (3.31)$$

Finally, we use $\partial u'_{j,k}(t)/\partial x$ to update the conservation law and march in time with a temporal scheme.

3.1.3 Multidimensional Extension

Following the one-dimensional flux reconstruction approach, consider the multidimensional conservation law of the form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F} = 0, \quad (3.32)$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the vector of conserved variables with a given initial distribution $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_0$, $\mathbf{F} = \mathbf{F}(\mathbf{u}, \nabla \mathbf{u})$ is the viscous flux. Following Huynh [58], similar to the one-dimensional approach, the computational domain is divided into N_E non-overlapping elements with physical coordinates $\mathbf{x} \in [\mathbf{x}_L, \mathbf{x}_R]$, and reference coordinates $\boldsymbol{\xi} \in [-1, 1]$.

Following the flux reconstruction approach, the solution is represented by a discrete approximation on each element such that [53, 57]

$$\mathbf{u}(\mathbf{x}, t) \approx \mathbf{u}^h(\mathbf{x}, t) = \bigoplus_{j=1}^{N_E} \mathbf{u}_j^h(\mathbf{x}, t), \quad (3.33)$$

where $\mathbf{u}^h(\mathbf{x}, t)$ is the global piecewise continuous approximation of the solution and $\mathbf{u}_j^h(\mathbf{x}, t)$ is a continuous representation of the solution on one of N_E elements in the domain. We take the approximate solution on each element to be a polynomial nodal basis representation such

that

$$\mathbf{u}_j^h(\mathbf{x}, t) = \sum_{k=1}^{N_{DOF}} \mathbf{u}_{j,k}(t) \phi_{j,k}(\mathbf{x}), \quad (3.34)$$

where $\mathbf{u}_{j,k}(t)$ is the value of the solution at one of N_{DOF} solution nodal basis points on a given element, as shown by the red points in Figure 3.6, and $\phi_{j,k}(\mathbf{x})$ is its corresponding nodal basis function. Figure 3.7 shows the two-dimensional Lagrange polynomials for $K = 3$ polynomial degrees generated using Gauss points. This approach ensures the solution is continuous on each element but allows the solution to be discontinuous on the interfaces between elements [53].

Following the flux reconstruction extension to simplex element types [57, 61], the physical conservation law that must be satisfied in the discrete sense on each element is

$$\frac{\partial \mathbf{u}_j^h}{\partial t} + \nabla \cdot \mathbf{F}_j^h + \boldsymbol{\delta}_j = 0, \quad (3.35)$$

where $\mathbf{F}_j^h = \mathbf{F}_j^h(\mathbf{u}_j^h, \nabla \mathbf{u}_j^h)$ and $\boldsymbol{\delta}_j$ is a correction field on the element that is in the same polynomial space as the solution. This correction field is analogous to the divergence of the penalty functions introduced in the original FR scheme for tensor product elements [53]. Finally, by applying the conservation law at each of the solution points we obtain

$$\left. \frac{d\mathbf{u}_{j,k}^h}{dt} + (\nabla \cdot \mathbf{F}_j^h) \right|_{\mathbf{x}_{j,k}} + \boldsymbol{\delta}_{j,k} = 0, \quad (3.36)$$

where $\mathbf{x}_{j,k}$ is the location of corresponding solution point k on element j with $k = 1, \dots, N_{DOF}$, where N_{DOF} is the number of degrees of freedom for a specific degree polynomial, which was simply $K + 1$ for a one-dimensional element. Formulas to calculate the number of degrees of freedom for different element types are shown in Table 3.3. Following the FR formulation

$$\boldsymbol{\delta}_{j,k} = \frac{1}{|\Omega_j|} \sum_{f \in S} \sum_l \hat{\alpha}_{j,k,f,l} [\hat{\mathbf{F}}]_{j,f,l} S_f, \quad (3.37)$$

where $|\Omega_j|$ is the element volume, f is one of the faces on the element surface S , l is one of the flux points shown in blue in Figure 3.6, $\hat{\alpha}_{j,k,f,l}$ is a constant lifting coefficient, $[\hat{\mathbf{F}}]_{j,f,l}$ is the difference between a common Riemann flux at the flux point, and the value of the

internal flux, and S_f is the area of the face. Depending on the specification of these lifting coefficients, a number of different energy stable schemes can be obtained for general element types, including the spectral difference, spectral volume, and discontinuous Galerkin methods. In this study, we use lifting coefficients based on the nodal basis functions to recover the DG method [53, 57].

In the case of polynomial adaptive simulations using the FR approach, refer to Section 4.1, the faces of elements of different orders do not have consistent numbers of flux points. Hence, in the current study, we introduce an additional set of points on the faces of each element, referred to as Riemann points, as shown in green in Figure 3.6. In the case that the two elements at an interface are of the same order, these Riemann points are taken to be the same as the flux points, which are consistent between both elements. However, in the case of unmatched polynomial degrees, these Riemann points are taken to be those of the higher-degree element. On the lower-degree side, the common Riemann flux at the Riemann points is then least squares projected down to the lower-degree flux points. This is similar to de-aliasing, or overintegration, of the common flux on the lower-degree element. This approach was found to be more readily generalizable to different element types and interface orders than the interface element approaches previously used with FR [62]. At the interfaces we used a Rusanov Riemann solver and the second method of Bassi and Rebay [60] to compute the common gradient for the viscous fluxes.

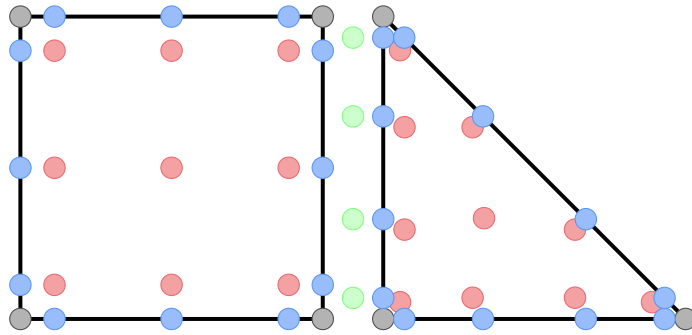
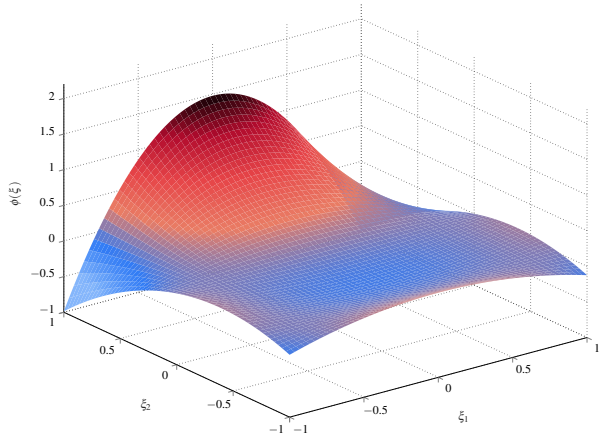
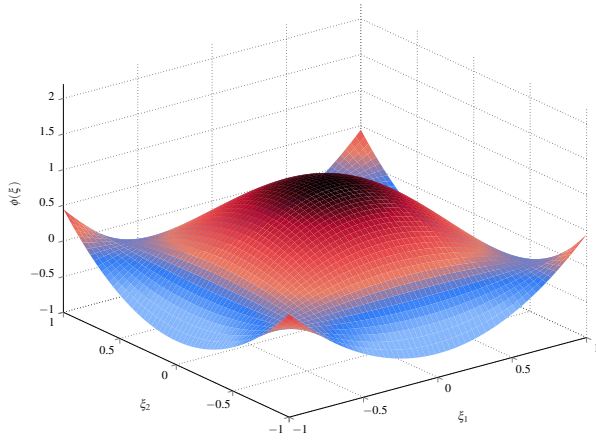
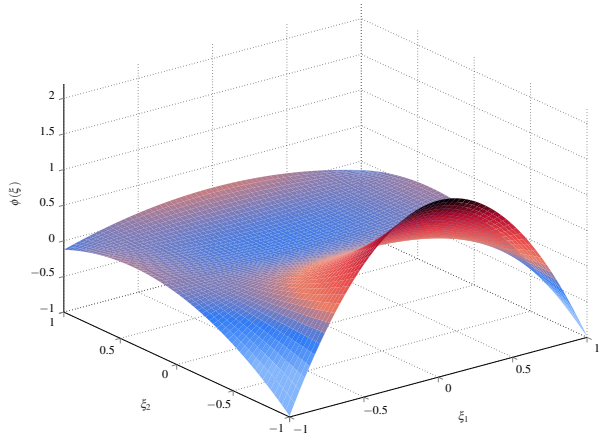
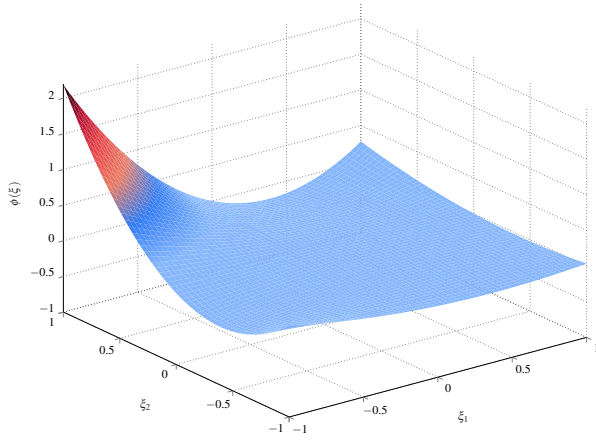
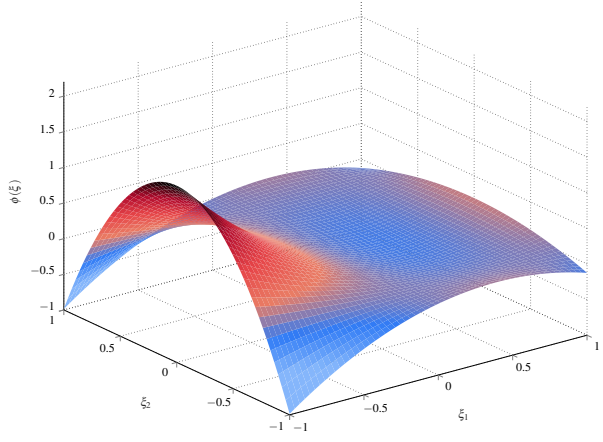
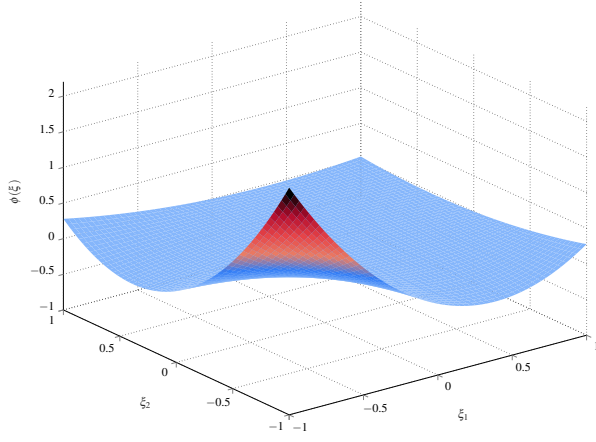


Figure 3.6. An example of a $K = 2$ quadrilateral element and a $K = 3$ triangular element including mapping points (grey), solution nodal basis points (red), flux points (blue), and Riemann points (green) on the interface between the elements. Elements are shown separated for clarity.



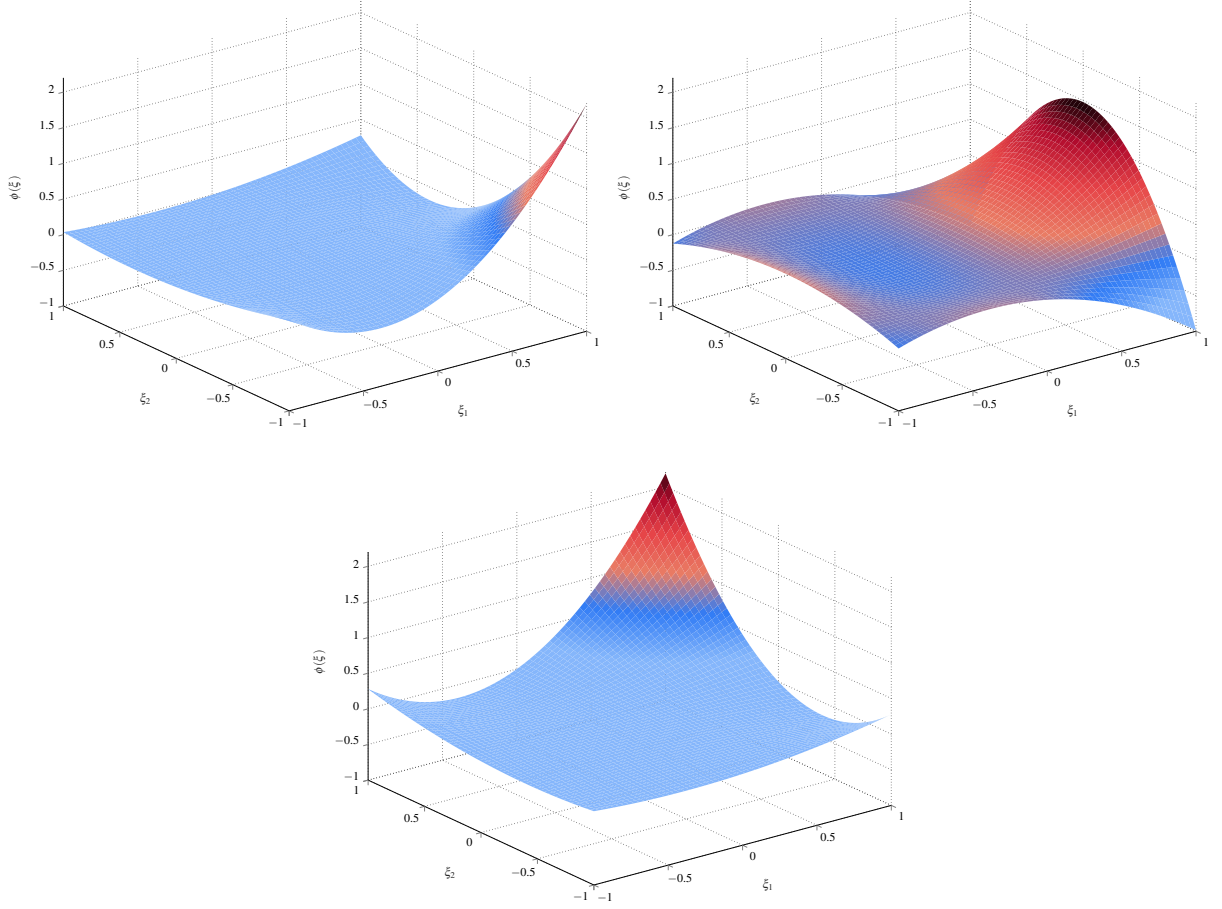


Figure 3.7. Two-dimensional nodal basis functions for a degree $K = 3$ polynomial using Gauss points.

3.1.4 Arbitrary Lagrangian Eulerian Extension

In this subsection, we present an ALE extension to the FR scheme for moving and deforming domains. Following the multi-dimensional flux reconstruction, consider the ALE form of the Navier–Stokes equations [56]

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F} + \mathbf{u} (\nabla \cdot \mathbf{v}_g) = 0, \quad (3.38)$$

where \mathbf{v}_g is the local mesh velocity, \mathbf{u} is the vector of conserved variables, and \mathbf{F} is the total flux consisting of the inviscid Euler fluxes, the viscous Navier–Stokes fluxes, and the ALE fluxes.

Following the flux reconstruction approach, as in the previous subsection, the solution is

Table 3.3. Number of degrees of freedom for different element types for degree polynomial K .

Element type	Dimensions	N_{DOF}
Line	1	$K + 1$
Triangular	2	$(K + 1) * (K + 2)/2$
Quadrilateral	2	$(K + 1) * (K + 1)$
Prism	3	$(K + 1) * (K + 2)/2 * (K + 1)$
Tetrahedral	3	$(K + 1) * (K + 2) * (K + 3)/6$
Hexahedral	3	$(K + 1) * (K + 1) * (K + 1)$

represented by a discrete approximation on each element such that [53, 57]

$$\mathbf{u}(\mathbf{x}, t) \approx \mathbf{u}^h(\mathbf{x}, t) = \bigoplus_{j=1}^{N_E} \mathbf{u}_j^h(\mathbf{x}, t), \quad (3.39)$$

where $\mathbf{u}^h(\mathbf{x}, t)$ is the global piecewise continuous approximation of the solution and $\mathbf{u}_j^h(\mathbf{x}, t)$ is a continuous representation of the solution on one of N_E elements in the domain. We take the approximate solution on each element to be a polynomial nodal basis representation such that

$$\mathbf{u}_j^h(\mathbf{x}, t) = \sum_{k=1}^{N_{DOF}} \mathbf{u}_{j,k}(t) \phi_{s,j,k}(\mathbf{x}), \quad (3.40)$$

where $\mathbf{u}_{j,k}(t)$ is the value of the solution at one of N_{DOF} solution nodal basis points on a given element, as shown by the red points in Figure 3.6, and $\phi_{s,j,k}(\mathbf{x})$ is its corresponding nodal basis function. This approach ensures the solution is continuous on each element but allows the solution to be discontinuous on the interfaces between elements [53]. We also use a polynomial nodal basis representation to map the mesh velocities from the element mapping nodes to the interior of the element as

$$\mathbf{v}_{g,j}^h(\mathbf{x}, t) = \sum_{m=1}^{N_g} \mathbf{v}_{g,j,m}(t) \phi_{g,j,m}(\mathbf{x}), \quad (3.41)$$

where N_g is the number of mapping points that define the element, as shown by the grey points in Figure 3.6, and $\phi_{g,i,j}(\mathbf{x})$ is the corresponding nodal basis function of the mapping points.

Following the flux reconstruction approach [53] and its extension to simplex element

types [57, 61], the physical conservation law that must be satisfied in the discrete sense on each element is

$$\frac{\partial \mathbf{u}_j^h}{\partial t} + \nabla \cdot \mathbf{F}_j^h + \boldsymbol{\delta}_j + \mathbf{u}_j^h (\nabla \cdot \mathbf{v}_{g,j}^h) = 0, \quad (3.42)$$

where $\mathbf{F}_j^h = \mathbf{F}_j^h(\mathbf{u}_j^h, \nabla \mathbf{u}_j^h)$ and $\boldsymbol{\delta}_j$ is a correction field on the element that is in the same polynomial space as the solution. This correction field is analogous to the divergence of the penalty functions introduced in the original FR scheme for tensor product elements [53].

Finally, by applying the conservation law at each of the solution points, we obtain

$$\left. \frac{d\mathbf{u}_{j,k}^h}{dt} + (\nabla \cdot \mathbf{F}_j^h) \right|_{\mathbf{x}_{j,k}} + \boldsymbol{\delta}_{j,k} + \mathbf{u}_{j,k}^h (\nabla \cdot \mathbf{v}_{g,j}^h) \Big|_{\mathbf{x}_{j,k}} = 0, \quad (3.43)$$

where $\mathbf{x}_{j,k}$ is the corresponding solution point location and $\boldsymbol{\delta}_{j,k}$ is defined as in Equation 3.37.

As explained in the previous subsection, at the interfaces we used a Rusanov Riemann solver and the second method of Bassi and Rebay [60] to compute the common gradient for the viscous fluxes. Furthermore, for efficiency, and following Liu et al. [56], we map the solution and discrete system of governing equations into a reference element with coordinates $\boldsymbol{\xi}$ using a one-to-one mapping $\mathbf{x} = M(\boldsymbol{\xi})$ such that $\boldsymbol{\xi} = M^{-1}(\mathbf{x})$. The mapping points are used to define a nodal polynomial representation of the mapping function M such that

$$\mathbf{x}_j^h(\boldsymbol{\xi}, t) = \sum_k^{N_g} \mathbf{x}_{g,j,k}(t) \phi_{g,j,k}(\boldsymbol{\xi}), \quad (3.44)$$

where $\mathbf{x}_j^h(\boldsymbol{\xi}, t)$ is the interpolated physical location, N_g is the number of mapping points that define the element, as shown by the grey points in Figure 3.6, $\phi_{g,i,j}(\mathbf{x})$ is the corresponding nodal basis function of the mapping points, and $\mathbf{x}_{g,j,k}(t)$ is the physical location of the mapping points as a function of time. The determinant of this mapping can be found at any location and time from

$$J = \left| \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \right|. \quad (3.45)$$

This allows all operations to be performed on the reference element and mapped back to the physical element as required [56]. In the current study we use quadratic mappings for all elements. We also note that in the case of pure translational motion the mapping metrics

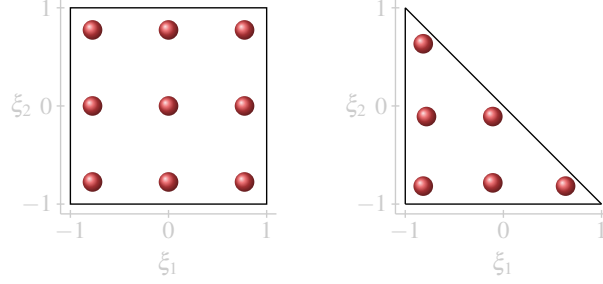
themselves are not a function of time. Hence, these metrics are only recomputed in this study for rotating or deforming domains.

3.1.5 Advantages of Flux Reconstruction

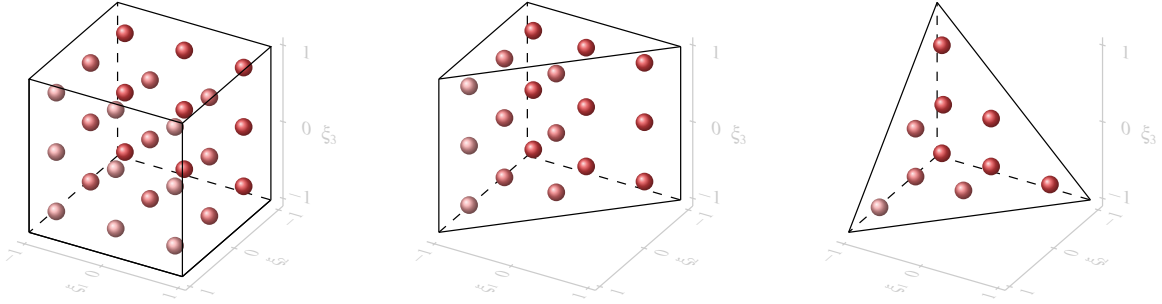
Flux reconstruction is appealing as it can recover several high-order methods depending on the choice of correction functions within a single framework. The FR method requires three computational operations, known as element-wise, point-wise direct, and point-wise indirect operations. Element-wise operations are performed on all solution points locally within each element by applying matrix multiplies, such as differentiation of polynomials. Point-wise direct operations are also performed locally within each element yet applied to every solution point on a point-by-point basis, such as evaluation of the flux on each solution point. Point-wise indirect operations, on the other hand, are performed on element interfaces on a point-to-point basis yet require interactions with adjacent elements, such as calculation of Riemann solvers on the element interfaces. While element-wise and point-wise direct operations are performed locally on each element, which enables rapid computations on the CPU, point-wise indirect operations require information from other elements, which comes with relatively slow indirect memory access. This is due to the fact that they require accessing multiple locations in memory before performing a calculation, a process that limits their performance by the memory bandwidth. In order to make a high-order numerical method efficient, we want the computations to be dominated by the element-wise and point-wise direct operations. The FR scheme is capable of adapting this behaviour as the number of solutions points within an element surpasses the number of points on element interfaces when the scheme uses higher polynomial orders. This allows the FR scheme to achieve about 55% of peak FLOP/s, while the traditional FV methods, dominated by point-wise indirect operations, are capable of achieving only about 3% of peak FLOP/s [47]. This characteristic makes the FR scheme promising for high-order simulations.

Furthermore, the FR scheme can be extended to multi-dimensions and different structured and unstructured elements. It was extended to three-dimensional problems by Haga and Wang [61] and to mixed element types by Wang and Gao [63]. These extensions enable the FR scheme to be used for LES and DNS simulations of transitional and turbulent flows. Figure

3.8 shows the schematic view of solution points on different element types for $K = 2$ degree polynomial. Finally, since FR uses an element-wise approach, it enables the application of polynomial adaptation to achieve high-order accuracy locally on each element. Please refer to Section 4.1 on adaptation techniques.



(a) Quadrilateral element. (b) Triangular element.



(c) Hexahedral element.

(d) Prism element.

(e) Tetrahedral element.

Figure 3.8. Two-dimensional and three-dimensional schematic view of solution points for a $K = 2$ degree polynomial for different reference element types.

3.2 Temporal Schemes

In the previous chapter, we explained the spatial discretization using the FR scheme. The semi-discretized conservation law reads the form

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{R}, \quad (3.46)$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the vector of conserved variables, $\mathbf{R} = \mathbf{R}(\mathbf{u})$ is the residual, which is our semi-discrete space operator, and t is time. Ultimately, we want to advance the solution in time as

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{\mathbf{u}^{t+1} - \mathbf{u}^t}{\Delta t}, \quad (3.47)$$

where \mathbf{u}^{t+1} is the solution at the next time level, \mathbf{u}^t is the solution at the current time level, and Δt is the time step size. Hence, a temporal discretization is required to obtain a fully-discrete scheme to advance in time.

Two distinct temporal discretization approaches can be used, namely explicit and implicit schemes. Explicit temporal discretization schemes use *a-priori* solutions from the current time level to approximate the solution at the next time level. These schemes are widely used partly due to their simple implementation and their efficiency in terms of computational cost and memory requirements when extended to higher orders of temporal accuracy. However, they have the disadvantage of conditional stability, making them unsuitable for stiff problems. In other words, for a stable solution, the time step size is limited by a Courant-Friedrichs-Lewy (CFL) condition where $\text{CFL} = a\Delta t/\Delta x$, and a is the characteristic wave speed of the system. Hence, for a fixed advection velocity and grid size Δx , the maximum time step size is limited by a maximum CFL number, which increases the cost of the scheme as a trade-off for acquiring a stable scheme. Implicit temporal schemes, on the other hand, use the unknown solution from the following time level to approximate the solution at the next time level, requiring solving a coupled system of non-linear equations at every iteration. Hence, implicit time-stepping schemes are computationally more expensive per step and require a larger amount of memory compared to explicit temporal schemes. However, they have the advantage of being unconditionally stable, which allows for a relatively larger time step. This advantage makes them suitable for stiff problems.

The most common explicit temporal schemes are multistep explicit Runge-Kutta (ERK) methods. They use s intermediate stages to approximate the solution at the next time level. Considering Equations 3.46 and 3.47, we can form the fully discrete scheme as

$$\frac{\mathbf{u}^{t+1} - \mathbf{u}^t}{\Delta t} = \mathbf{R}(\mathbf{u}^t), \quad (3.48)$$

Table 3.4. Butcher tableau for a general Runge-Kutta scheme used in the current study.

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

where the residual is evaluated at the current time level. By rearranging, we can obtain the solution at the next time level as

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \mathbf{R}(\mathbf{u}^t)\Delta t, \quad (3.49)$$

where as explained earlier, we only need the vector of solutions and evaluated residual at the current time level to update the solution at the following time level. Following the Runge-Kutta formulation [64], the solution at the next level can be expressed as

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \Delta t \sum_{m=1}^s b_m \mathbf{R}(\mathbf{u}^m), \quad (3.50)$$

where $\mathbf{R}(\mathbf{u}^m)$ is the residual evaluated with the solution \mathbf{u}^m at an intermediate Runge-Kutta stage m , such that

$$\mathbf{u}^m = \mathbf{u}^t + \Delta t \sum_{n=1}^s a_{mn} \mathbf{R}(\mathbf{u}^n), \quad (3.51)$$

where Equations 3.50 and 3.51 are the general form of an s -stage Runge-Kutta method, b_m is the weight coefficient, and coefficient a_{mn} can be obtained from the stage coefficient matrix A . The Runge-Kutta reference Butcher tableau is shown in Table 3.4 where c is a vector containing the intermediate time steps and $c_m = \sum_{n=1}^s a_{mn}$.

The stage coefficient matrix A is strictly lower triangular for explicit Runge-Kutta methods, $a_{mn} = 0$ for $m \leq n$, since explicit temporal schemes only rely on solutions from previous stages. On the other hand, for implicit Runge-Kutta methods, the stage coefficient matrix A is not strictly lower triangular, which, as mentioned earlier, forms a system of s non-linear equations that need to be solved at every iteration, making them computationally expensive. If A is a lower triangular matrix with identical diagonal elements, $a_{mn} = 0$ for

$m < n$ and $a_{mm} = \gamma^*$, the temporal scheme is called a Singly Diagonally Implicit Runge-Kutta (SDIRK) method [64, 65]. The general Butcher tableaux for the SDIRK_{2,2} and SDIRK_{3,3} methods are shown in Table 3.5, where γ^* is defined as $(2 - \sqrt{2})/2$ for SDIRK_{2,2} [65], and is a root of $x^3 - 3x^2 + 3x/4 - 1/6 = 0$ for SDIRK_{3,3} [66]. The other parameters are defined

Table 3.5. Butcher tableaux for the second two-stage and three-stage third order singly diagonally implicit Runge-Kutta methods.

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|cc} \gamma^* & \gamma^* & \\ 1 & (1 - \gamma^*) & \gamma^* \\ \hline & (1 - \gamma^*) & \gamma^* \end{array}$$

(a) SDIRK_{2,2} method.

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|ccc} \gamma^* & \gamma^* & & \\ c & (c - \gamma^*) & & \gamma^* \\ 1 & (1 - b - \gamma^*) & b & \gamma^* \\ \hline & (1 - b - \gamma^*) & b & \gamma^* \end{array}$$

(b) SDIRK_{3,3} method.

as [65]

$$\begin{aligned} b &= \frac{-3\alpha^2}{4\beta}, \\ c &= \frac{2 - 9\gamma^* + 6\gamma^{*2}}{3\alpha}, \end{aligned} \tag{3.52}$$

where $\alpha = (1 - 4\gamma^* + 2\gamma^{*2})$ and $\beta = (-1 + 6\gamma^* - 9\gamma^{*2} + 3\gamma^{*3})$. For the SDIRK methods, Equation 3.51 requires the solution of a non-linear system of equations. For this, we use a quasi-Newton approach with the Generalized Minimal Residual Method (GMRES) and an additive-Schwarz preconditioner in the Portable, Extensible Toolkit for Scientific Computing (PETSc) [67, 68, 69]. The Butcher tableaux for the RK_{4,4}, SDIRK_{2,2}, and SDIRK_{3,3} used in the current study are given in Table 3.6.

An optimized Runge-Kutta temporal scheme with more RK stages is introduced for the FR scheme by Hedayati Nasab et al. [70]. This scheme is optimized to allow for a larger time step size without jeopardizing stability. Hence, it has the advantage of reducing the final computational cost by using fewer time-steps with a relatively large time-step size. A speed-up factor of 2 is achieved compared to classical RK_{4,4} scheme, making them suitable for

Table 3.6. Butcher tableaus for Runge-Kutta methods.

$$\begin{array}{c|ccc} & 0 & & \\ \mathbf{c} | \mathbf{A} & 1/2 & 1/2 & \\ \mathbf{b} & 1/2 & 0 & 1/2 \\ & 1 & 0 & 0 & 1 \\ \hline & & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

(a) Classical RK_{4,4} method.

$$\begin{array}{c|cc} & 0.2928932188 & 0.2928932188 \\ \mathbf{c} | \mathbf{A} & 1 & 0.7071067811 & 0.2928932188 \\ \mathbf{b} & & 0.7071067811 & 0.2928932188 \\ \hline & & & \end{array}$$

(b) SDIRK_{2,2} method.

$$\begin{array}{c|ccc} & 0.4358665215 & 0.4358665215 & \\ \mathbf{c} | \mathbf{A} & 0.7179332607 & 0.2820667392 & 0.4358665215 \\ \mathbf{b} & 1 & 1.2084966490 & -0.644363171 & 0.4358665215 \\ \hline & & 1.2084966490 & -0.644363171 & 0.4358665215 \end{array}$$

(c) SDIRK_{3,3} method.

three-dimensional turbulent test cases in this study. The Butcher tableaus for second-order and forth-order optimized Runge-Kutta methods with 12 stages used in this study can be found in the electronic supplementary material provided by Hedayati Nasab et al. [70].

The following chapter introduces the load-balanced adaptation technique used in this work. We start by discussing the state-of-the-art adaptation strategies, including the adjoint, truncation-error, and feature-based adaptation techniques. We then introduce a novel vorticity-based adaptation indicator. Finally, we conclude the chapter by discussing the method employed in the study for load balancing.

Chapter 4

Dynamically Load Balanced Adaptation Algorithm

In this chapter, we discuss different adaptation strategies, and advantages and disadvantages of each method. We then introduce a novel non-dimensional vorticity-based indicator used in this study for p -adaptation. Finally, we present the load balancing technique employed when adaptation is performed.

4.1 Adaptation Techniques

In the context of massively separated turbulent flows, such as slats, flaps, and landing gear, *a-priori* information about the flow-field and its related mesh resolution requirements are often not readily available. Furthermore, resolution requirements may change as the simulation evolves. In the absence of *a-priori* information about the flow field, a prohibitively large total number of DOF can be required to capture the complex physics governed by the Navier-Stokes equations, making such simulations inherently computationally expensive. Although a simulation with a greater number of DOF usually benefits from improved accuracy, it can suffer considerably from higher computational costs. In practice, a relatively large number of DOF is needed only in regions where higher resolution is required for an accurate discrete approximation of the solution. Hence, the number of DOF can be increased or decreased locally in order to minimize the overall computational cost, yet attaining almost

the same level of accuracy, a practice which is called adaptation. Adaptation can be classified into three types that could be used independently or combined: r -adaptation, in which grid points are rearranged to change the element size according to the complexity of the flow field; h -adaptation, in which the mesh connectivity is modified to increase or decrease the number of elements; and p -adaptation, in which the order of accuracy of the scheme is locally increased or decreased by changing the degree of the solution polynomials [71]. In this study, we are specifically interested in element-wise polynomial representations of the flow using the FR approach, owing to the fact that such a representation allows us to dynamically adapt the degree of the solution polynomial, p -adaptation, simultaneously increasing the local resolution of an element and its order of accuracy without requiring any complex mesh splitting operations. An important prerequisite to deploying any adaptation strategy is a resolution indicator, which, in the case of p -adaptation, is a technique to indicate elements where the use of higher degree polynomials is needed for an accurate numerical approximation of the solution. These techniques can be classified into three main categories of adjoint-based [72, 73, 74], truncation-error-based [75, 76, 77], and feature-based [71, 78, 79, 80, 81] adaptation indicators. Whereas some are designed to resolve turbulent features of the flow, such as the separated wake and the turbulent boundary layer regions, most have been applied primarily to steady flow problems. It should be noted that, although using a good adaptation strategy will either minimize the error for a given number of DOF, or provide the same level of accuracy with fewer DOF, the cost allocated to compute the adaptation indicator should be reasonable so that the adaptation process stays efficient. In this section, state-of-the-art adaptation strategies and their pros and cons are discussed.

4.1.1 Adjoint-Based Adaptation Indicators

Scalar outputs such as lift or drag coefficients approximated using numerical methods have always been interesting in an engineering context. In fact, the main goal of CFD is often to estimate these outputs with minimal error. An adjoint solution can link the local residual to these scalar outputs. Adjoint solutions have been used in a wide variety of applications, including design optimization, optimal control, and error estimation [72, 82, 83]. It can also be used as a reliable adaptation indicator by estimating the error and then adapting the solution

polynomial degree if the error exceeds a predefined constant. One of the advantages of adjoint-based indicators is that the adjoint solution is linked to the output function of interest; hence, it is guaranteed that they detect the elements that directly contribute to the error. Consequently, the algorithm will not tend to flag regions that could have no effect on accuracy improvement. Definitely, there are some drawbacks to this method. First, the adjoint solution could be inherently expensive to solve because of the inverse of the Jacobian matrix embedded in its formulation, so it conflicts with the main objective of the adaptation concept to develop an algorithm that itself will be cheap to solve and will not add considerable extra computational burden to the simulation. Furthermore, adjoint-based indicators can not be used for chaotic turbulent flows [83, 84, 85, 86, 87, 88], which narrows their applications and makes them unsuitable for our purpose, which is to develop a general package that could be applied to all flow regimes.

4.1.2 Truncation-Error-Based Adaptation Indicators

In this section, a recently proposed truncation error adaptation methodology, known as τ -estimation, introduced by Rubio et al. [77] is reviewed. In this technique, regions that require refinement are flagged by estimating the truncation error. Rubio et al. introduced this method and went through different scenarios to compare the accuracy as well as the simulation cost [76]. Results included two test cases of an inviscid NACA 0012 and a viscous flat plate boundary layer. The accuracy of the flow solution and the required total number of degrees of freedom are compared. It is shown that the truncation error adaptation technique provides meshes with polynomial order distribution that leads to a smaller number of DOF, yet provides almost the same level of accuracy as if a high uniform degree had been applied.

Consider the following conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F}(u) = 0. \quad (4.1)$$

It is assumed that the solution converged in time such that $u_t = 0$, so the residual is independent of time.

If we define the truncation error as the difference between the discrete PDE and the exact

PDE operator, both applied to the exact solution u_e [89], we get

$$\tau^K = \mathcal{R}^K(u_e) - \mathcal{R}(u_e) \quad (4.2)$$

where τ^K is the truncation error for polynomial order K , \mathcal{R} is the exact partial differential operator, \mathcal{R}^K is the discrete spatial partial differential operator of order K , and u_e is the exact solution.

Flux reconstruction allows a separation between the interior and inter-element contributions. This allows us to define the isolated discrete PDE operator by not solving the Riemann problem on element interfaces. So every element would be isolated from its neighbours. The advantage of calculating the isolated discrete PDE is that it only considers the interior element contributions of the truncation error, which is shown to perform better than the truncation error [77].

To approximate the truncation error, the problem is solved with a higher polynomial degree, P , on each element. Then the error is estimated for all lower orders, $K = 1, 2, 3, \dots, P-1$, for example, τ_P^K would be the truncation error on a coarse mesh K using a finer simulation with order P . To obtain the estimated solution on the fine mesh, a quasi-*a priori* method is used for the sake of time efficiency. It means that the simulation is continued until the convergence tolerance is reached. Consequently, the fine solution is a non fully time-converged solution. The approximated truncation error is defined as

$$\tau_P^K \equiv \mathcal{R}^K(I_P^K \tilde{u}^P) - \bar{I}_P^K \mathcal{R}^P(\tilde{u}^P) \quad (4.3)$$

where \tilde{u}^P is the non fully time-converged solution estimated on the fine mesh, \bar{I}_P^K is the transfer operator of the residual from order P to K , and I_P^K is an operator to interpolate from a fine mesh of order P to a coarse mesh of order K .

If we substitute the definitions of the discretization and iteration errors into Equation 4.3 and expand it using the Taylor series, the truncation error will read

$$\tau_P^K \equiv \tau^K - \frac{\partial \mathcal{R}^K}{\partial u_e^K} \bigg|_{u^K} \epsilon_h^P + \mathcal{O}(\epsilon_h^P)^2 + \mathcal{O}(\epsilon_{itr}^P)^2 \quad (4.4)$$

where ϵ_h^P is the discretization error defined as the difference between the exact solution of the PDE and the exact solution of the discretized PDE, and ϵ_{itr}^P is the iteration error defined as the difference between the fully time-converged solution and the non fully time-converged solution, both applied to the discretized PDE. The same procedure can be applied to the isolated truncation error.

$$\begin{aligned}\epsilon_h^P &= u_e - u^P \\ \epsilon_{itr}^P &= u^P - \tilde{u}^P\end{aligned}\tag{4.5}$$

As mentioned, the truncation error would be accurate only if the asymptotic rate of convergence is reached. So, we only proceed with the simulation until we reach a solution that is not fully converged in time yet stable enough to be used as an accurate reference for truncation error estimation. Obviously, the cost of computing the solution on the fine mesh u^P is huge. In fact, the sole purpose of adaptation strategies is to avoid solving for the fine mesh. It is shown in [90] that although computing the matrix inversion embedded in equation 4.3 is dramatically high, the error estimation cost is negligible compared to the cost of converging the solution on the fine mesh.

The second term on the right hand side of Equation 4.3 acts as a correction term for the iteration error ϵ_{itr}^P . Notice the high cost of its computation since it requires the solution of a linear system. A quasi-*a priori* estimation without the correction term has been performed by Kompenhans et al. [76] in order to compare the accuracy and the computation cost.

If the correction term is not computed, the difference between the exact and estimated truncation error would be defined as

$$\tau_P^K \equiv \tau^K - \left. \frac{\partial \mathcal{R}^K}{\partial u^K} \right|_{u^K} (\epsilon_h^P + \epsilon_{itr}^P) + \mathcal{O}(\epsilon_h^P)^2 + \mathcal{O}(\epsilon_{itr}^P)^2\tag{4.6}$$

One might think the computational cost could be reduced if we neglect this term. However, in reality, by neglecting the correction term, not only is the accuracy sacrificed but also the computational cost gets higher [76]. It is simply because of the fact that by using a poor estimation of the truncation error, the adaptation algorithm will flag the regions where adaptation is not really necessary, leading to a larger number of DOF and a higher

computational cost. Consequently, although computing the correction term is computationally expensive, it will lead to a lower number of DOF, making the algorithm more cost-efficient.

As stated, the main idea for this adaptation strategy is to apply a quasi-*a priori* method by obtaining a non fully time-converged solution with a high polynomial order P , and then interpolating it to estimate the truncation error for all lower polynomial degrees $K < P$. The elements in the adapted mesh then use the lowest possible polynomial order that satisfies the required refinement criteria. So if we define the maximum polynomial degree used in the simulation as K_m , we first begin by integrating in time on the fine mesh with a uniform K_m polynomial degree until the asymptotic rate of convergence is reached. Then we calculate the truncation errors using Equation 4.3 and perform the adaptation as

$$\begin{aligned}
& K = 1, \\
& \text{get } \tau_P^2 \text{ if } \left\| \tau_P^2 \right\|_{\infty} \leq \tau_{max} \text{ then } K = 2, \\
& \text{get } \tau_P^3 \text{ if } \left\| \tau_P^3 \right\|_{\infty} \leq \tau_{max} \text{ then } K = 3, \\
& \dots \\
& \text{get } \tau_P^{P-1} \text{ if } \left\| \tau_P^{P-1} \right\|_{\infty} \leq \tau_{max} \text{ then } K = K_m
\end{aligned} \tag{4.7}$$

where τ_{max} is the desired maximum threshold. The solution is then interpolated to the new adapted mesh, and the simulation continues.

For the test case in [77], it is shown that the truncation error approach provides satisfactory resolution near the trailing and leading edge, as well as the wake region, while it lowers the simulation runtime by avoiding high polynomial degrees in elements that have no contribution to the truncation error. Also, it is shown in [76] that although the quasi-*a priori* method with the correction function is more expensive per iteration, the total run time is lower compared to the quasi-*a priori* method without the correction function.

This technique has been applied only to steady flows, a limitation that inherently narrows its application. Furthermore, this algorithm requires us to compute/approximate the fine-mesh solution, which is not only expensive, it also requires a relatively large Random-Access Memory (RAM) to store the fine solutions.

4.1.3 Feature-Based Adaptation Indicators

Feature-based adaptation methods developed based on physical quantities such as pressure, entropy, density, velocity, temperature, or vorticity have been used broadly to improve the accuracy of numerical solutions for unsteady flow. These methods tend to flag regions that need adaptation by calculating the physical quantities or their gradients. Feature-based adaptation methods are very popular because of their simplicity and their ability to solve unsteady problems; however, they have some drawbacks. These indicators do not rely on the direct relation between the computed gradient and numerical errors and, therefore, the resulting mesh does not guarantee a reduction of error [77]. In other words, they might indicate regions to be adapted, which would not contribute to accuracy improvement and would only make the simulation more expensive. For instance, if an unsteady flow around an airfoil is the case study, then a vorticity-based indicator could be a good choice as an adaptation indicator if the output function of interest is mostly sensitive to the vorticity magnitude. However, if there were shocks generated along the airfoil, they would not be captured by this indicator which could result in a numerical error if one wants to study the pressure distribution, for instance. However, this can be addressed by defining multiple gradient indicators to capture various flow features based on the output function of interest.

As mentioned earlier, the main drawback of the feature-based indicators is that the choice of physical quantity depends on the purpose of the simulation. An example of a feature-based adaptation indicator is an entropy residual indicator studied by Ching et al., borrowed from an approach developed by Fidkowski and Roe [74], which has developed an entropy adjoint approach in which the residual weighted by the entropy variables provides an error estimate for a global entropy balance, to illustrate how the entropy residual performs as a general feature-based indicator in different configurations. The entropy residual is defined as [71, 91]

$$R_{ent} = \frac{\partial(\rho\mathcal{S})}{\partial t} + \nabla \cdot (\rho\mathcal{S}\mathbf{v}) - \frac{p}{\rho}[\nabla \cdot (k\nabla T) + \boldsymbol{\sigma} \circ \nabla \mathbf{v}], \quad (4.8)$$

where the first term on the right-hand side corresponds to the inviscid Euler term, while the second term corresponds to the viscous term. The fourth term is a Hadamard product

$\sigma \circ \nabla \mathbf{v} = \sigma_{ij} \partial v_j / \partial x_i$. The physical entropy, \mathcal{S} , is defined as

$$\mathcal{S} = \ln \frac{p}{\rho^\gamma}. \quad (4.9)$$

The entropy residual, $R_{ent}(\mathbf{u})$, can be used as a good indicator to determine how well the numerical solution approximates the true solution. For a sufficiently smooth solution, $|R_{ent}(\mathbf{u})|$ is expected to be very close to zero. Having defined the entropy residual, we can calculate the maximum residual within an element j with a cell volume of Ω_j and time interval of length T_{int} as

$$R_{ent,j} = \frac{1}{T_{int}|\Omega_j|} \int_t^{t+T_{int}} \int_{\Omega} |R_{ent}| d\Omega dt. \quad (4.10)$$

At each time interval of length T_{int} , $Y\%$ of all elements with the highest $\mathcal{R}_{ent,j}$ are subject to a higher resolving power by increasing the degree of solution polynomial. Of these elements, only those with polynomial order less than K_m can be adapted to higher degree polynomials. So, at each step, a fixed number of elements are refined. This will continue until the target number of degrees of freedom is reached. From this point onward, at each time interval, $F\%$ of all elements with the highest $\mathcal{R}_{ent,j}$ are selected for further refinement. Again, only those elements that have not reached the maximum polynomial degree, K_m , can be refined further. In order not to waste the computational power, elements with the lowest $\mathcal{R}_{ent,j}$ and K_n greater than K_{min} , are coarsened. The number of elements that are selected for coarsening is defined such that the DOF is maintained. At every adaptation stage, the polynomial order increment is set to one. T_{int} , Y , F , K_m , K_{min} , and DOF are all inputs that could differ for any simulation. Usually, the initial polynomial order is set to K_{min} , which itself is defined as one in most simulations. Although the entropy residual polynomial adaptation indicator provides an accurate approximation for 2D LES, the computational cost of Equation 4.10 is quite high as it requires an integral in space and time. Hence, it is not in line with the objectives of our research. Furthermore, Ching et al. have only assessed this strategy for 2D simulations, so the cost efficiency of this algorithm for 3D LES, which is expected to be relatively low due to the high computational cost of Equation 4.10, remains unknown.

4.2 Vorticity-Based Polynomial Adaptation

As mentioned earlier, feature-based adaptation methods are popular due to their simplicity, low computational cost, and their applicability to unsteady problems. This study explores the utility of a novel non-dimensional vorticity-based indicator for p -adaptation when solving the unsteady Navier-Stokes equations. We begin by computing the maximum non-dimensional vorticity magnitude at any solution point within an element according to

$$\kappa_j = \max_{1 \leq k \leq N_{DOF}} \bar{\omega}_{j,k}, \quad (4.11)$$

where $\bar{\omega}_{j,k}$ is the non-dimensionalized counterpart of vorticity magnitude

$$\bar{\omega}_{j,k} = \frac{|\boldsymbol{\omega}_{j,k}| \Delta x_{max,j}}{U_\infty}, \quad (4.12)$$

where $\Delta x_{max,j}$ is the maximum mesh dimension of the element, N_{DOF} is the number of solution points within the element, U_∞ is the free-stream velocity, and $\boldsymbol{\omega}_{j,k}$ is the vorticity at solution point k on element j defined as

$$\boldsymbol{\omega}_{j,k} = \nabla \mathbf{v}_{j,k} = \begin{bmatrix} \partial v_z / \partial x_y - \partial v_y / \partial x_z \\ \partial v_z / \partial x_x - \partial v_x / \partial x_z \\ \partial v_y / \partial x_x - \partial v_x / \partial x_y \end{bmatrix}_{j,k}. \quad (4.13)$$

We then adapt the solution polynomial degree on any element using Algorithm 1, where $K_{j,c}$ is the current degree polynomial of element j , before applying the adaptation, $K_{j,r}$ is the required degree polynomial of element j for an accurate solution determined by the p -adaptation algorithm, ϵ is a constant *a-posteriori* resolution threshold, ϑ is a constant that controls the relative threshold for different solution polynomial degrees and K_m is the maximum polynomial degree. Following this approach, elements with large vorticity magnitudes relative to the effective mesh resolution are adapted to higher-degree polynomials, increasing their resolving power. To minimize the aliasing errors induced by the change of higher-order solutions to lower orders, it is important to keep the polynomial adaptation within increments of one at each adaptation call.

```

for  $j \in [1, 2, \dots, N_E]$  do
  Read the current polynomial degree;
   $K_{j,c} = K_j$ ;
  Find the required polynomial degree;
  Compute  $\kappa_j$ ;
  if  $\kappa_j < \vartheta_1 \epsilon$  then
     $K_{j,r} = 1$ ;
  else
    for  $n \in \{1, 2, \dots, K_m - 1\}$  do
      if  $\kappa_j > \vartheta_{n+1} \epsilon$  then
         $K_{j,r} = n + 1$ ;
      end
    end
  end
  Set the adaptation increment to one;
  if  $K_{j,r} - K_{j,c} > 1$  then
     $K_j = K_{j,c} + 1$ ;
  else if  $K_j - K_{j,c} < -1$  then
     $K_j = K_{j,c} - 1$ ;
  else
     $K_j = K_{j,r}$ ;
  end
end

```

Algorithm 1: Vorticity-based p -adaptation algorithm.

4.3 Load Balancing

4.3.1 Introduction

In the context of massively parallel computing, where we tend to exploit the computational power of state-of-the-art hardware architectures, the maximum efficiency of an adaptation algorithm can only be reached if the overheads of parallel computation are minimized. One of the primary sources of overhead is the time spent idle by processors due to unbalanced load distributions. This is because upon applying p -adaptation, the degree of solution polynomials will be increased or decreased locally, leaving the initial mesh unbalanced. Figure 4.1 shows a mesh initially partitioned with METIS allocating almost the same number of elements to each processing unit. Recently, dynamic load balancing techniques have been exploited to enhance the effect of adaptation techniques [92, 93, 94]. In this study, we circumvent this overhead by dynamically balancing the computational load throughout the domain as the

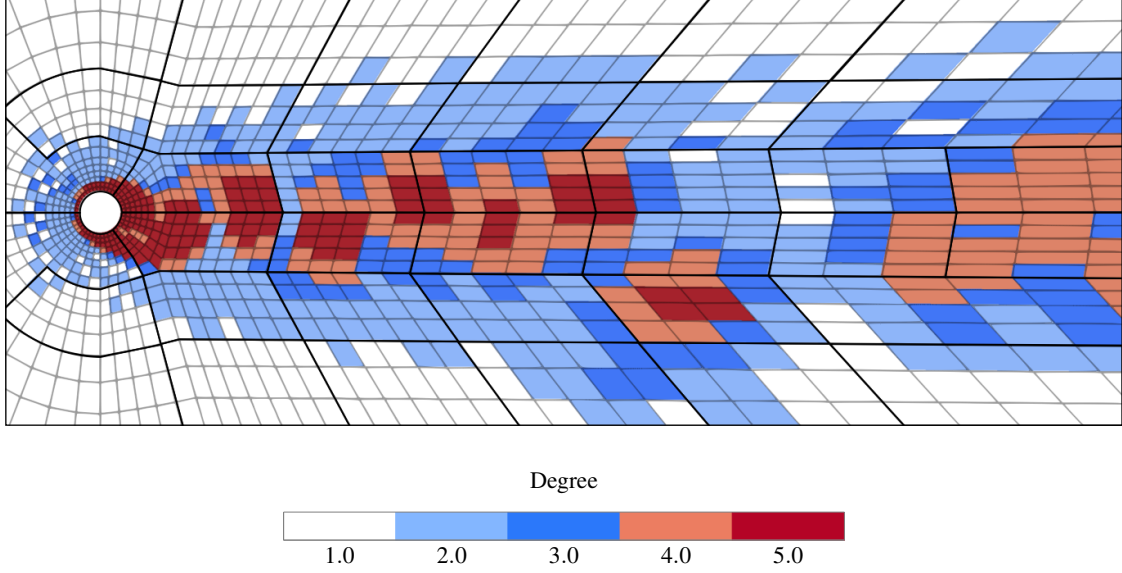


Figure 4.1. Schematic view of the initial mesh partition for a polynomial adaptive simulation with an unbalanced computational load distribution.

simulation evolves. Figure 4.2 shows different regions in terms of computational load and compares the initial load-unbalanced mesh with the load-balanced mesh.

4.3.2 ParMETIS

In this study, a load balancing technique is employed using the repartitioning routine of the ParMETIS Parallel Graph Partitioning and Sparse Matrix Ordering library developed by Karypis et al. [95] to circumvent the overhead due to an imbalanced mesh. Using this approach, the mesh is considered as a weighted graph, where elements and the computational burden are its vertices and weight accordingly. The computational burden varies based on the element type, solution polynomial degree, number of dimensions, and number of conserved variables. The compute time for a single element is used as the reference to define the element weight.

ParMETIS repartitioning routine takes the adjacency structure of the mesh, `xadj` and `adjncy`, weight of elements `vwgt`, and element distribution among processors `vtxdist`. It then gives a vector of the locally-stored vertices, `part`, as the output. ParMETIS uses Compressed Storage Format (CSF) to represent the graph structure. In a parallel simulation, each processor stores the adjacency of its elements into an array of size $N_{d,C}$, `adjncy`[$N_{d,C}$],

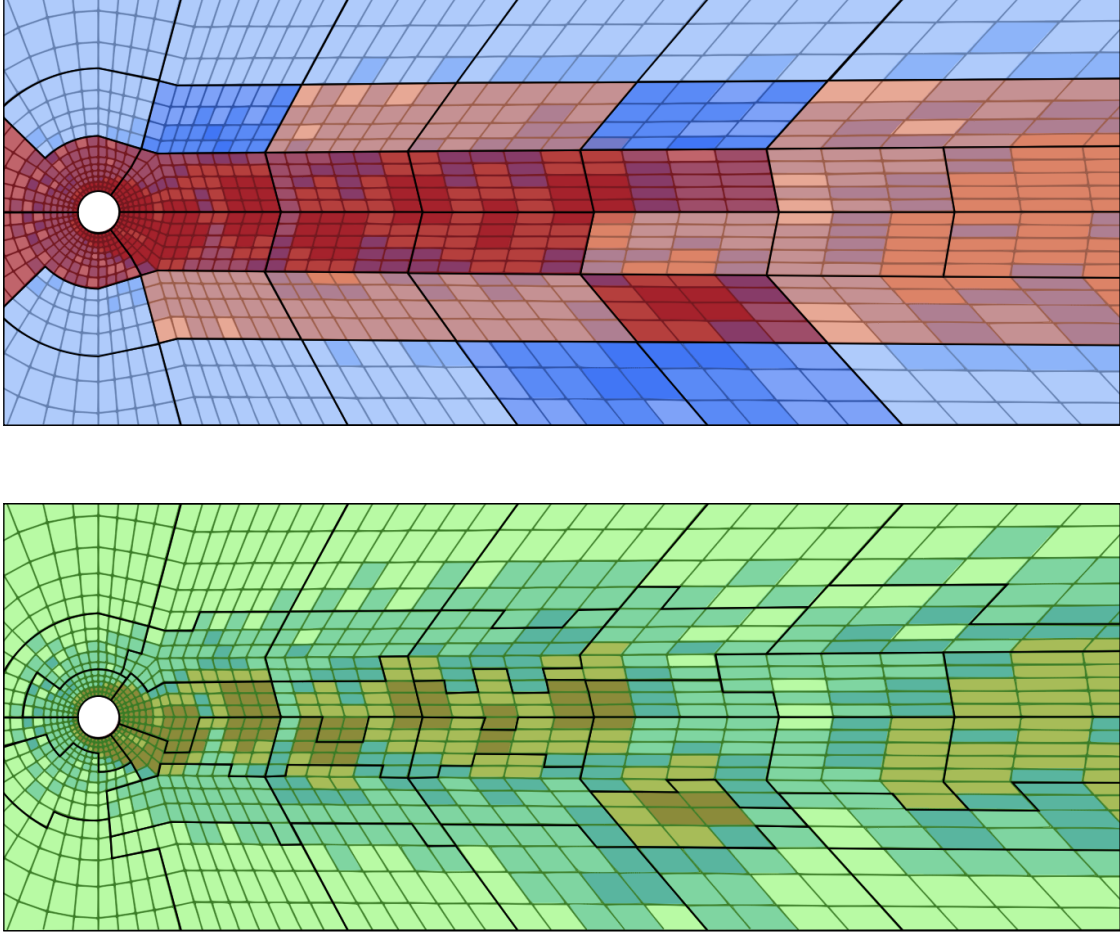


Figure 4.2. Schematic view of an unbalanced mesh partition (top) with highly ■, moderately ■, lightly ■, and barely ■ expensive regions along with a balanced mesh (bottom) with equal computational load distribution for a polynomial adaptive simulation. Black lines denote the partition boundaries.

where $N_{d,C}$ is the number of total edges between local elements contained by processor C . Some of the edges may be counted twice as there is a two-way adjacency for elements within a processor. However, for those elements located on the interface of two processors, the edges would be listed once as the adjacent element belongs to another processor. Furthermore, an array of size $N_{E,C} + 1$, `xadj` $[N_{E,C} + 1]$, is required for each processor to point to where each of the local elements adjacency lists begin and end, where $N_{E,C}$ is the number of local elements allocated to processor C . An array of size $N_P + 1$, `vtxdist` $[N_P + 1]$, where N_P is the number of processors used in the parallel simulation, indicates which elements are allocated to each processor. This array is the same for all processors. Finally, local element weights are stored

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Figure 4.3. Schematic view of a mesh with 16 elements partitioned over four processors.

in an array of size $N_{E,C}$, `vwgt`[$N_{E,C} + 1$], which is used as the balance factor. The output of the ParMETIS repartitioning routine is an array of size $N_{E,C}$, `part`[$N_{E,C} + 1$], which indicates the desired vector of the locally-stored elements. In the DLB algorithm, elements are transferred from over-loaded to under-loaded processors based on the `part`[$N_{E,C} + 1$] array. As an example, Figure 4.3 shows a mesh with 16 quadrilateral elements that is partitioned over 4 processors. The corresponding input arrays are reported in Table 4.1. The ParMETIS adaptive repartitioning is a multi-objective optimization problem and performs based on tuning parameters. A thorough explanation along with other set parameters can be found in ParMETIS manual [95].

4.3.3 Implementation

As mentioned earlier, there is a direct relation between the element weight and the element type, solution polynomial degree, number of dimensions, and number of conserved variables to be solved at each time step. In this study, the element weight W_E is defined for all element types with different degree polynomials, based on the required wall clock time to solve a single iteration on a single Intel Skylake core (2.4GHz, AVX512) of the Niagara, a Compute Canada cluster. The table of weights relative to the weight of a $K = 1$ triangular element is included in Appendix A for different element types and degree polynomials. As shown in

Table 4.1. Arrays of `xadj`, `adjncy`, and `vtxdist` for the schematic mesh with 16 elements shown in Figure 4.3.

Processor 0	<code>xadj</code>	0	2	5	8	12							
	<code>adjncy</code>	1	2	0	3	4	0	3	8	1	2	6	9
	<code>vtxdist</code>	0	4	8	12								
Processor 1	<code>xadj</code>	0	3	5	9	12							
	<code>adjncy</code>	1	5	6	4	7	3	4	7	12	5	6	13
	<code>vtxdist</code>	0	4	8	12								
Processor 2	<code>xadj</code>	0	3	7	9	12							
	<code>adjncy</code>	2	9	10	3	8	11	12	8	11	9	10	12
	<code>vtxdist</code>	0	4	8	12								
Processor 3	<code>xadj</code>	0	4	7	10	12							
	<code>adjncy</code>	6	9	13	14	7	12	15	11	12	15	13	14
	<code>vtxdist</code>	0	4	8	12								

Part A - Define the initial parameters;

Define the `vtxdist` $[N_P + 1]$ array;

for $c \in [1, 2, \dots, N_P]$ **do**

 Find the local number of elements $N_{E,C}$;

for $i \in [1, 2, \dots, N_{E,C}]$ **do**

 Find the total number of edges $N_{d,E,C}$;

end

 Define the `xadj` $[N_{E,C} + 1]$ pointer using $N_{d,E,C}$;

 Define the `adjncy` array size, $N_{d,C}$, using $N_{d,E,C}$;

 Define the `adjncy` $[N_{d,C}]$ pointer;

 Write element weights to `vwgt` $[N_{E,C}]$ array;

end

Part B - Call Load balancing;

Call ParMETIS and pass the parameters;

Read the array of locally-stored elements `part` $[N_{E,C} + 1]$;

Part C - Transfer elements;

Flag the elements to be transferred;

Transfer information of flagged elements to the destination processor using

`part` $[N_{E,C} + 1]$;

Create the transferred elements on the destination processors;

Delete the transferred elements on the original processors;

Algorithm 2: Dynamic load balancing algorithm for polynomial adaptation.

Algorithm 2, the algorithm starts by defining arrays of `vtxdist` $[N_P + 1]$, `xadj` $[N_{E,C} + 1]$, `adjncy` $[N_{d,C}]$, and `vwgt` $[N_{E,C}]$ for each processor. It then passes the input to the ParMETIS AdaptiveRepart routine to get the `part` $[N_{E,C} + 1]$ array, which indicates the optimised distribution of elements within processors based on the predefined weights to achieve a

balanced domain. Finally, the algorithm transfers elements from over-loaded to under-loaded processors based on the `part`[$N_{E,C} + 1$] array.

The following chapter verifies the ALE and DLB implementations by solving problems for which an analytical solution exists, and validates them by solving different known two-dimensional test cases for which we have reference results. This is to ensure the discrete model and the adaptation algorithm are accurate if applied to larger-scale turbulent test cases we are ultimately trying to solve.

Chapter 5

Verification and Validation Test Cases

The objective of the current chapter is to verify and then validate the FR scheme and HORUS to solve the problems we are interested in, when coupled with non-dimensional vorticity-based polynomial adaptation and DLB algorithms. Verification determines if the discretized system of equations, being the Euler and Navier-Stokes equations, are capable of approximating the continuous equations accurately. On the other hand, validation focuses on comparing numerical results of the verified discretized systems against experimental or other numerical results to ensure that they match physical observations.

The FR scheme and HORUS were verified and validated for numerous test cases as shown in reference [96]. In this chapter, we verify the ALE and DLB implementations by solving problems for which an analytical solution exists. Furthermore, we validate the p -adaptation algorithm by solving different known two-dimensional test cases and compare the numerical results against reference data to ensure the adaptive simulations will be accurate when applied to the larger-scale turbulent test cases we are ultimately trying to solve.

5.1 ALE Verification

Advection of an isentropic vortex in a deforming periodic domain using the Euler equations is considered for verification of the ALE implementation. This problem has an exact analytical solution, which is simply the propagation of the initial isentropic vortex with the mean flow.

The initial flow field is specified similar to previous studies as [97, 98]

$$\begin{aligned}
\rho &= \left[1 - \frac{S_v^2 M^2 (\gamma - 1) e^{2f}}{8\pi^2} \right]^{\frac{1}{\gamma-1}}, \\
v_x &= 1 + \frac{S_v y e^f}{2\pi R_c}, \\
v_y &= -\frac{S_v x e^f}{2\pi R_c}, \\
p &= \frac{\rho^\gamma}{\gamma M^2},
\end{aligned} \tag{5.1}$$

where $f = (1 - x^2 - y^2)/2R_c^2$, $S_v = 13.5$ is the strength of the vortex, $R_c = 1.5$ is the characteristic vortex radius, and $\gamma = 1.4$. This allows the vortex to advect at unit velocity vertically in the domain. A 20×20 two-dimensional domain centred at the origin was used with periodic boundary conditions on all four edges using 5×5 , 10×10 , 20×20 , and 40×40 element meshes. Initially isotropic, this mesh is perturbed using a prescribed mesh deformation $\mathbf{x}_g = [x_g, y_g]$ where

$$\begin{aligned}
x_g &= x_0 + \frac{3}{2} \sin\left(\frac{\pi t}{10}\right) \sin\left(\frac{\pi y_0}{10}\right), \\
y_g &= y_0 + \frac{3}{2} \sin\left(\frac{\pi t}{10}\right) \sin\left(\frac{\pi x_0}{10}\right),
\end{aligned} \tag{5.2}$$

and $\mathbf{x}_0 = [x_0, y_0]$ is the initial mesh position. This significantly deforms the original mesh up to $t = 5$, reverses this deformation at $t = 15$, and then returns it to its original shape at the final solution time $t = 20$. Solution polynomials of degree $K = 1$ to $K = 5$ were considered for each level of mesh resolution. The solution and flux points were located at Gauss points, Rusanov fluxes were used at the interfaces between elements, and the classical RK_{4,4} scheme was used in time. To evaluate the accuracy of each scheme, the L_2 norm of the density error was evaluated at the end of each simulation in a 4×4 region at the center of the domain. This area normalized L_2 error is defined as

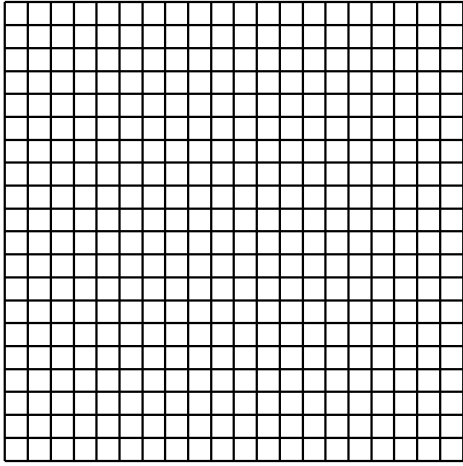
$$\sigma = \sqrt{\frac{\int_{-2}^2 \int_{-2}^2 (\rho_\delta(\mathbf{x}) - \rho_e(\mathbf{x}))^2 d\mathbf{x}}{8}}, \tag{5.3}$$

where $\rho_\delta(\mathbf{x})$ is the final numerical solution and $\rho_e(\mathbf{x})$ is the exact analytical solution, which is identical to the initial condition. To get a good approximation of the true L_2 error we used an 81-point Gaussian quadrature rule within each element.

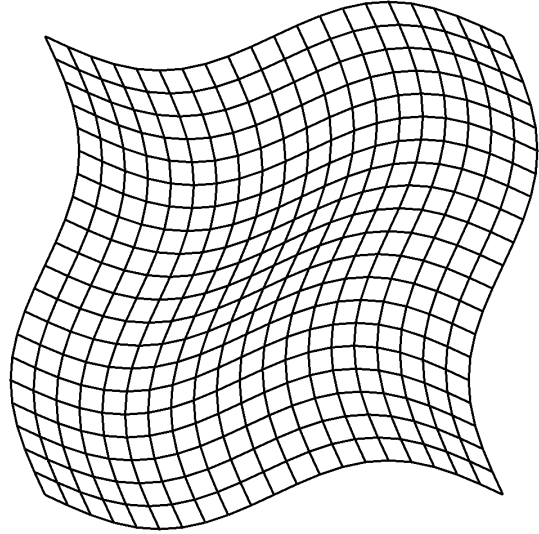
The shape of the deformed mesh is shown in Figure 5.1 for the initial condition, time of maximum mesh deformation, and final solution time for a $K = 5$ polynomial degree on the 20×20 element mesh. Corresponding contours of density are shown in Figure 5.2 for the same simulation. This shows that the mesh undergoes a significant deformation, reverses this deformation, and then returns to its original shape by the end of the simulation. There is no difference qualitatively between the initial and final solutions for this polynomial degree and, even at the time of maximum deformation, the solution remains smooth and similar to the initial condition. Table 5.1 shows the measured error for each polynomial degree and mesh, with and without the mesh deformation applied. All of the schemes achieve their designed orders of accuracy on the deforming mesh, with only a small penalty incurred in terms of accuracy relative to the static mesh cases. This is also demonstrated in Figure 5.3, which plots the convergence for each polynomial degree and element mesh on the static and deforming meshes. We see that all of the deforming mesh cases achieve similar error levels to those in the static mesh cases, obtain their designed high-order accuracy, and demonstrate the benefits of using a higher-order scheme for a given number of elements.

5.2 DLB Verification

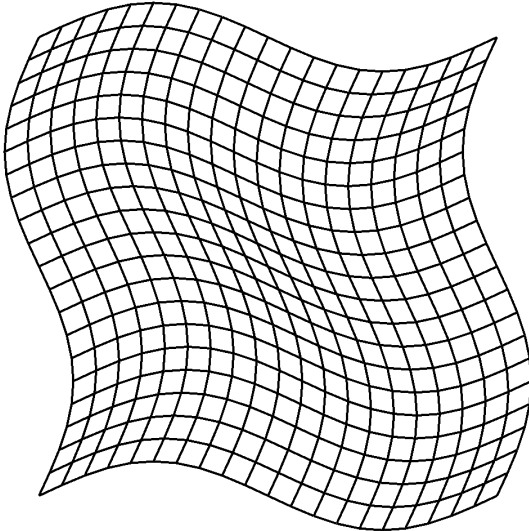
Advection of an isentropic vortex in a periodic domain using the Euler equations is reconsidered for verification of the DLB implementation. The initial flow field is specified, similar to the previous section. We first consider a 20×20 two-dimensional domain centred at the origin with periodic boundary conditions on all four faces using a 40×40 grid mesh. The mesh is initially partitioned poorly over 4 and 32 processors to explore the utility of the dynamic load balancing algorithm with and without weight allocation for elements with different degree polynomials. The solution and flux points were located at Gauss points, Rusanov fluxes were used at the interfaces between elements, and the classical RK_{4,4} scheme was used in time. Simulations were carried out on Intel Skylake cores (2.4 GHz, AVX512).



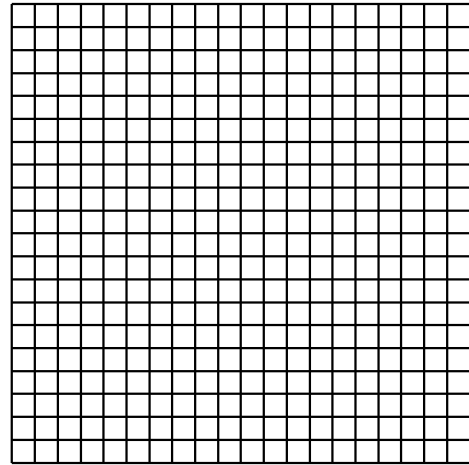
(a) Mesh at $t = 0$.



(b) Mesh at $t = 5$.

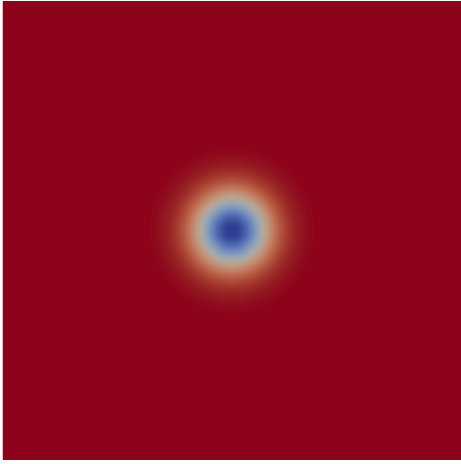


(c) Mesh at $t = 15$.

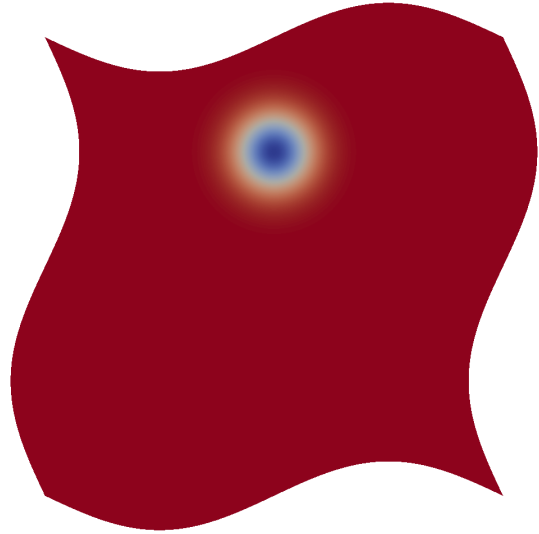


(d) Mesh at $t = 20$.

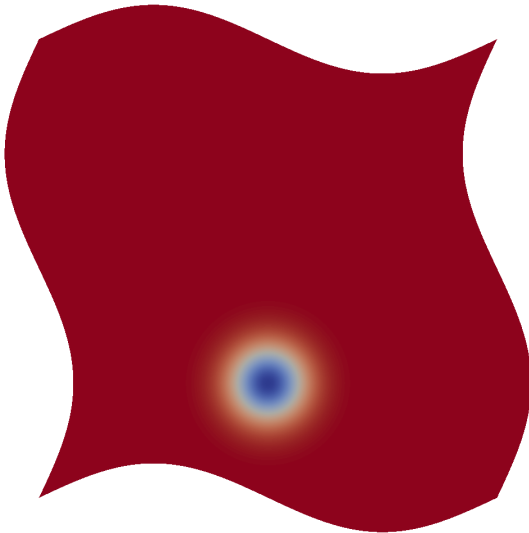
Figure 5.1. Mesh deformation for the isentropic vortex case on the 20×20 element mesh.



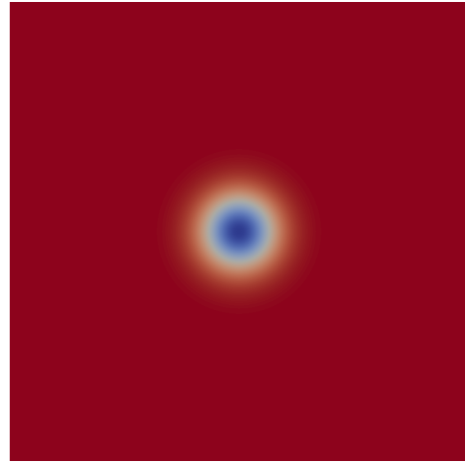
(a) Density contours at $t = 0$.



(b) Density contours at $t = 5$.



(c) Density contours at $t = 15$.



(d) Density contours at $t = 20$.

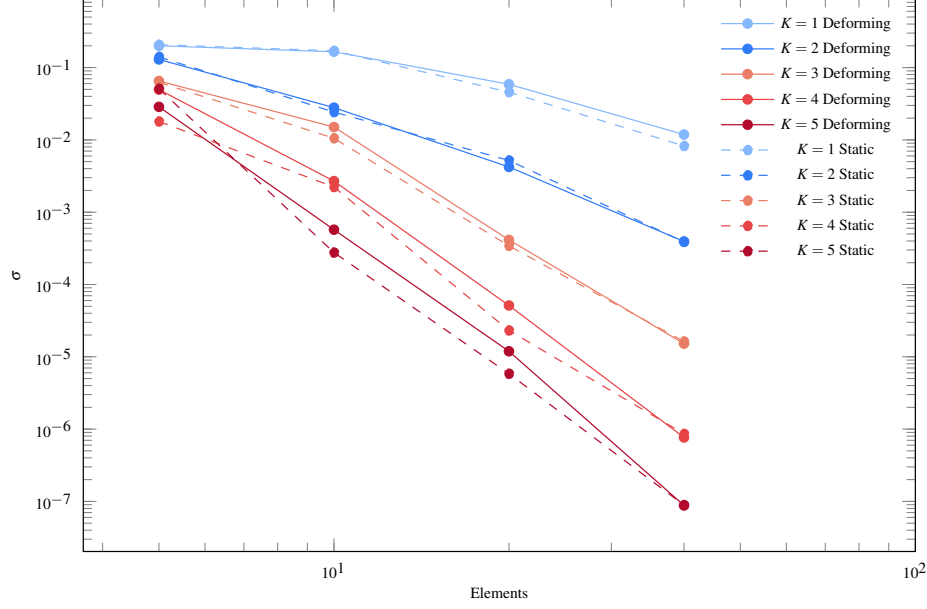
Figure 5.2. Density contours for the isentropic vortex case on the 20×20 element mesh with solution polynomials of degree $K = 5$.

Table 5.1. Density error for the deforming and static isentropic vortex cases.

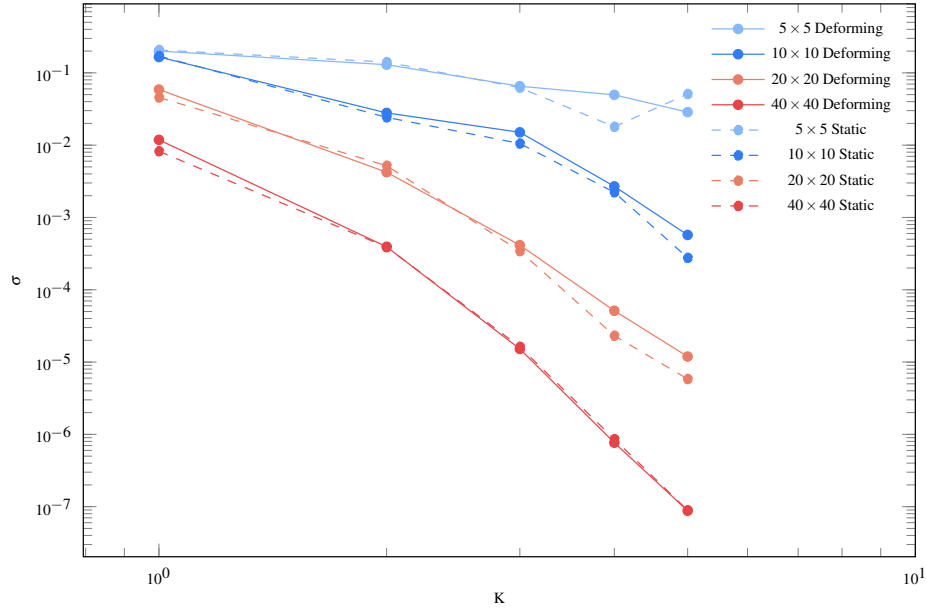
Scheme	Mesh	Deforming σ	Order	Static σ	Order
$k = 1$	5×5	2.001×10^{-1}	-	2.074×10^{-1}	-
	10×10	1.656×10^{-1}	0.27	1.708×10^{-1}	0.27
	20×20	5.880×10^{-2}	1.49	4.574×10^{-2}	1.90
	40×40	1.185×10^{-2}	2.31	8.244×10^{-3}	2.47
$k = 2$	5×5	1.297×10^{-1}	-	1.408×10^{-1}	-
	10×10	2.794×10^{-2}	2.21	2.426×10^{-2}	2.53
	20×20	4.225×10^{-3}	2.72	5.198×10^{-3}	2.22
	40×40	3.931×10^{-4}	3.42	3.885×10^{-4}	3.74
$k = 3$	5×5	6.533×10^{-2}	-	6.243×10^{-2}	-
	10×10	1.505×10^{-2}	2.11	1.054×10^{-2}	2.56
	20×20	4.141×10^{-4}	5.18	3.425×10^{-4}	4.94
	40×40	1.521×10^{-5}	4.76	1.629×10^{-5}	4.39
$k = 4$	5×5	4.961×10^{-2}	-	1.797×10^{-2}	-
	10×10	2.692×10^{-3}	4.20	2.228×10^{-3}	3.01
	20×20	5.125×10^{-5}	5.71	2.314×10^{-5}	6.58
	40×40	7.657×10^{-7}	6.06	8.592×10^{-7}	4.75
$k = 5$	5×5	2.862×10^{-2}	-	5.099×10^{-2}	-
	10×10	5.729×10^{-4}	5.64	2.765×10^{-4}	7.52
	20×20	1.192×10^{-5}	5.58	5.833×10^{-6}	5.56
	40×40	8.801×10^{-8}	7.08	8.892×10^{-8}	6.03

The mesh was initially partitioned over 4 and 32 processors, MPI was used for parallel communication [99].

A total of three adaptive simulations were carried out on each mesh to verify the utility of the dynamic load-balanced polynomial adaptation algorithm, including a simulation with static load balancing, where the dynamic load balancing routine is not called throughout the simulation, and two simulations with dynamic load balancing with and without assigning a computational burden to elements with different degree polynomials. Table 5.2 shows different load balancing schemes used in this verification study. As mentioned in Chapter 4, the computational burden varies based on the element type, solution polynomial degree, number of dimensions, and number of conserved variables. The compute time for a single triangular element with the polynomial degree of $K = 1$ is used as the reference to define the element weight in the algorithm for all element types and different degree polynomials. The adaptation parameters K_m , ϵ , and $\vartheta_{1:5}$ are set to 5, 6.25×10^{-2} , and $[0.25, 0.25, 1, 4, 16]$ respectively. The adaptation routine is called every 20 iterations, and the DLB routine is



(a) Convergence plot for each polynomial degree.



(b) Convergence plot for each element mesh.

Figure 5.3. Convergence plot of the density error for the deforming and static isentropic vortex cases.

called every 20 iterations for the LB_2 and LB_3 schemes, refer to Table 5.2. Contours of density, polynomial distribution, and the corresponding repartitioned mesh are shown in Figures 5.4 for $t = 0$, $t = 5$, $t = 10$, $t = 15$, and $t = 20$ for different load balancing schemes for a mesh partitioned over 4 processors. Comparing these results confirms that the weighted

LB_3 load balancing scheme is capable of repartitioning the mesh according to the polynomial distribution. The resulting mesh remains balanced for the entire simulation as the mesh is repartitioned based on the computation loads allocated to processors. Furthermore, the LB_2 scheme results in an initially balanced mesh, partitioned based on the total number of elements, which remains constant throughout the simulation. Although initially balanced, this scheme is not capable of retaining the domain balanced as the simulation evolves. The LB_1 scheme is the most unbalanced scheme as no repartitioning is performed.

We compare the performance of load balancing schemes in terms of averaged weight distribution $W_C = \left(\sum_1^{N_{itr}} \sum_{j=1}^{N_{E,C}} W_{E,j} \right) / N_{itr}$, which is the average load assigned to each core throughout the simulation based on the element type and degree polynomials, weight fluctuation ratio W_f which is defined as the ratio of minimum weight to the maximum weight assigned to processors at every iteration averaged throughout the entire simulation, averaged maximum computational burden $\bar{W}_{C,max}$ which is defined as the maximum computation load on a core in each iteration averaged throughout the simulation, and simulation time per iteration T_{itr} in millisecond defined as $T_{itr} = 1000T_w/N_{itr}$, where T_w is the wall clock time, and N_{itr} is the number of iterations. Table 5.3 shows the measured values of W_C , W_f , $\bar{W}_{C,max}$, and T_{itr} for each mesh and load balancing scheme. These results show that the LB_3 scheme results in the most balanced computation as the value of the weight fluctuation ratio is about unity, which simply means almost an equal computational load is allocated to each processor. Furthermore, we observe that the LB_3 scheme is the fastest. This is also demonstrated in Figure 5.5, which plots the reported values for both meshes. Finally, a set of parallel simulations were carried out on the Compute Canada Niagara cluster with different numbers of cores to compare the speed-up and scalability of the LB_2 and LB_3 load balancing schemes. The same 20×20 two-dimensional domain using a 40×40 grid mesh was considered. Table 5.4 shows the speed-up factor of the parallel algorithm for different numbers of cores for LB_2 and LB_3 DLB schemes, where C is the number of cores, $S_P = T_S/T_C$ is the parallel speed-up based on the serial simulation on a single core, $E_s = S_P/C$ is the efficiency, T_S is the compute time of a serial simulation, and T_C is the compute time of a parallel adaptive simulation on C cores. These results verify the scalability of the algorithm and illustrate that the LB_3 is faster than the LB_2 scheme. This is also demonstrated in Figure 5.6, which plots

Table 5.2. Load balancing schemes employed for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$.

Scheme	Load Balancing Type	Weight Assigned	Repartitioning Call (itr)
LB_1	Static	-	-
LB_2	Dynamic	No	20
LB_3	Dynamic	Yes	20

Table 5.3. Numerical values of $\bar{W}_{C,max}$, W_f , and T_{itr} for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over 4 and 32 processors.

Number of Cores	Scheme	$\bar{W}_{C,max}$	W_f	T_{itr} [ms]
4	LB_1	1.912×10^4	8.397×10^{-2}	6.865×10^1
	LB_2	1.126×10^4	2.944×10^{-1}	4.090×10^1
	LB_3	6.123×10^3	9.677×10^{-1}	3.091×10^1
32	LB_1	1.915×10^4	8.387×10^{-3}	6.811×10^1
	LB_2	2.295×10^3	1.733×10^{-1}	1.224×10^1
	LB_3	7.778×10^2	9.303×10^{-1}	9.464×10^0

the compute time of each polynomial degree for different numbers of cores.

Table 5.4. Scalability metrics of the LB_2 and LB_3 schemes for the isentropic vortex case on the two-dimensional 40×40 element mesh.

Scheme	Number of Cores	T_{itr} [ms]	S_P
LB_2	1	3.668×10^2	1.000
	2	2.082×10^2	1.762
	4	1.401×10^2	2.618
	8	1.237×10^2	2.965
	16	6.789×10^1	5.403
	32	3.976×10^1	9.226
	64	2.336×10^1	15.704
LB_3	1	3.666×10^2	1.000
	2	1.965×10^2	1.866
	4	1.197×10^2	3.064
	8	6.821×10^1	5.374
	16	3.822×10^1	9.591
	32	2.270×10^1	16.149
	64	1.699×10^1	21.576

To further verify the DLB algorithm when applied to a relatively larger three-dimensional domain and compare the efficiency of the algorithm, a $100 \times 100 \times 1$ three-dimensional domain centred at the origin was used with periodic boundary conditions on all six faces

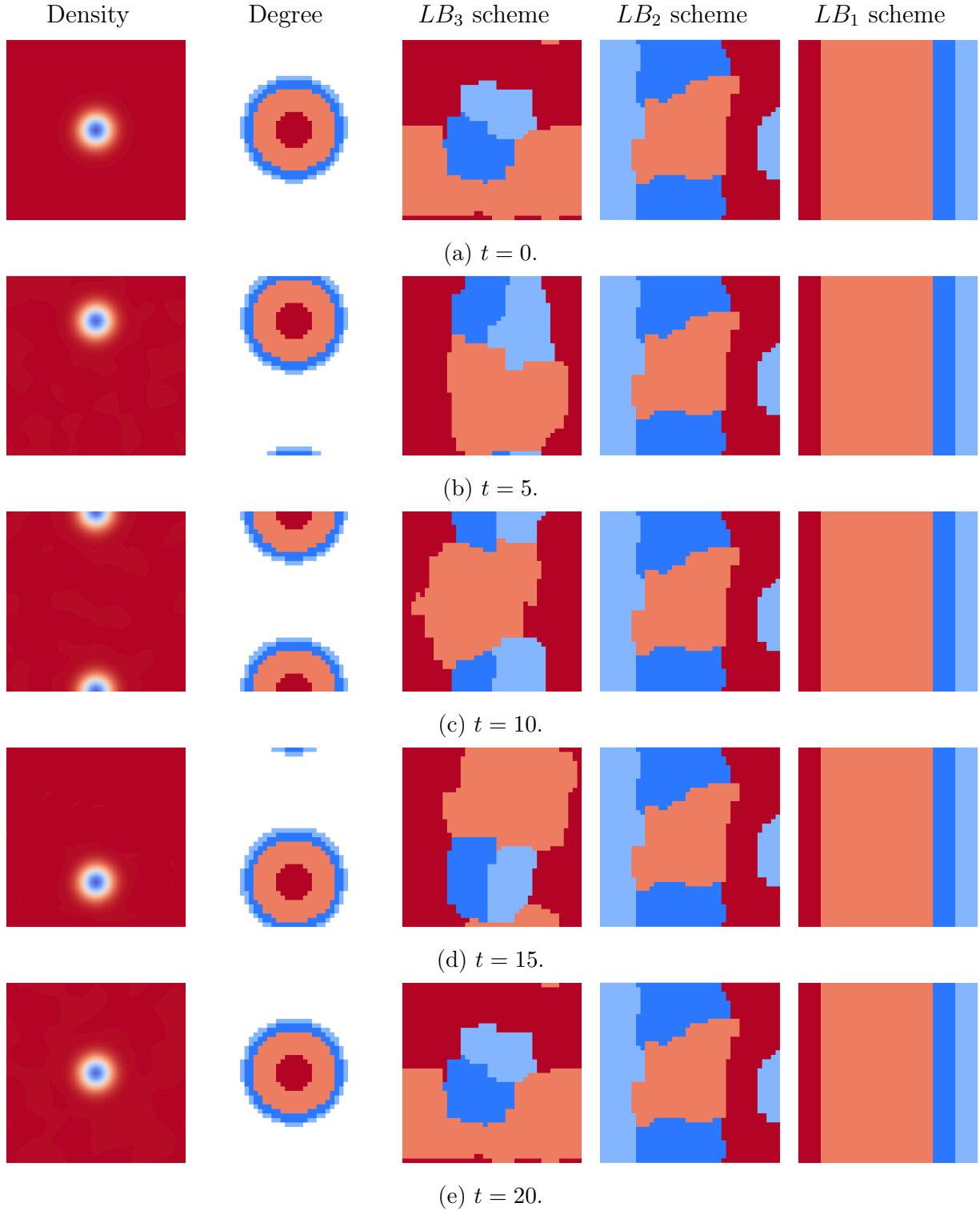
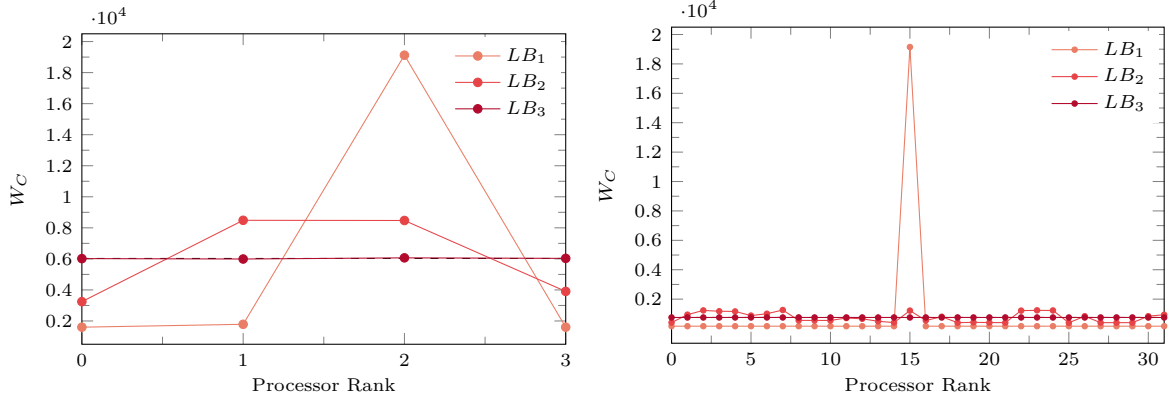


Figure 5.4. Contours of density, polynomial distribution, and the corresponding repartitioned mesh with different load balancing schemes at several time instants for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over 4 processors.



(a) For the mesh partitioned over 4 cores. (b) For the mesh partitioned over 32 cores.

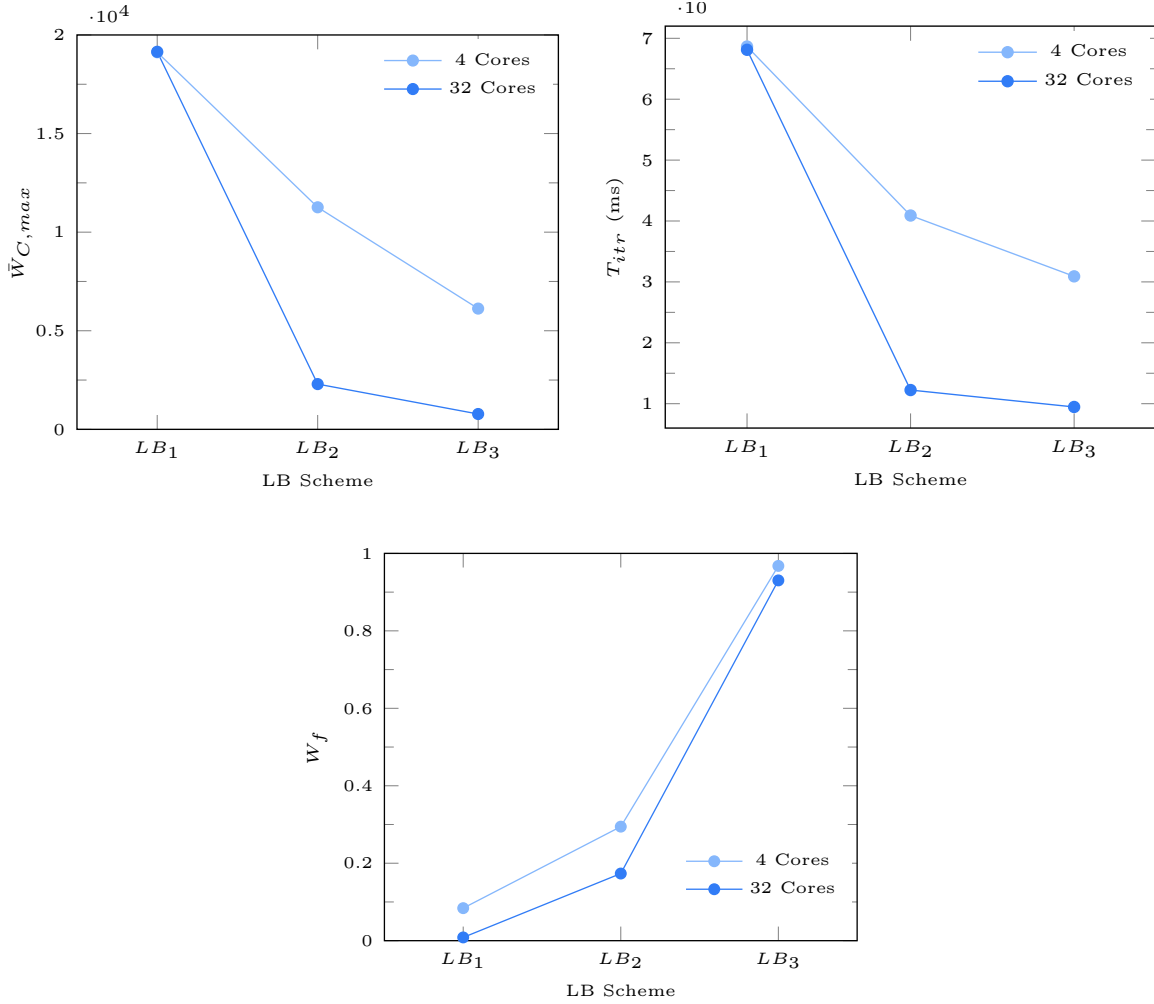


Figure 5.5. Plots of averaged weight distribution W_C , averaged maximum computational burden $\bar{W}_{C,max}$, simulation time per iteration T_{itr} , and weight fluctuation ratio W_f for the isotropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over 4 and 32 processors.

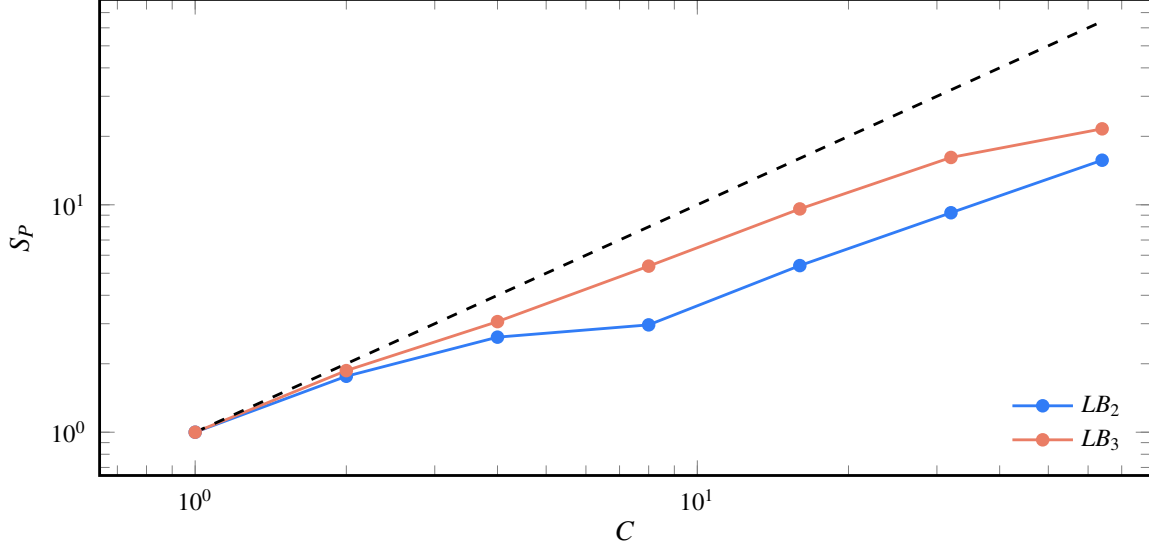


Figure 5.6. Plot of the speed-up of the dynamic load-balanced adaptive algorithm for the isentropic vortex case on the two-dimensional 40×40 element mesh with adaptive solution polynomial of degree $K = 1 - 5$ partitioned over different numbers of processors. The dashed line represents the ideal efficiency of one.

using 50×50 , 100×100 , 200×200 , and 400×400 element meshes with three elements in the z-direction for all meshes. Uniform polynomials of degree $K = 1$ to $K = 5$ and a five-level adaptive polynomials of degree $K = 1 - 5$ were considered for each level of mesh resolution. The solution and flux points were located at Gauss points, Rusanov fluxes were used at the interfaces between elements, and the classical $RK_{4,4}$ scheme was used in time. To validate the performance of the dynamic load-balanced polynomial adaptation algorithm, a set of parallel simulations were carried out on two nodes of Niagara, a Compute Canada cluster, for different polynomial degrees and mesh resolutions. Each node consists of 2 sockets with 20 Intel Skylake cores (2.4 GHz, AVX512), for a total of 40 cores per node, and a total of 202 GB of RAM. The compute time is calculated for each simulation. The adaptation parameters K_m , ϵ , and $\vartheta_{1:5}$ are set to 5, 2.5×10^{-1} , and $[0.25, 0.25, 1, 4, 16]$ respectively. The adaptation and the DLB routines are called every 100 and 1000 iterations, respectively.

20×20 centre-portions of different mesh resolutions are shown in Figure 5.7. Contours of density, polynomial distribution, and the corresponding repartitioned mesh are shown in Figures 5.8 for the initial condition, halfway through the simulation, and the final solution for an adaptive $K = 1 - 5$ polynomial degree on the 400×400 element mesh. This confirms that

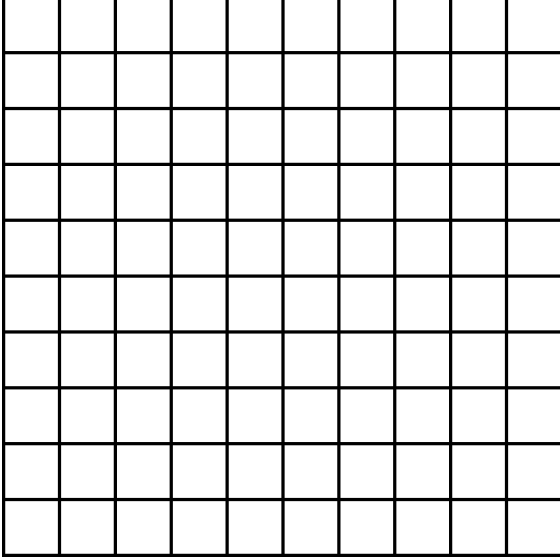
elements with large vorticity magnitudes relative to the effective mesh resolution are adapted to higher-degree polynomials. At the initial condition, the mesh is partitioned equally to allocate almost the same number of elements to each processor. Yet, upon calling the load balancing routine, the mesh is repartitioned to allocate almost the same computational load to each processor. Hence, processors computing the high-order elements will have fewer elements. There is no difference qualitatively between the initial and final solutions for the adaptive polynomial degree, and the solution remains smooth and similar to the initial condition. Table 5.5 shows the measured error, average total number of degrees of freedom \overline{DOF} , the compute time for each polynomial degree and mesh, and the achieved speed-up factor of the adaptive simulations, where T_{itr} is the compute time of one iteration of a parallel simulation on two nodes of the Niagara cluster, $S_A = T_U/T_A$ is the speed-up factor based on the uniform $K = 5$ simulation with the same mesh resolution, T_A is the compute time of an adaptive $K = 1 - 5$ simulation featured with DLB, and T_U is the compute time of a uniform $K = 5$ simulation. The adaptive simulation is faster for all cases compared to the uniform $K = 5$ simulations. Furthermore, we observe that the speed-up gets more significant as the mesh gets finer. This is also demonstrated in Figure 5.9, which plots the compute time of each polynomial degree for different meshes.

5.3 Validation: 2D Circular Cylinder

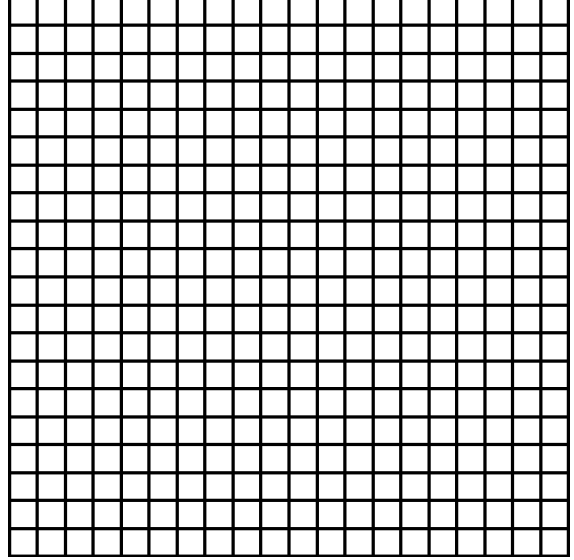
As a validation case for the polynomial adaptation algorithm when applied to the Navier-Stokes equations, we consider unsteady laminar flow over a two-dimensional circular cylinder. Several previous experimental and numerical studies have focused on flow over a circular cylinder, both due to its simple experimental setup and its engineering significance, including the prediction of aerodynamics forces, namely lift and drag, and the study of its vortex shedding regimes. Its characteristics are known to be highly dependent on the Reynolds number Re , defined as

$$Re = \frac{\rho U_\infty D}{\mu}, \quad (5.4)$$

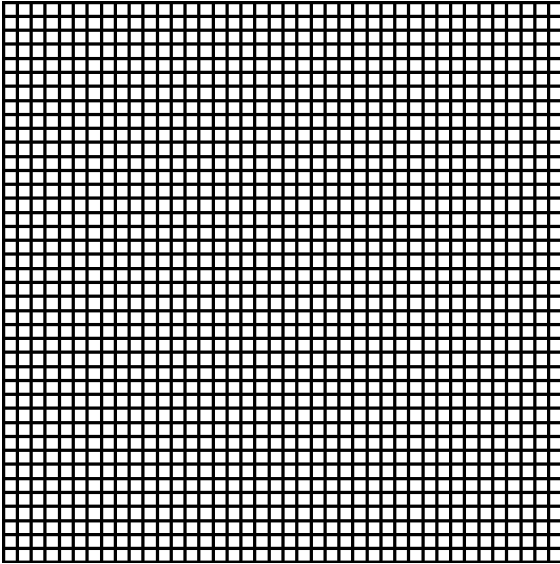
where U_∞ is the freestream velocity, ρ is the fluid density, D is the cylinder diameter, and μ is the fluid viscosity. At Re below about 49, the wake flow is in the laminar steady regime, which



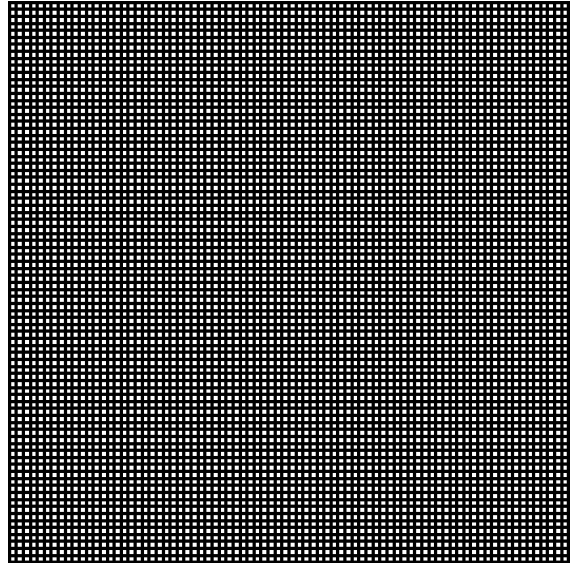
(a) 50×50 element mesh



(b) 100×100 element mesh



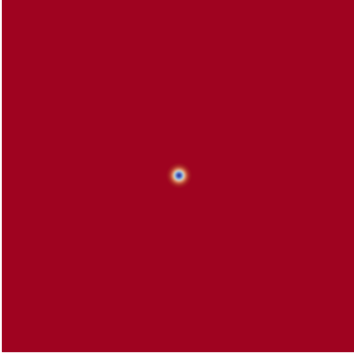
(c) 200×200 element mesh



(d) 400×400 element mesh

Figure 5.7. 20×20 centre-portion of the isentropic vortex element meshes.

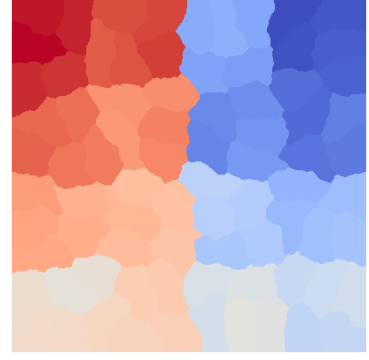
is comprised of symmetric vortices. As the Re increases, at Re above 49 and below about 170, vortices start to shed, yet the wake flow is still laminar. In fact, the wake structure remains 2D for Re smaller than about 170. Hence, in this section, flow over a 2D circular cylinder at $Re = 150$, which sits in the laminar vortex shedding regime identified by Williamson [100], is studied to validate the effectiveness of the vorticity-based polynomial adaptation technique. Inoue and Hatakeyama [101] studied the pressure waves generated by the vortex shedding



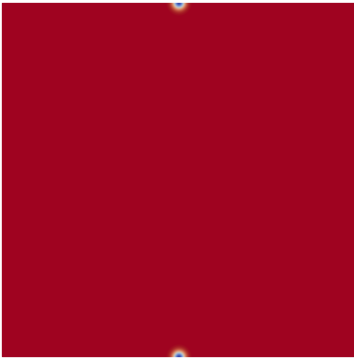
(a) Density at $t = 0$



(b) Degree at $t = 0$



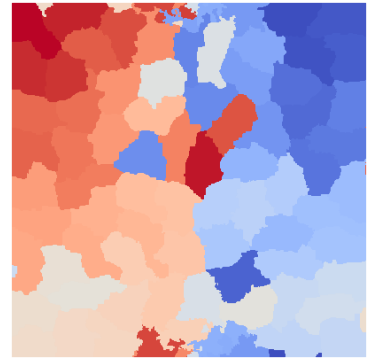
(c) Mesh partition at $t = 0$



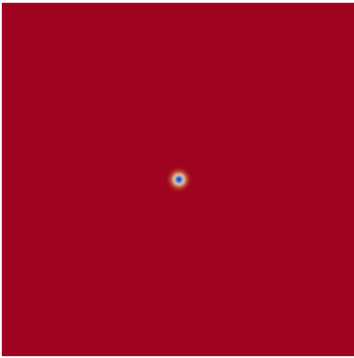
(d) Density at $t = 50$



(e) Degree at $t = 50$



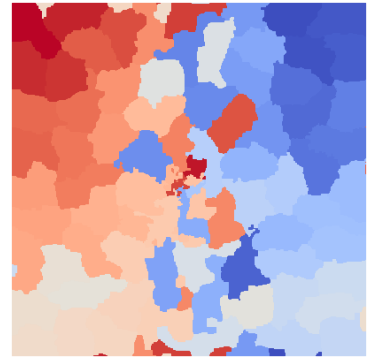
(f) Mesh partition at $t = 50$



(g) Density at $t = 100$



(h) Degree at $t = 100$



(i) Mesh partition at $t = 100$

Figure 5.8. Density contours, polynomial degree distributions, and mesh partitions for the isentropic vortex case on the three-dimensional $400 \times 400 \times 3$ element mesh with adaptive solution polynomial of degree $K = 1 - 5$.

from a cylinder surface in a flow at a range of low Mach numbers. Cagnone et al. [62] presented a polynomial-adaptive Lifting Collocation Penalty (LCP) formulation for the compressible Navier–Stokes equations and studied the effectiveness of their p -adaptive strategy when

Table 5.5. Density error, mean number of degrees of freedom, simulation time per iteration, and speed-up factor for the isentropic vortex case on the three-dimensional $400 \times 400 \times 3$ element mesh.

Scheme	Load Balancing	Mesh	Error σ	\overline{DOF}	T_{itr} [ms]	S_A
$K = 1$	Static	50×50	2.269×10^{-1}	6.000×10^4	0.971×10^1	-
		100×100	1.307×10^{-1}	2.400×10^5	4.843×10^1	
		200×200	2.817×10^{-2}	9.600×10^5	2.078×10^2	
		400×400	4.894×10^{-3}	3.840×10^6	8.300×10^2	
$K = 2$	Static	50×50	7.348×10^{-2}	2.025×10^5	3.479×10^1	-
		100×100	1.815×10^{-2}	8.100×10^5	1.413×10^2	
		200×200	8.025×10^{-4}	3.240×10^6	5.661×10^2	
		400×400	6.853×10^{-5}	1.296×10^7	2.242×10^3	
$K = 3$	Static	50×50	4.950×10^{-2}	4.800×10^5	7.959×10^1	-
		100×100	1.646×10^{-3}	1.920×10^6	3.125×10^2	
		200×200	3.348×10^{-5}	7.680×10^6	1.236×10^3	
		400×400	5.911×10^{-7}	3.072×10^7	4.862×10^3	
$K = 4$	Static	50×50	6.924×10^{-3}	9.375×10^5	1.676×10^2	-
		100×100	5.098×10^{-5}	3.750×10^6	6.439×10^2	
		200×200	1.617×10^{-6}	1.500×10^7	2.535×10^3	
		400×400	4.420×10^{-8}	6.000×10^7	1.012×10^4	
$K = 5$	Static	50×50	7.896×10^{-4}	1.620×10^6	3.274×10^2	-
		100×100	1.557×10^{-5}	6.480×10^6	1.283×10^3	
		200×200	1.643×10^{-7}	2.592×10^7	5.061×10^3	
		400×400	1.876×10^{-9}	1.037×10^8	2.012×10^4	
$K = 1 - 5$	Static	50×50	5.895×10^{-3}	7.634×10^4	1.561×10^2	2.09
		100×100	5.903×10^{-4}	$2,948 \times 10^5$	5.955×10^2	2.16
		200×200	4.278×10^{-5}	1.160×10^6	1.515×10^3	3.34
		400×400	3.859×10^{-6}	4.594×10^6	4.359×10^3	4.61
$K = 1 - 5$	Dynamic	50×50	5.895×10^{-3}	7.634×10^4	8.437×10^1	3.88
		100×100	5.903×10^{-4}	$2,948 \times 10^5$	2.078×10^2	6.18
		200×200	4.278×10^{-5}	1.160×10^6	6.019×10^2	8.41
		400×400	3.859×10^{-6}	4.594×10^6	1.983×10^3	10.14

applied to the unsteady laminar flow past a circular cylinder. Ching et al. [71] introduced an entropy residual polynomial adaptation indicator, explained in detail in Chapter 4, and evaluated its performance for viscous flow past a circular cylinder.

In this study, we demonstrate the application of our p -adaptation strategy for weakly-compressible flow over a circular cylinder. Two two-dimensional computational domains, with their origin located at the centre of the cylinder, one containing a total of 4342 quadrilateral elements (hereafter referred to as mesh 1), and the other containing a total of 8684 triangle

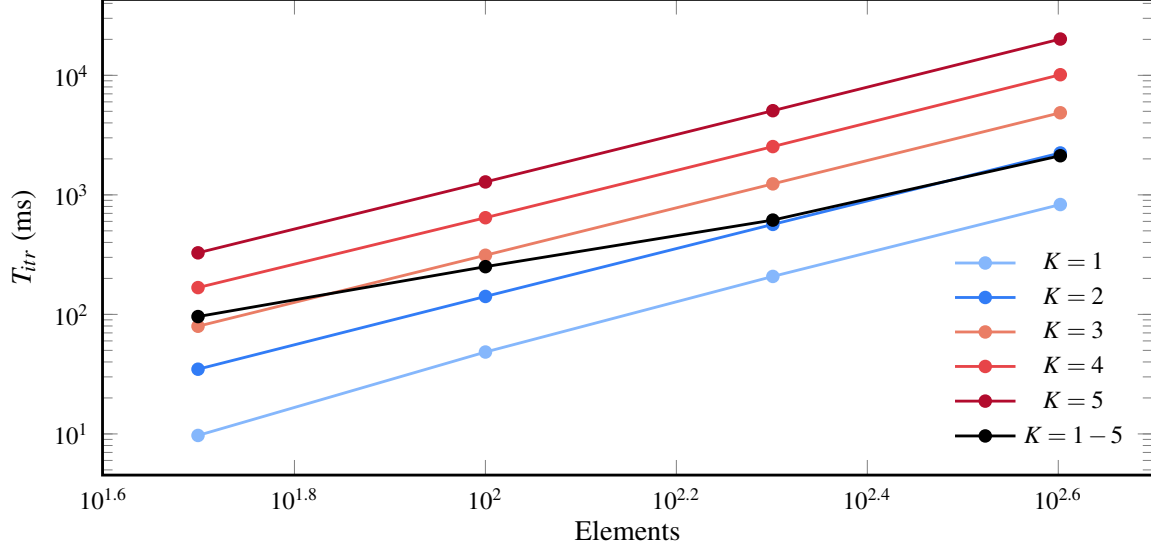


Figure 5.9. Plot of the compute time per iteration for the isentropic vortex cases on the three-dimensional 400×400 element mesh.

elements (hereafter referred to as mesh 2) were used with Riemann invariant boundary conditions at the far-field and a no-slip adiabatic wall boundary condition at the surface of the cylinder. The domains extend to $40D$ above, below, and upstream of the cylinder, $110D$ downstream as shown in Figure 5.10. Meshes are moderately refined in the near-cylinder and wake regions and use quadratically curved elements at the boundaries to match the cylinder geometry. Table 5.6 lists the computational meshes employed. The mesh is initially partitioned over 4 processors using METIS [102], and MPI is used for parallel communication [99].

Simulations are carried out at Mach number $M = 0.1$, for a total of $200t_c$, where $t_c = D/U_\infty$ is the time required for the flow to traverse one cylinder diameter. Time integration is carried out with the two-stage second order singly diagonally implicit Runge-Kutta scheme, SDIRK22, with implicit tolerance of 10^{-4} , and the non-dimensional time step is set to $\Delta t^* = \Delta t U_\infty / D = 1.0 \times 10^{-2}$. The Jacobian matrix is calculated every 10 iterations, and the adaptation routine is called every 10 iterations. A total of four simulations were carried out to verify the utility of the adaptation algorithm, including two uniform simulations on Mesh 1 with solution polynomials of degree $K = 1$ and $K = 3$, and two simulations with three-level adaptive polynomials $K = 1 - 3$ on both meshes. The adaptation parameters K_m , ϵ , and $\vartheta_{1:3}$ are set to 3, 2.5×10^{-1} , and $[0.25, 0.25, 1]$ respectively. Results

are compared in terms of mean drag coefficient \bar{C}_D , the peak fluctuation in the lift ΔC_L and drag ΔC_D coefficients, Strouhal number St , and the wall clock time T_w , where the lift and drag coefficients are defined in a general form as

$$\begin{aligned} C_L &= \frac{F_L}{qA_f}, \\ C_D &= \frac{F_D}{qA_f}, \end{aligned} \tag{5.5}$$

where A_f is the frontal area, which is equal to the length of the projected line normal to the flow, $q = \rho_\infty U_\infty^2/2$ is the dynamic pressure, and ρ_∞ is the freestream density. For this case study, the lift and drag coefficients reduce to

$$\begin{aligned} C_L &= \frac{2F_L}{\rho_\infty U_\infty^2 D}, \\ C_D &= \frac{2F_D}{\rho_\infty U_\infty^2 D}, \end{aligned} \tag{5.6}$$

and the Strouhal number is defined as

$$St = \frac{fD}{U_\infty}, \tag{5.7}$$

where f is the shedding frequency of the dominant vortices in the wake.

Figure 5.11 shows the polynomial degree distribution for the three-level adaptive simulations ($K = 1 - 3$) for both meshes. As mentioned earlier in Chapter 4, elements with large vorticity magnitudes relative to the effective mesh resolution are adapted to higher-degree polynomials. It is evident that most of the $K = 3$ elements are employed where the vorticity magnitude is maximal relative to the element size, which illustrates that high-order elements track vorticity behind the cylinder. $K = 2$ elements occur further downstream, out of the wake region, leaving the far-field region for $K = 1$ elements. This verifies that the algorithm successfully tracks the location of elements with large vorticity magnitude relative to their element size. Figure 5.12 shows contours of non-dimensional velocity magnitude at the time of a maximal lift for uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ for mesh 1, along with the adaptive $K = 1 - 3$ for mesh 2. While we observe large inter-element jumps

in the flow solution of the uniform $K = 1$ simulation, the flow solution is smooth for the uniform $K = 3$ and both adaptive $K = 1 - 3$ simulations. Figure 5.13 shows contours of non-dimensional spanwise vorticity at the time of maximal lift for uniform $K = 1$, uniform $K = 3$, along with the adaptive $K = 1 - 3$ for mesh 1, and adaptive $K = 1 - 3$ for mesh 2, where vorticity is defined as $\boldsymbol{\omega} = \nabla \times \mathbf{u}$. Comparing these results confirms that the three-level adaptive simulations qualitatively provide satisfactory resolution in the near-cylinder and the wake regions while requiring fewer degrees of freedom. To further evaluate the accuracy of the adaptive method, we turn to Table 5.7 which reports the numerical values of \bar{C}_D , ΔC_L , ΔC_D , St , and T_w for uniform simulations and the three-level adaptive simulations and compares them with results from Cagnone et al. [62] and Inoue and Hatakeyama [101]. These results show the adaptive simulations quantitatively agree the uniform $K = 3$ simulation within 0.27%, 35.38%, and 0.14% numerical error for mesh 1 and 0.241%, 43.46%, and 0.14% numerical error for mesh 2 in the ΔC_L , ΔC_D , and \bar{C}_D values respectively, while the adaptive simulations are 2.32 and 2.42 times faster for mesh 1 and 2, respectively. Figure 5.14 compares the lift and drag coefficient plots of the uniform and adaptive simulations, which shows excellent agreement with small oscillations observed in the C_D profile. Since the locations of higher-order solutions are changing as the vortices are shed in the near-cylinder and/or wake regions, the vorticity indicator continually refines and coarsens the elements in these regions. Due to viscous effects observed only at low Reynolds numbers, continuous change of higher-order solutions to lower orders induces aliasing errors and small oscillations in the C_D profile causing the error in the ΔC_D value. However, this was not found to affect the accuracy of \bar{C}_D , since the magnitude of the fluctuations is relatively small compared to the mean drag coefficient. Upon increasing the Reynolds number, these oscillations are reduced. Figure 5.16 illustrates the dependency of the C_D fluctuations on the Reynolds number for adaptive simulations on mesh 1, with the oscillations being reduced when the Reynolds number is increased from $Re = 150$ to $Re = 1000$. Furthermore, Figure 5.15 compares the lift and drag coefficient plots of adaptive simulations for different meshes. Although the results obtained from both meshes are in excellent agreement, we observe that mesh 1, which contains quadrilateral elements, is more accurate both qualitatively and quantitatively and has a better agreement with the reference data. This is in agreement with previously observed

Table 5.6. Computational meshes employed for the circular cylinder the circular cylinder with $Re = 150$ and $M = 0.1$.

Mesh Type	Number of Elements	Element Type	Polynomial Order
Mesh 1, $K = 1$	4342	Quadrilateral	1
Mesh 1, $K = 3$			2
Mesh 1, $K = 1 - 3$			1-3
Mesh 2, $K = 1 - 3$	8684	Triangular	1-3

Table 5.7. Numerical values of ΔC_L , ΔC_D , \bar{C}_D , St , T_w , and error percent for the circular cylinder with $Re = 150$ and $M = 0.1$.

Mesh	Degree	ΔC_L	ΔC_D	\bar{C}_D	St	$\varepsilon_{\Delta C_L}$	$\varepsilon_{\Delta C_D}$	$\varepsilon_{\bar{C}_D}$	T_w (s)
1	$K = 1$	0.4862	0.0223	1.3049	0.1751	6.03%	14.23%	1.55%	10,759
1	$K = 3$	0.5174	0.0260	1.3255	0.1839	-	-	-	112,409
1	$K = 1 - 3$	0.5160	0.0352	1.3236	0.1835	0.27%	35.38%	0.14%	48,528
2	$K = 1 - 3$	0.5153	0.0373	1.3237	0.1860	0.41%	43.46%	0.14%	32,821
Cagnone et al. [62]	$K = 4$	0.5166	0.0258	1.3246	0.1836	-			
Inoue et al. [101]	-	0.52	0.026	1.32	0.183	-			

spectral analyses where tensor-product elements display increased accuracy with respect to simplices [103, 104, 105].

5.4 Validation: SD 7003 Airfoil

To further validate the performance and accuracy of our p -adaptation method for the unsteady Navier-Stokes equations, we consider unsteady laminar flow over an SD 7003 airfoil at a Reynolds number $Re = 10,000$, where $Re = \rho U_\infty c / \mu$, c is the airfoil chord length, and μ is the constant dynamic viscosity. We use a Mach number of $M = 0.2$ and an angle of attack of $\alpha = 4^\circ$. This is a common test case to validate CFD solvers with high-order reference results available from Uranga et al. [106] using a DG scheme, Lopez-Morales et al. [107], and Vermeire [108] using the FR scheme.

At this Reynolds number, the flow is expected to be two-dimensional with negligible variation in the span-wise direction [106]. Hence, we used a two-dimensional computational domain, with its origin located at the leading edge containing a total of 8190 quadrilateral

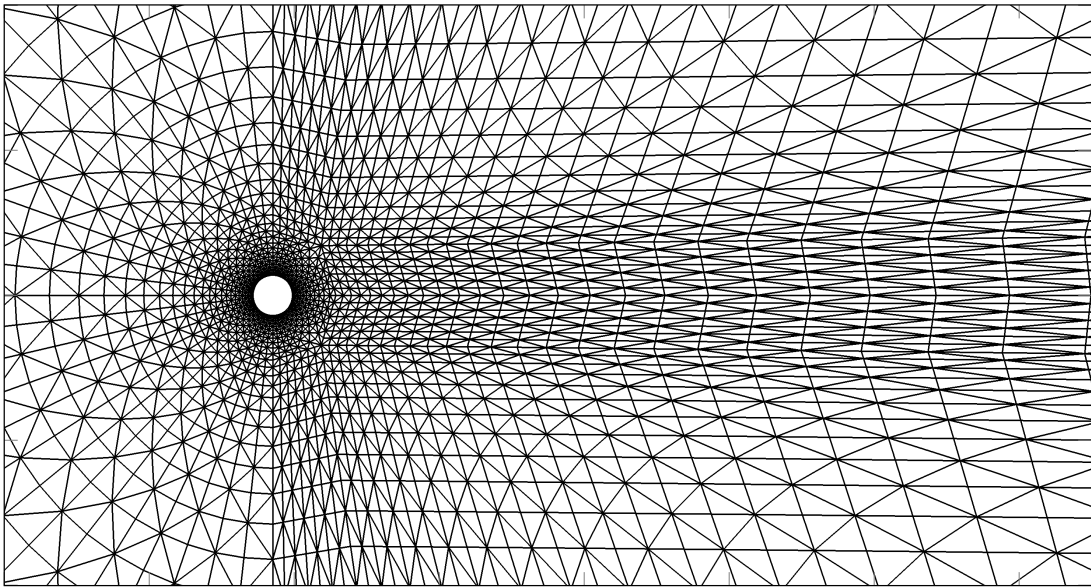
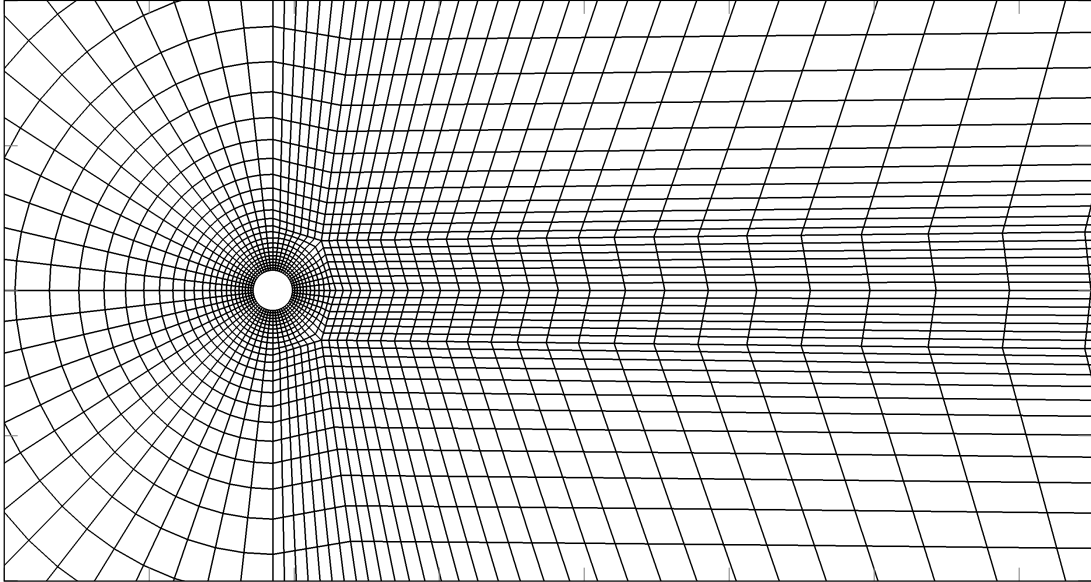


Figure 5.10. Circular cylinder quadrilateral structured mesh with 4342 elements (mesh 1), on the top, and circular cylinder triangular structured mesh with 8684 elements (mesh 2), on the bottom.

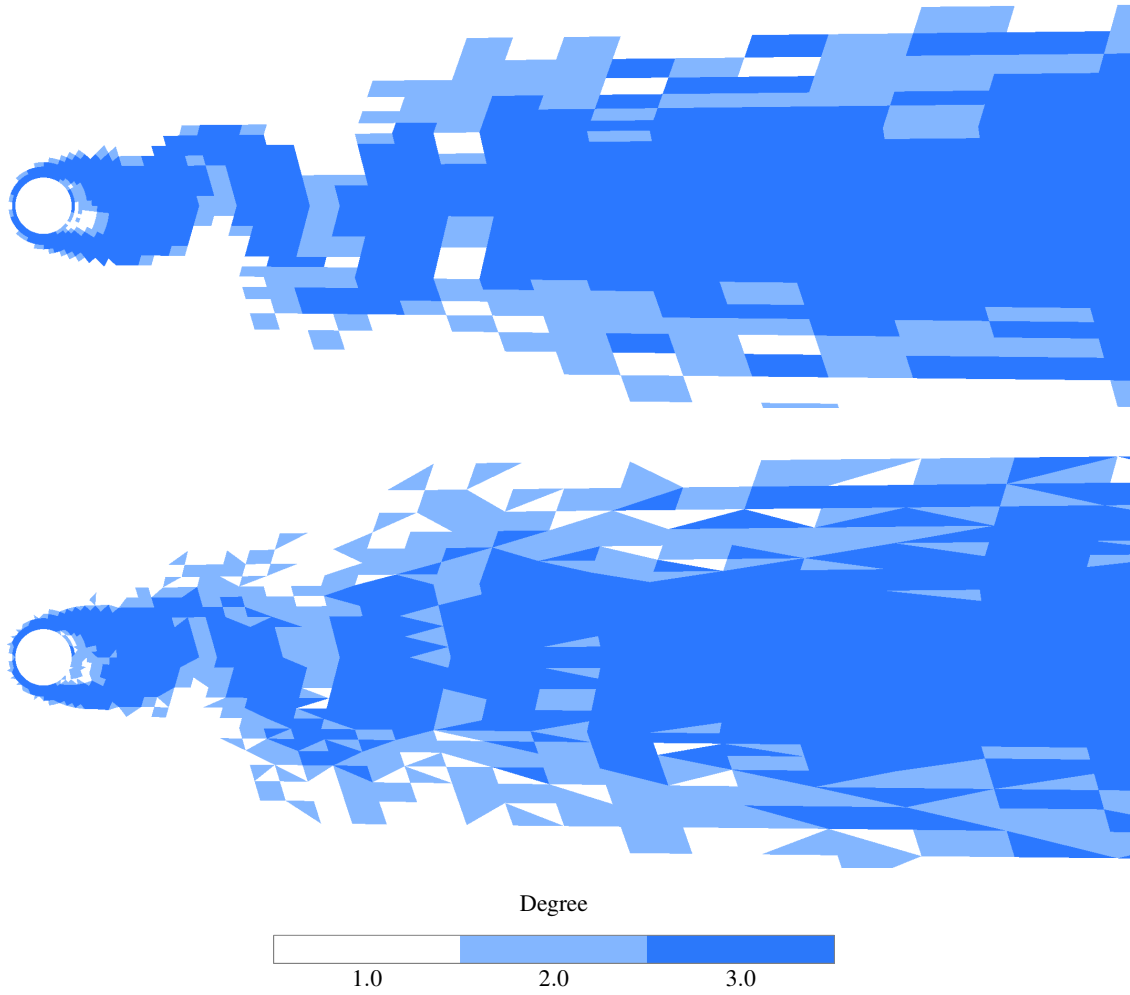


Figure 5.11. Solution polynomial distribution for three-level adaptive computation ($K1 - K3$) based on vorticity magnitude indicator corresponding to maximal lift for the circular cylinder with $Re = 150$ and $M = 0.1$ for mesh 1 (on the top) and mesh 2 (on the bottom).



(a) Velocity field for $K = 1$ for Mesh 1.



(b) Velocity field for $K = 3$ for Mesh 1.



(c) Velocity field for $K = 1 - 3$ for Mesh 1.



(d) Velocity field for $K = 1 - 3$ for Mesh 2.

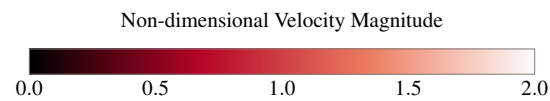


Figure 5.12. Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations for mesh 1, and three-level adaptive computation $K = 1 - 3$ for mesh 2 based on the vorticity magnitude indicator for the circular cylinder with $Re = 150$ and $M = 0.1$.



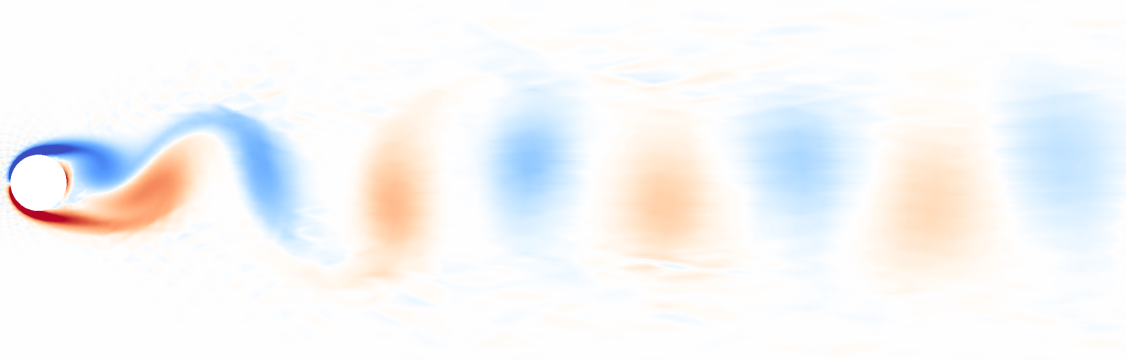
(a) Vorticity field for $K = 1$ for Mesh 1.



(b) Vorticity field for $K = 3$ for Mesh 1.



(c) Vorticity field for $K = 1 - 3$ for Mesh 1.



(d) Vorticity field for $K = 1 - 3$ for Mesh 2.

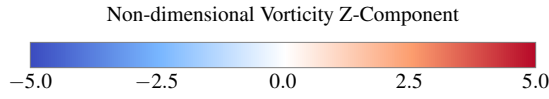


Figure 5.13. Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations for mesh 1, and three-level adaptive computation $K = 1 - 3$ for mesh 2 based on the vorticity magnitude indicator for the circular cylinder with $Re = 150$ and $M = 0.1$.

elements with Riemann invariant boundary conditions at the far-field and a no-slip adiabatic wall boundary condition at the surface of the airfoil. The computational domain extends to $20c$ above, below, and upstream of the airfoil, and $40c$ downstream, as shown in Figure 5.17. The mesh is moderately refined near the trailing and leading edges to resolve the viscous effects and uses quadratically curved elements at the boundaries to match the airfoil geometry. The mesh is initially partitioned over 4 processors using METIS [102], and MPI is used for parallel communication [99].

Simulations are carried out for a total of $100t_c$, where $t_c = D/U_\infty$ is the time required for the flow to traverse one chord length. Time integration is carried out with the SDIRK22 scheme, with implicit tolerance of 10^{-4} , and the non-dimensional time step is set to $\Delta t^* = \Delta t U_\infty / c = 2.5 \times 10^{-3}$. The Jacobian matrix is calculated every 10 iterations and when adapting, the adaptation routine is called every 10 iterations. A total of three simulations were carried out to verify the utility of the adaptation algorithm, including two uniform simulations with solution polynomials of degree $K = 1$ and $K = 3$, and a three-level adaptive simulations $K = 1 - 3$. The adaptation parameters K_m , ϵ , and $\vartheta_{1:3}$ are set to 3, 1.25×10^{-1} , and $[0.25, 0.25, 1]$ respectively. Results are compared in terms of mean lift coefficient \bar{C}_L

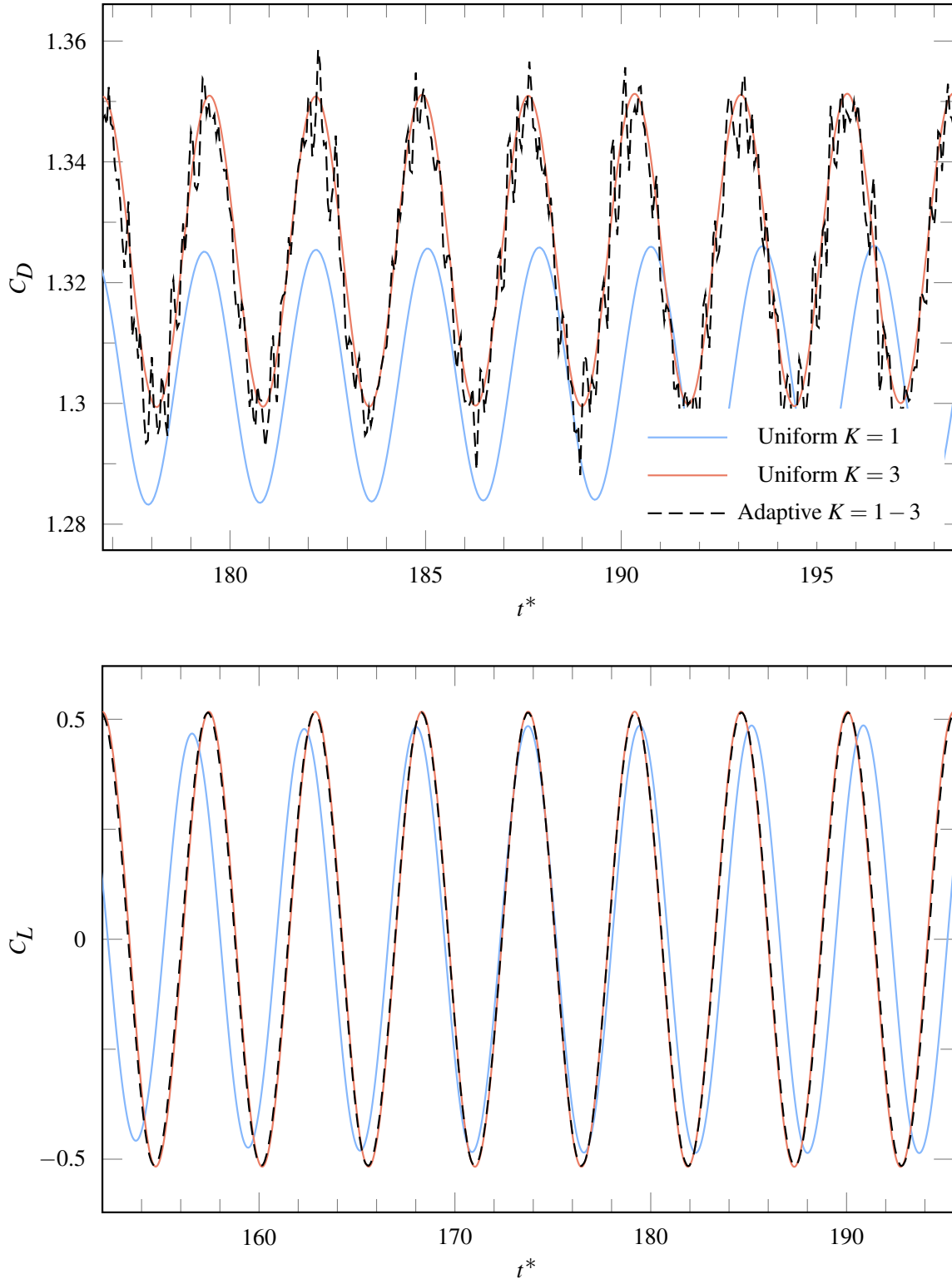


Figure 5.14. Drag and lift coefficient profiles of uniform and adaptive simulations for the circular cylinder with $Re = 150$, $M = 0.1$, and quadrilateral elements.

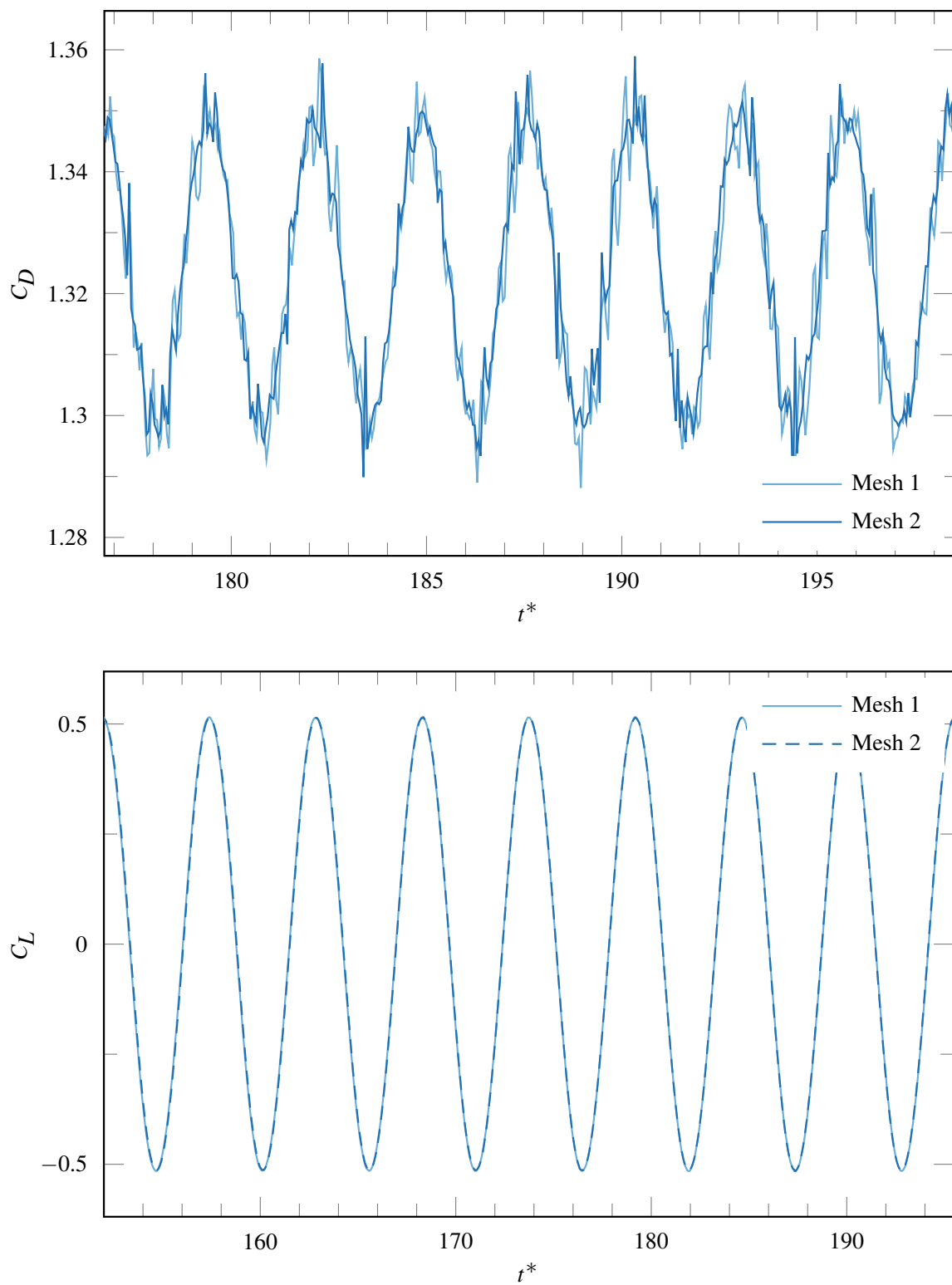


Figure 5.15. Drag and lift coefficient profiles of mesh 1 and mesh 2 adaptive simulations for the circular cylinder with $Re = 150$ and $M = 0.1$.

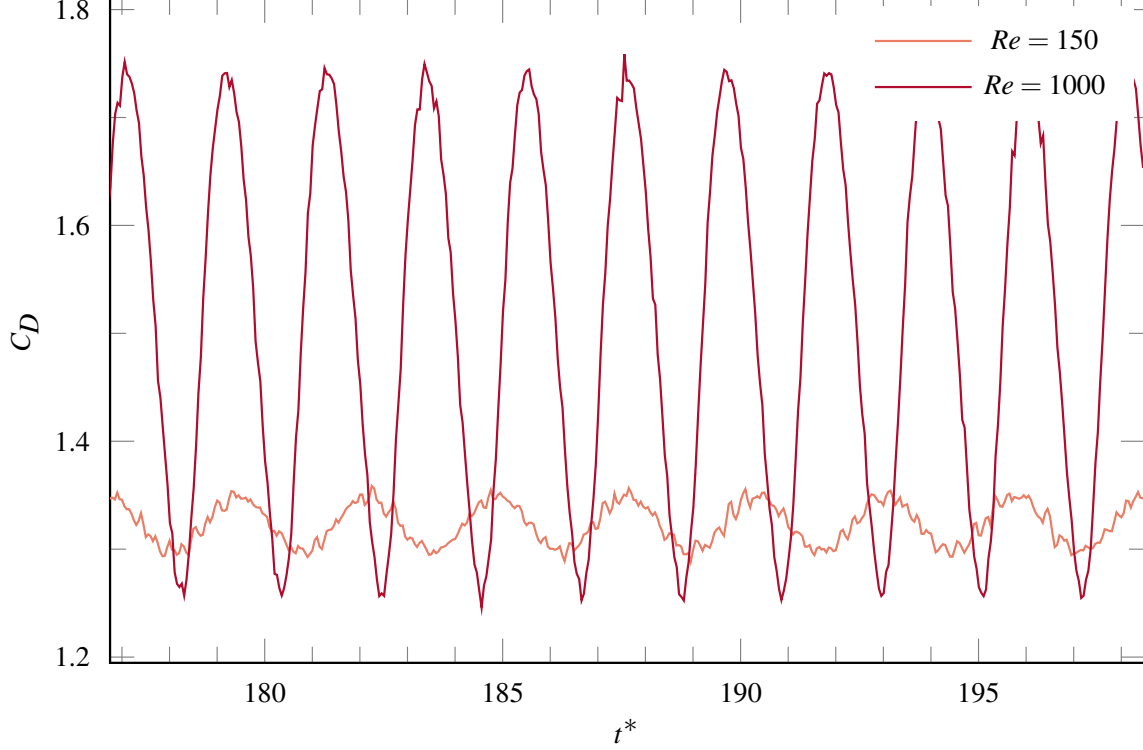


Figure 5.16. Drag coefficient profile of adaptive simulations of mesh 1 with different Reynolds numbers for the circular cylinder with $M = 0.1$.

and drag coefficient \bar{C}_D , the peak fluctuation in the drag ΔC_D coefficients, Strouhal number St , and the wall clock time T_w , where Equation 5.5 is used to calculate the lift and drag coefficients, where A_f is equal to the airfoil chord length.

Figure 5.18 shows the polynomial degree distribution for the three-level adaptive simulations ($K = 1 - 3$). It is evident that high-order elements track vorticity behind the airfoil, meaning most of the $K = 3$ elements are employed where the vorticity magnitude is maximal relative to the element size, in the vicinity of the trailing edge, on the airfoil surface before the separation point, and in the wake region. $K = 2$ elements occur further downstream, out of the wake region, leaving the far-field region for $K = 1$ elements. Figure 5.19 shows contours of non-dimensional velocity magnitude at the time of a maximal lift for uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$. While we observe unresolved regions in the flow solution of the uniform $K = 1$ simulation, the flow solution is smooth and well-resolved for the uniform $K = 3$ and the adaptive $K = 1 - 3$ simulation. Figure 5.20 shows contours of non-dimensional spanwise vorticity at the time of a maximal lift for uniform $K = 1$, uniform

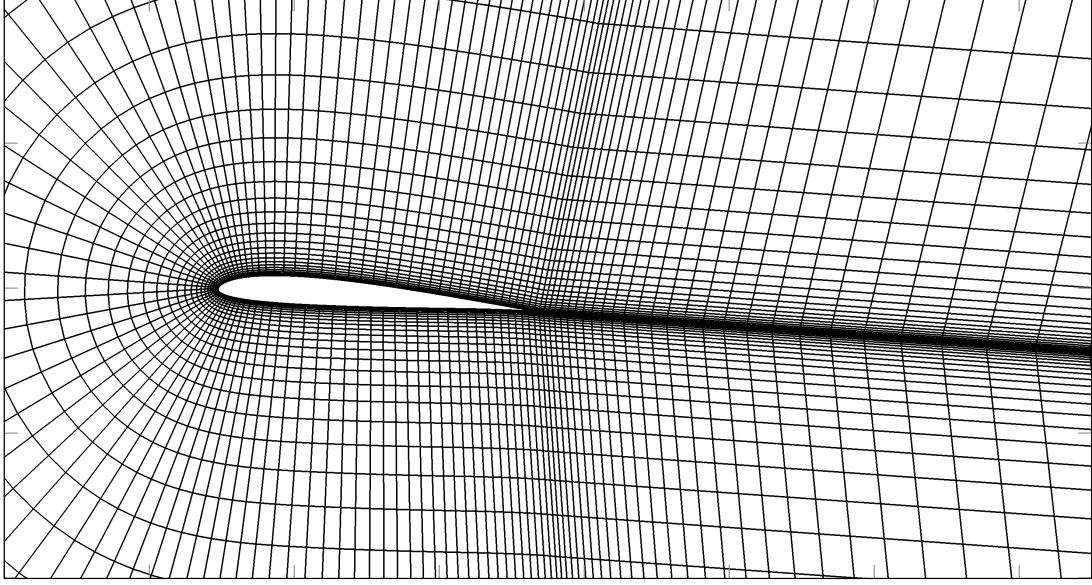


Figure 5.17. SD 7003 airfoil quadrilateral structured mesh with 8190 elements.

$K = 3$, and the adaptive $K = 1 - 3$. Comparing these results confirms that the three-level adaptive simulations qualitatively provide satisfactory resolution in the boundary layer and the wake regions, while requiring fewer degrees of freedom. The accuracy of the adaptive simulation is further illustrated in Table 5.8, which reports the numerical values of \bar{C}_L , \bar{C}_D , ΔC_D , St , and T_w for all simulations and compares them with the reference data. These results quantitatively show agreement within 0.18%, 0.20%, and 3.33% in the \bar{C}_L , \bar{C}_D , and ΔC_D values respectively between the adaptive and uniform $K = 3$ simulations, while the former is 2.17 times faster. Figure 5.21 compares the lift and drag coefficient plots of the uniform and adaptive simulations, which shows excellent agreement. Due to a higher Reynolds number compared to the circular cylinder case, the viscous drag makes up a smaller percentage of the total drag; thus, almost no spurious oscillations are observed in the C_D profile.

The following chapter will further validate the ALE implementation and exploit the utility of the novel p -adaptation algorithm for a range of applications on moving and deforming domains, including two classical problems of oscillating circular cylinders and dynamic stall of a heaving and pitching NACA 0012 airfoil, a shallow dynamic stall of a three-dimensional SD 7003 airfoil undergoing a heaving and pitching motions, and a practical test case of flow over a vertical axis wind turbine.

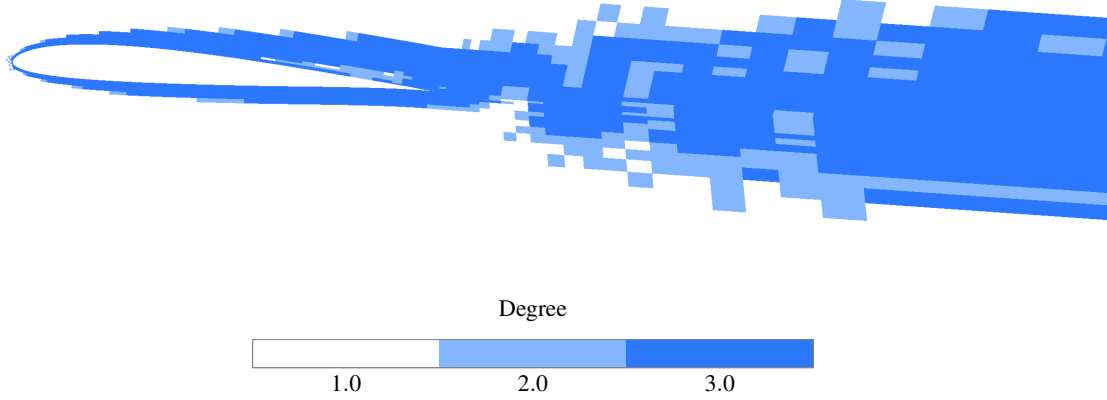


Figure 5.18. Solution polynomial distribution for three-level adaptive computation ($K = 1-3$) based on vorticity magnitude indicator corresponding to maximal lift for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$.

Table 5.8. Numerical values of ΔC_L , ΔC_D , \bar{C}_D , St , T_w , and error percent for the circular cylinder with $Re = 150$ and $M = 0.1$.

Degree	\bar{C}_L	\bar{C}_D	ΔC_D	St	$\varepsilon_{\bar{C}_L}$	$\varepsilon_{\bar{C}_D}$	$\varepsilon_{\Delta C_D}$	T_w (s)
$K = 1$	0.3559	0.0490	0.0028	1.2327	7.03%	1.8%	6.67%	53,132
$K = 3$	0.3828	0.0499	0.0030	1.2924	-	-	-	615,545
$K = 1-3$	0.3821	0.0498	0.0029	1.2966	0.18%	0.20%	3.33%	283,108
Uranga et al. Grid 2, 2D [106]	0.3755	0.04978	-	-	-			
Lopez-Morales et al. [107]	0.3719	0.04940	-	-	-			
Vermeire $K = 3$ [108]	0.3841	0.0499	-	-	-			



(a) Velocity field for $K = 1$.



(b) Velocity field for $K = 3$.



(c) Velocity field for $K = 1 - 3$.

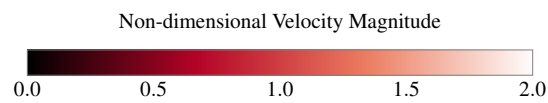
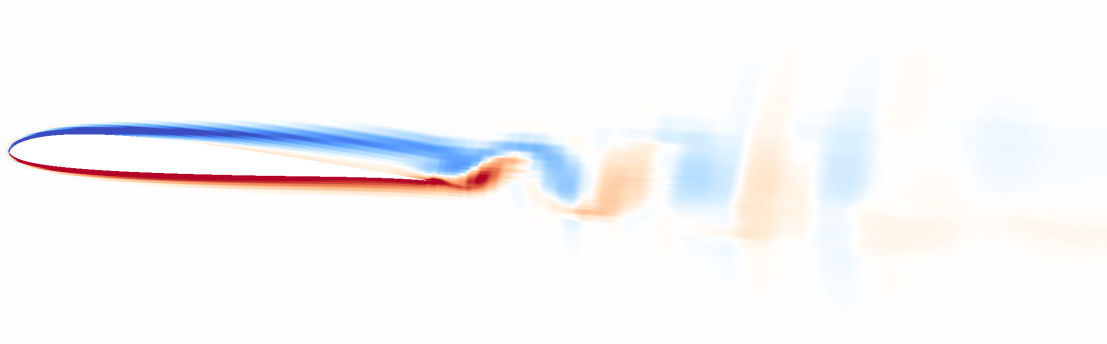
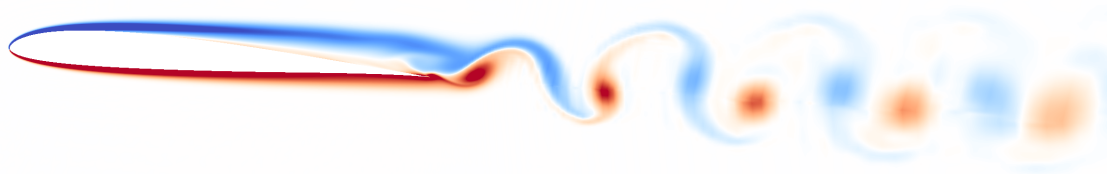


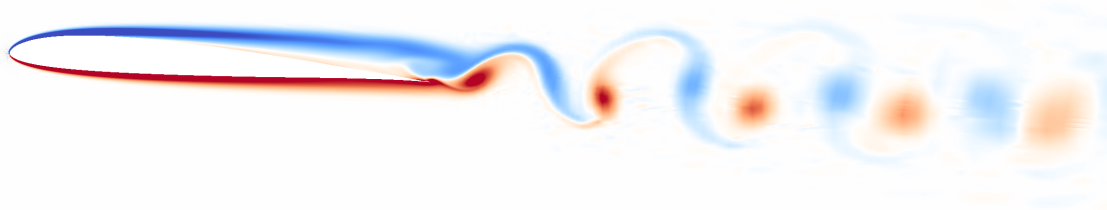
Figure 5.19. Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations based on the vorticity magnitude indicator for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$.



(a) Vorticity field for $K = 1$.



(b) Vorticity field for $K = 3$.



(c) Vorticity field for $K = 1 - 3$.

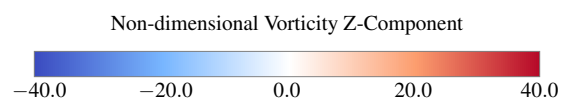


Figure 5.20. Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 3$, and adaptive $K = 1 - 3$ computations based on the vorticity magnitude indicator for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$.

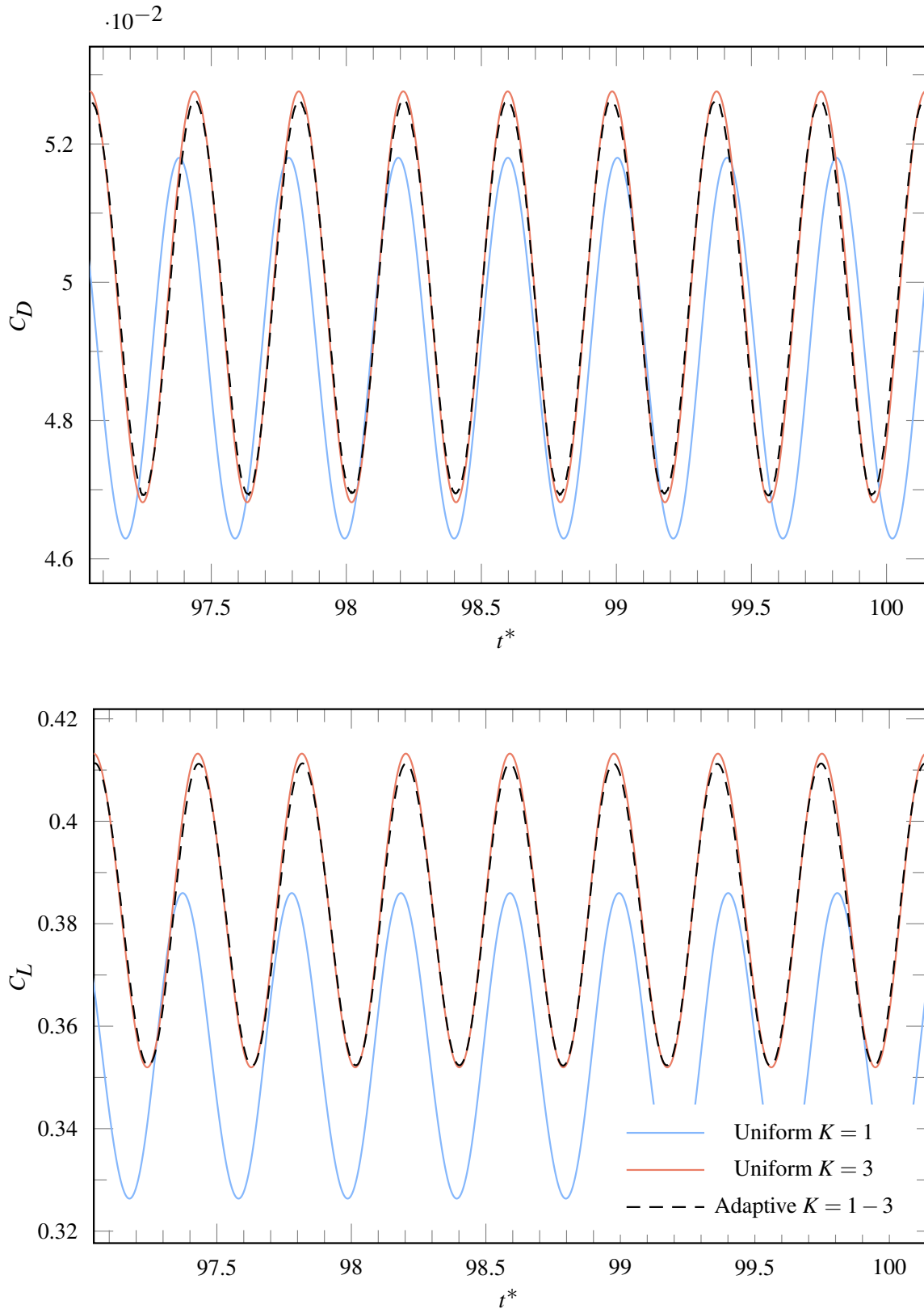


Figure 5.21. Drag and lift coefficient profiles of uniform and adaptive simulations for the SD 7003 airfoil with $Re = 10000$ and $M = 0.2$.

Chapter 6

Polynomial Adaptation for Moving and Deforming Domains

A large number of engineering applications involve moving and deforming domains, such as wind turbines, dynamic stall prediction, or deploying slats, flaps, and landing gear. These applications benefit from high-fidelity unsteady solution techniques, such as LES and DNS. As mentioned in previous chapters, it has been demonstrated previously that high-order unstructured spatial discretizations methods such as the discontinuous Galerkin [109, 110, 111], spectral volume [50], spectral difference [112], and flux reconstruction approaches [53] are particularly appealing for scale-resolving LES/DNS simulations of unsteady turbulent flows [41, 113, 114, 115, 116], and that they are particularly well-suited for modern many-core hardware architectures [47, 117, 118]. Furthermore, since they use an element-wise polynomial representation of the solution and are able to obtain arbitrarily high-order accuracy on mixed-element unstructured meshes, they are suitable for p -adaptation. This being said, in the current work, we are specifically interested in element-wise polynomial representations of the flow using the FR approach, owing to the fact that such a representation allows us to dynamically adapt the degree of the solution polynomial simultaneously increasing the local resolution of an element and its order of accuracy without requiring any complex mesh splitting operations, inherent to h -adaptation. FR approaches have been developed for moving and deforming domains with uniform polynomial degrees [119, 120]; however, in

the case of moving and deforming domains, it is particularly appealing to be able to locally increase or decrease the number of DOF to maximize accuracy and, simultaneously, reduce computational cost.

The implementation of the ALE form of the Navier-Stokes equations was verified earlier in Chapter 5. In this chapter, we explore the utility of the novel non-dimensional vorticity-based indicator for p -adaptation for a range of applications on moving and deforming domains. The rest of this chapter is organized as follows. In Section 6.1 we validate the adaptation algorithm by studying the cross-flow over an oscillating cylinder with two different configurations. We continue validation by studying dynamic stalls of a 2D NACA 0012 airfoil, in Section 6.2. Furthermore, we demonstrate the performance of the adaptation routine in a real-life application on moving and deforming domains by studying a vertical axis wind turbine composed of two NACA 0012 airfoils in Section 6.3. Finally, we present conclusions in Section 6.4.

6.1 Oscillating Circular Cylinder

6.1.1 Introduction

Cross-flow over an oscillating circular cylinder is considered to illustrate the effectiveness of the vorticity-based polynomial adaptation technique on moving domains. This problem has been the focus of several previous studies, for example, Sarpkaya [121] and Bearman [122] who studied the Vortex-Induced Vibration (VIV) of bluff bodies, an oscillation excited by the lift fluctuations due to asymmetric vortices formed around a body interacting with a flow. Bishop and Hassan [123] studied the forced oscillation of a cylinder in a water channel. Williamson and Roshko [6] extended the previous works by studying forced cross-flow oscillation of a circular cylinder for a wide range of amplitude and frequency to investigate different vorticity patterns. Blackburn et al. [124] studied the wake structures of two-dimensional flows past a cylinder for $Re = 500$ for a range of oscillation frequencies close to the natural shedding frequency of the fixed cylinder. Guilmineau et al. [125] studied the vorticity patterns of a two-dimensional forced-oscillating cylinder in a uniform flow at $Re = 185$ for a range of

excitation frequencies.

When a bluff body oscillates in the cross-flow direction by means of a driven force, the oscillation frequency f_e can synchronize with the vortex shedding frequency f_s to form the lock-in regions [6]. Williamson and Roshko mapped the lock-in regions based on different amplitude ratios A/D and wavelength ratios λ/D , where A is the oscillation amplitude, D is the cylinder diameter, and λ is the wavelength of the corresponding oscillation function defined as $\lambda = f_e U_\infty$. The wavelength ratio correlates with T_e/T_s as

$$\frac{T_e}{T_s} = St \left(\frac{\lambda}{D} \right), \quad (6.1)$$

where $T_e = 1/f_e$ is the excitation or oscillation period, $T_s = 1/f_s$ is the vortex shedding period, and $St = Df_s/U_\infty$ is the Strouhal number. The constructed A - λ plane mapping the synchronization patterns is shown in Figure 6.1, where the critical curve separates different modes of vortex formation. Williamson and Roshko categorized the vortex patterns in the wakes based on a combination of Single vortex S and vortex Pair P . Hence a $P + S$ vortex mode is a pattern where a single vortex and a vortex pair are shed at each oscillation cycle. Since the transition happens from one mode to another mode when getting close to the critical curve, an oscillation with a set of amplitude and wavelength close to the transition can result in an unrecognizable pattern. Different vortex modes are shown in Figure 6.2. Major vortex patterns observed by Williamson and Roshko near the lock-in region are $2S$, $2P$, and $P + S$ modes, where the $2S$ is the natural Karman vortex shedding. They mapped the vortex patterns based on the assumption that $St = 0.2$. This is how the T_e/T_s axis is related to the λ/D axis in Figure 6.1. Although ideally, the A - λ plane should be constructed for different Reynolds numbers, Williamson's assumption to set $St = 0.2$ seems to be precise enough for all $Re < 10000$. Furthermore, Williamson and Roshko observed that the defined boundary between $2P$ and $P + S$ modes in Figure 6.1 holds true for $300 < Re < 1000$, and for $Re < 300$, the $P + S$ mode occurred instead of the $2P$ mode.

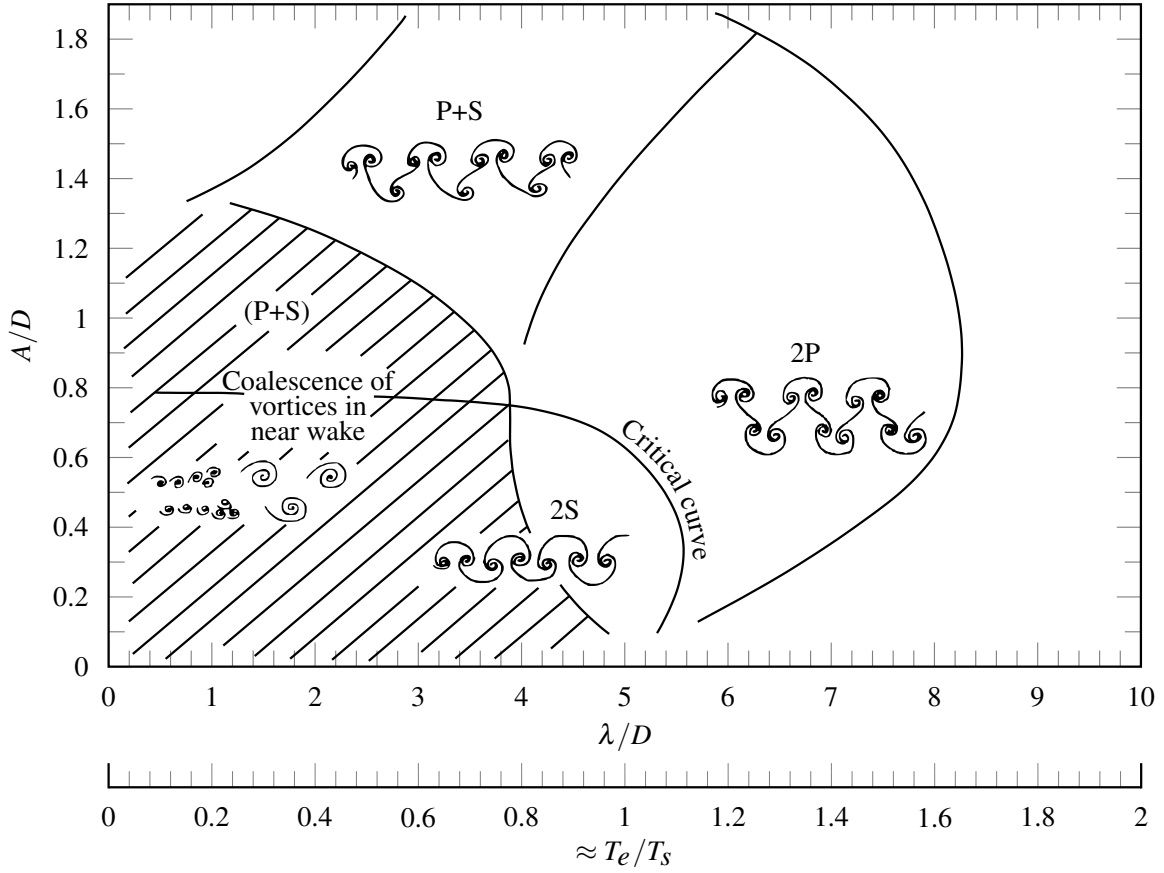


Figure 6.1. Map of vortex synchronization patterns near the fundamental lock-in region [6]. S represents a single vortex, P denotes a vortex pair, and $P + S$ indicates combination of a vortex pair and a single vortex shed at each oscillation cycle.

6.1.2 Computational Details

In this section, we study the forced-oscillation of a circular cylinder with two different sets of parameters to create different vortex patterns. In the case of a circular cylinder oscillating transversely in the free stream, the coordinates of the center of the cylinder are defined as

$$\begin{aligned} x(t) &= 0, \\ y(t) &= A \cos(2\pi f_e t), \end{aligned} \tag{6.2}$$

where A and f_e are the oscillating amplitude and excitation frequency respectively, which are non-dimensionalized by the cylinder diameter D and the fixed-cylinder vortex shedding

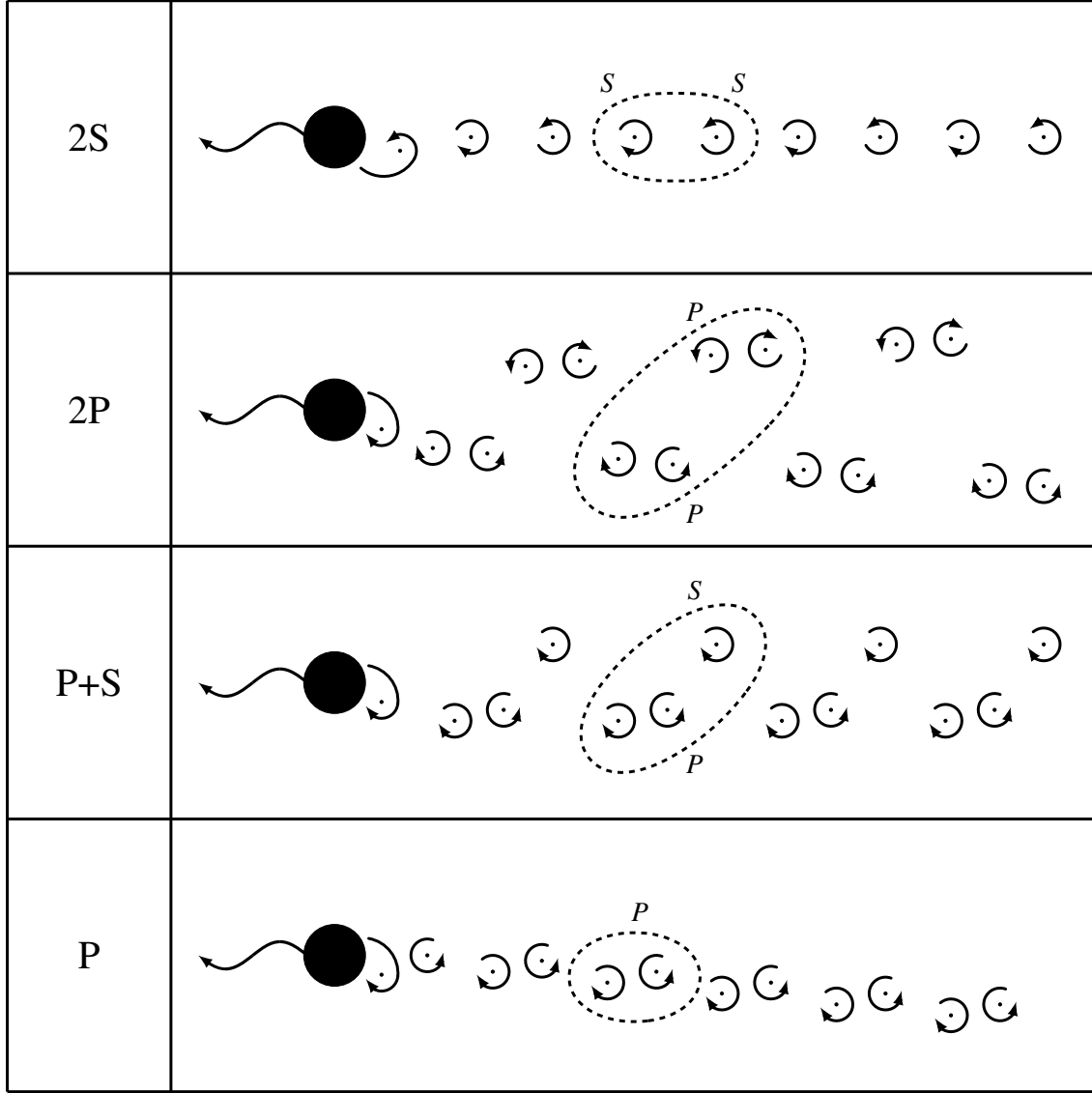


Figure 6.2. Schematic of major vortex patterns observed near the fundamental lock-in region adapted from Williamson and Roshko [6]. The dashed circle shows the vortices shed in one complete oscillation cycle.

frequency $f_s = StU_\infty/D$, known also as the Strouhal frequency. The Reynolds number is defined as $Re = \rho U_\infty D/\mu$. The location function $y(t)$ is defined as a cosine function so that the velocity function will be a sine function. This prevents a high velocity at the onset of simulation. A two-dimensional computational domain, with its origin at the center of the cylinder, consisting of 2668 quadrilateral elements moderately refined in the near-cylinder and wake regions was used with Riemann invariant boundary conditions at the far-field and a

Table 6.1. Numerical values of \bar{C}_D , C_{Lrms} , and St for a fixed circular cylinder at $Re = 185$.

Scheme	\bar{C}_D	C_{Lrms}	St
Current study, $K = 5$	1.333	0.4489	0.1928
Guilmineau et al. [125] Mesh= 120×100	1.287	0.443	0.195
Numerical results Lu and Dalton (1996) [127]	1.31	0.422	0.195
"Universal" Strouhal Williamson (1988) [126]	-	-	0.193

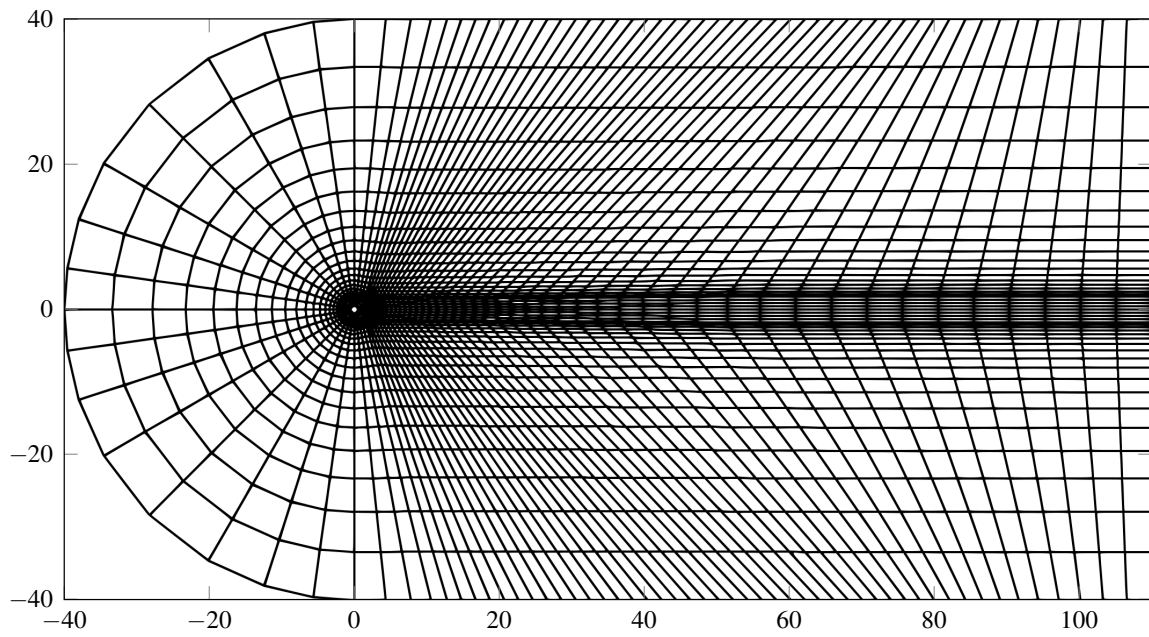
no-slip adiabatic wall boundary condition at the surface of the cylinder. The domain extends to $40D$ above, below, and upstream of the cylinder, and $80D$ downstream, as shown in Figure 6.3. Quadratically curved elements were used at the boundaries to match the cylinder geometry. The mesh is initially partitioned over 96 processors using METIS [102], and MPI is used for parallel communication [99]. Simulations are performed with two different flow settings, both with the Mach number set to $M = 0.1$, to validate the ALE implementation and the algorithm's ability to reduce the global DOF count.

6.1.3 Numerical Results

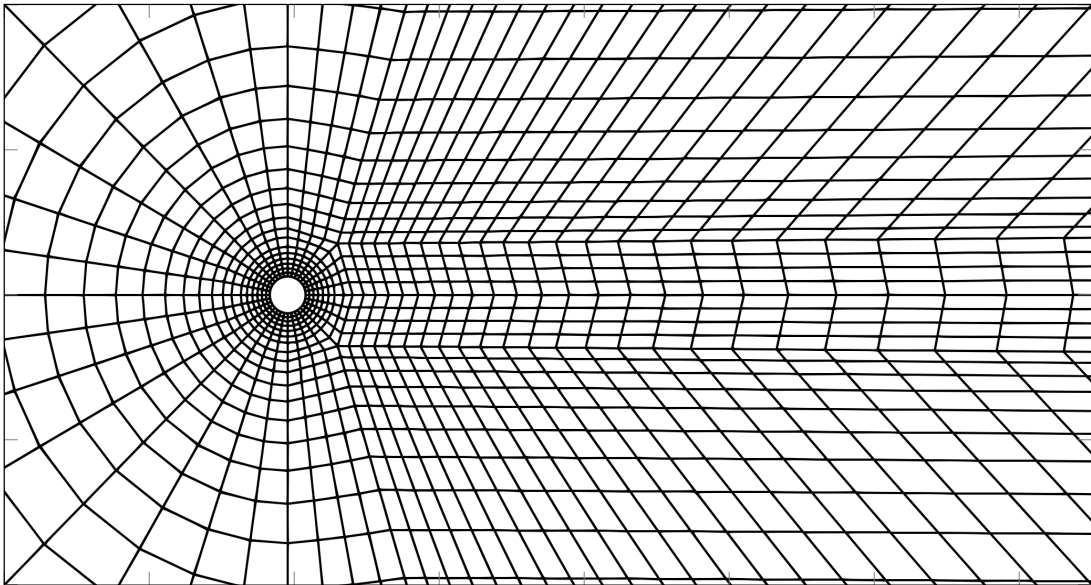
2S Vortex Mode

The first case is carried out at $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$ to recover the results of Guilmineau et al. [125]. The total simulation time is set to $400t_c$, where $t_c = D/U_\infty$ is the time required for the flow to traverse one cylinder diameter. Time integration is carried out with the classical $RK_{4,4}$ scheme, and the non-dimensional time step is set to $\Delta t^* = \Delta t U_\infty / D = 2.5 \times 10^{-4}$. We first find the static Strouhal number by running a preliminary simulation of flow past a fixed cylinder at $Re = 185$. Then, using f_s and the frequency ratio f_e/f_s , we find the cylinder oscillation frequency f_e . Table 6.1 shows the results for the fixed cylinder at $Re = 185$, results from previous studies, and the "Universal" Strouhal-Reynolds number relation given by Williamson [126], which shows excellent agreement. This also shows that Williamson's assumption to set $St = 0.2$ for the mentioned range of Re is acceptable.

Using the fixed-cylinder vortex shedding frequency f_s , we specify the excitation frequency from $f_e/f_s = 1.1$. In order to validate the utility of the adaptation algorithm, six simulations are performed, five uniform simulations with solution polynomials of degree $K = 1$ to $K = 5$,



(a) Complete view. Dimensions are given in terms of cylinder diameter D .



(b) Close-up view.

Figure 6.3. Circular cylinder quadrilateral structured mesh with 2668 elements.

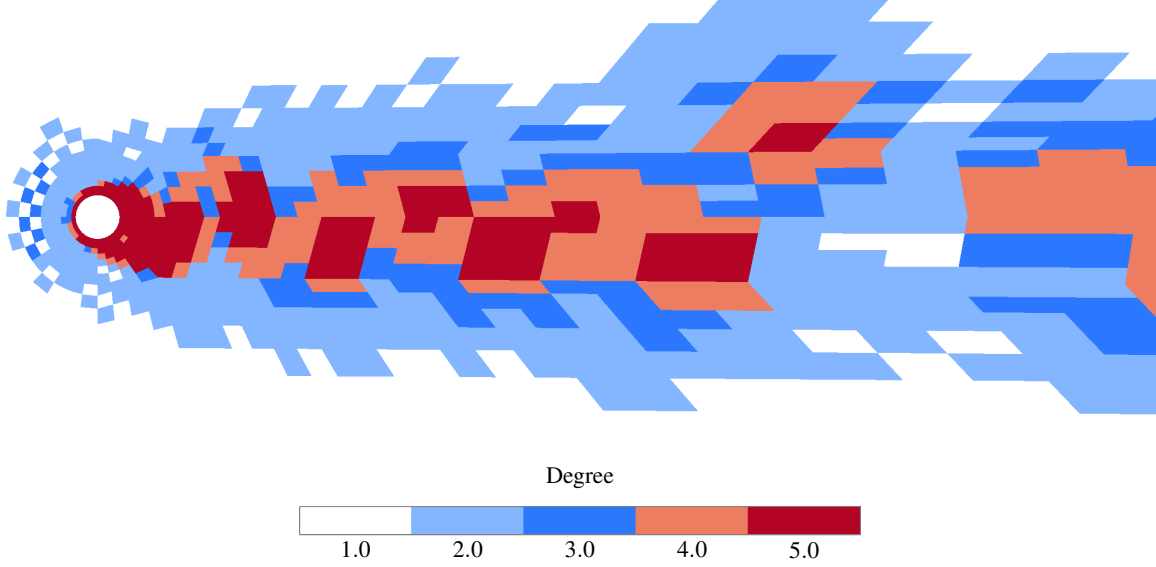


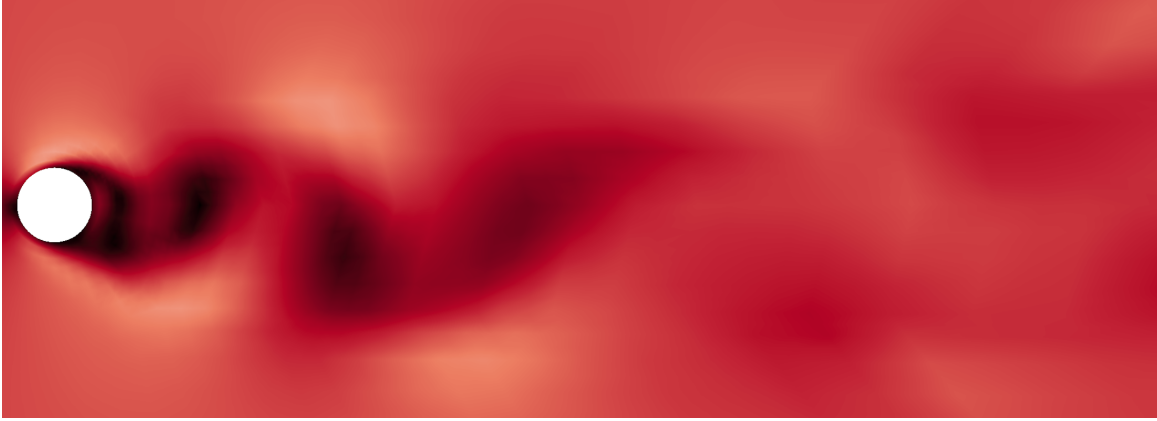
Figure 6.4. Polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the time corresponding to maximal lift for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$.

and one adaptive simulation with parameters K_m , ϵ , and $\vartheta_{1:5}$ set to 5, 6.25×10^{-3} , and $[0.25, 0.25, 1, 4, 16]$ respectively. The adaptation routine is called every 10 iterations. Results are compared in terms of mean drag coefficient \bar{C}_D , root mean square of the lift coefficient C_{Lrms} , root mean square of the drag coefficient C_{Drms} , Strouhal number St , and the average total number of degrees of freedom \overline{DOF} , where the lift and drag coefficients are defined as

$$\begin{aligned} C_L &= \frac{2F_L}{\rho_\infty U_\infty^2 L}, \\ C_D &= \frac{2F_D}{\rho_\infty U_\infty^2 L}, \end{aligned} \quad (6.3)$$

where L is the characteristic length which, for this case study, is taken to be the cylinder diameter D . The average and the root mean square (rms) value of any time dependant variable $v(t)$ over a time interval $T_{int} = t_n - t_1$ are defined as

$$\begin{aligned} \bar{v} &= \frac{1}{n} \sum_{t_1}^{t_n} v(t), \\ v_{rms} &= \sqrt{\frac{1}{n} \sum_{t_1}^{t_n} (v(t)^2 - \bar{v}^2)} \end{aligned} \quad (6.4)$$



(a) Velocity field for $K = 1$.



(b) Velocity field for $K = 5$.



(c) Velocity field for $K = 1 - 5$.

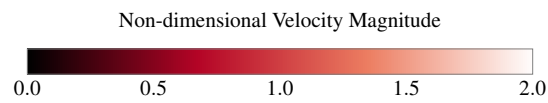


Figure 6.5. Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$.



(a) Vorticity field for $K = 1$.



(b) Vorticity field for $K = 5$.



(c) Vorticity field for $K = 1 - 5$.

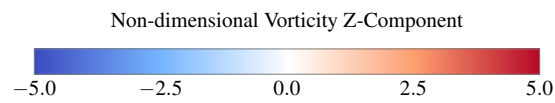


Figure 6.6. Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$.

respectively, and the Strouhal number is defined as

$$St = \frac{f_s D}{U_\infty} \quad (6.5)$$

where f_s is the shedding frequency of the dominant vortices in the wake. Figure 6.4 shows the polynomial degree distribution for the five-level adaptive simulation ($K = 1 - 5$). As mentioned earlier, elements with large vorticity magnitudes relative to the effective mesh resolution are adapted to higher-degree polynomials. It is evident that most of the $K = 5$ and $K = 4$ elements are employed where the vorticity magnitude is maximal relative to the element size, which illustrates high-order elements track vorticity behind the cylinder. The $K = 3$ and $K = 2$ elements occur further downstream, out of the wake region, leaving the far-field region for $K = 1$ elements. This verifies that the algorithm successfully tracks the location of elements with large vorticity magnitude relative to their element size. Figure 6.5 shows contours of non-dimensional velocity magnitude at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. While we observe large inter-element jumps in the flow solution of the uniform $K = 1$ simulation, the flow solution is smooth for the uniform $K = 5$ and adaptive $K = 1 - 5$ simulations. Figure 6.6 shows contours of non-dimensional spanwise vorticity at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$, where vorticity is defined as $\boldsymbol{\omega} = \nabla \times \mathbf{u}$. Observing the vortex pattern, we see a von Karman vortex street defined as the $2S$ mode. Comparing the vortex pattern for $A/D = 0.2$ and $T_e/T_s = f_s/f_e \approx 0.91$ against Williamson's vortex map, we can see the pattern sits in the $2S$ mode region close to a critical curve, which validates the ALE implementation and the computer code. Furthermore, comparing these results confirms that the five-level adaptive simulation qualitatively provides satisfactory resolution in the near-cylinder and the wake regions, while requiring fewer degrees of freedom. To further evaluate the accuracy of the adaptive method, we turn to Table 6.2, which reports the numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , and the average total number of degrees of freedom \overline{DOF} for all uniform simulations and the five-level adaptive simulation and compares them with results from Guilmineau et al. [125]. These results quantitatively show agreement within 0.03%, 0.21%, and 0.30% in the \bar{C}_D , C_{Drms} , and C_{Lrms} values respectively between

Table 6.2. Numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , and error percent for the oscillating circular cylinder at $Re = 185$ with $A/D = 0.2$ and $f_e/f_s = 1.1$.

Degree	\bar{C}_D	C_{Drms}	C_{Lrms}	St	$\varepsilon_{\bar{C}_D}$	$\varepsilon_{C_{Drms}}$	$\varepsilon_{C_{Lrms}}$	\overline{DOF}
$K = 1$	1.2872	0.0720	0.7685	0.2121	8.09%	49.22%	13.93%	10672
$K = 2$	1.3784	0.1411	0.8822	0.2120	1.58%	0.49%	1.20%	24012
$K = 3$	1.4065	0.1444	0.8916	0.2120	0.43%	1.83%	0.15%	42688
$K = 4$	1.4001	0.1414	0.8928	0.2120	0.03%	0.28%	0.01%	66700
$K = 5$	1.4005	0.1418	0.8929	0.2120	-	-	-	96048
$K = 1 - 5$	1.4009	0.1421	0.8902	0.2119	0.03%	0.21%	0.30%	27816
Guilmineau et al. [125] Mesh= 240×200	1.420	0.149	0.897	0.24	-	-	-	-

the adaptive and uniform $K = 5$ simulations, while the adaptive simulation requires 3.45 times fewer degrees of freedom. Figure 6.7 compares the lift and drag coefficient plots of our adaptive simulation with the results of Guilmineau et al. [125], and Figure 6.8 compares the lift and drag coefficient plots of the uniform and adaptive simulations, which show excellent agreement.

$P + S$ Vortex Mode

The effect of the amplitude and frequency of the oscillation on the vortex shedding modes has been studied previously [6]. To validate the ability of the adaptation algorithm to capture more complex shedding modes, a different set of oscillation amplitude and frequency are selected to produce the $P + S$ wake mode, with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$. Unlike the previous case, we have directly selected the excitation frequency, hence there is no need to find the static Strouhal number by running an extra simulation for the fixed cylinder. As in the previous case, a p -refinement study with five uniform simulations with solution polynomials of degree $K = 1$ to $K = 5$, and a five-level adaptive simulation with the same adaptation parameters are performed. The mesh, boundary conditions, total simulation time, time integration scheme, and non-dimensionalized time step are the same as in the first case. Results are compared again in terms of mean drag coefficient \bar{C}_D , C_{Lrms} , C_{Drms} , Strouhal number St , and the average total number of degrees of freedom \overline{DOF} . Figure 6.9 shows the polynomial degree distribution of the five-level adaptive simulation ($K = 1 - 5$), which confirms that the algorithm successfully tracks elements with large

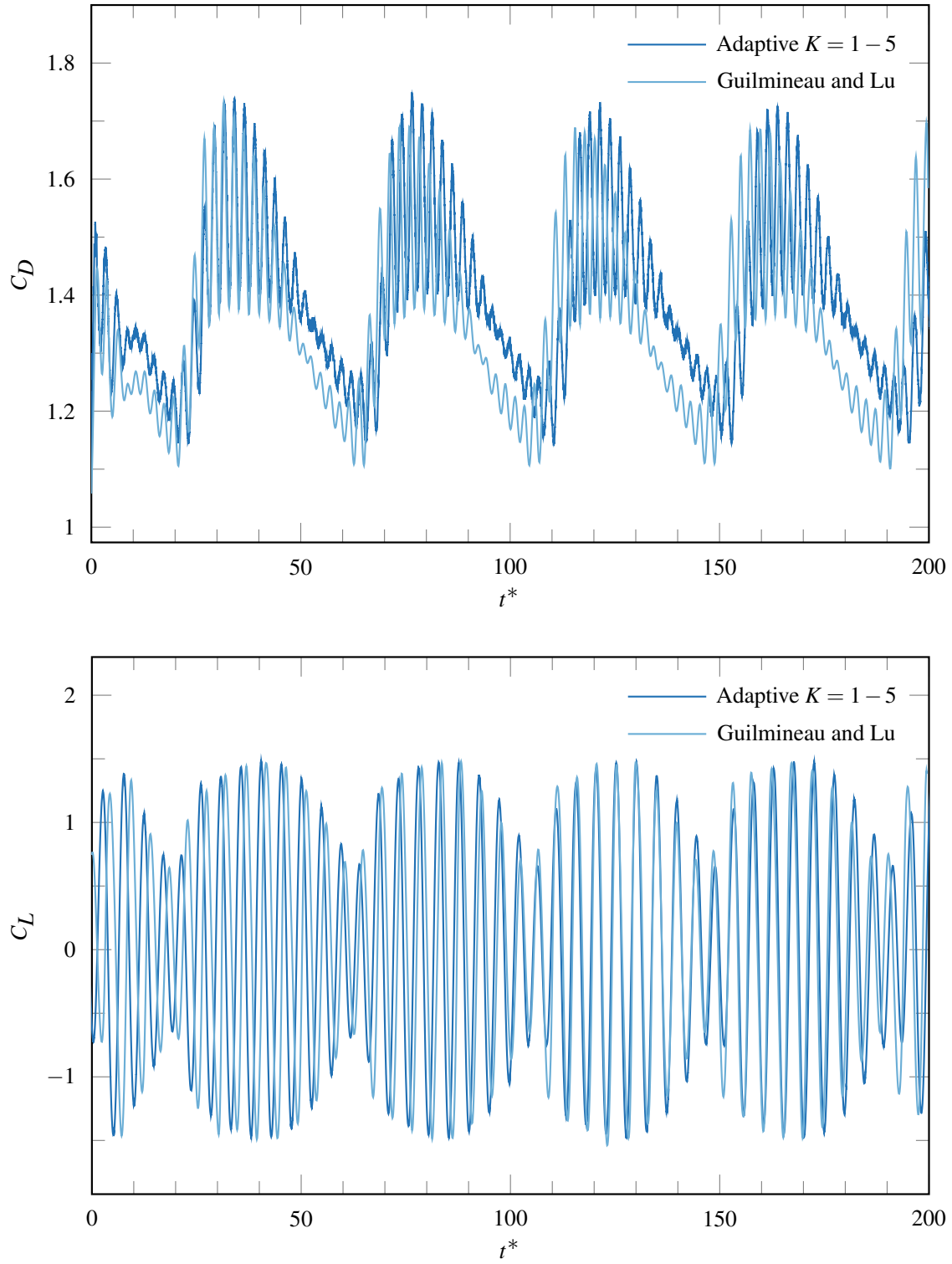


Figure 6.7. Drag and lift coefficient profiles of the adaptive and reference simulations performed by Guilmineau et al. for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$.

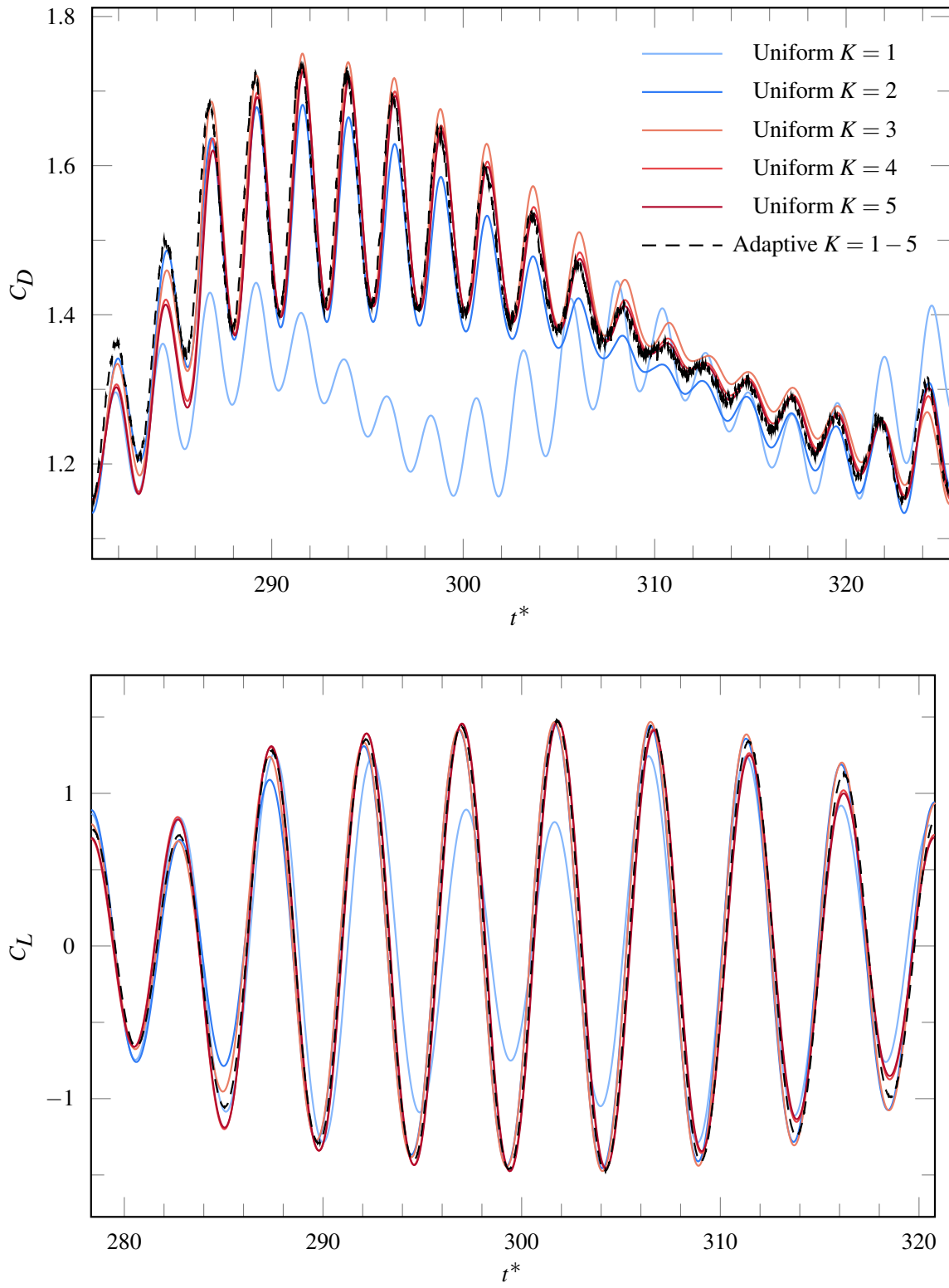


Figure 6.8. Drag and lift coefficient profiles of uniform and adaptive simulations for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, and $f_e/f_s = 1.1$.

Table 6.3. Numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , and error percent for the oscillating circular cylinder at $Re = 200$ with $A/D = 1$ and $f_e = 0.0193$.

Degree	\bar{C}_D	C_{Drms}	C_{Lrms}	St	$\varepsilon_{\bar{C}_D}$	$\varepsilon_{C_{Drms}}$	$\varepsilon_{C_{Lrms}}$	\overline{DOF}
$K = 1$	1.9981	0.7700	1.0417	0.1632	1.32%	7.24%	6.85%	10672
$K = 2$	2.0135	0.8239	1.0772	0.1632	0.56%	0.75%	3.68%	24012
$K = 3$	2.0194	0.8276	1.1119	0.1632	0.27%	0.30%	0.57%	42688
$K = 4$	2.0251	0.8304	1.1186	0.1632	0.01%	0.04%	0.03%	66700
$K = 5$	2.0249	0.8301	1.1183	0.1632	-	-	-	96048
$K = 1 - 5$	2.0240	0.8302	1.1190	0.1632	0.04%	0.01%	0.06%	30864

vorticity magnitude relative to element size behind the cylinder. Figure 6.10 shows contours of non-dimensional velocity magnitude at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. The flow solution is smooth for the uniform $K = 5$ and adaptive $K = 1 - 5$ simulations, and large inter-element jumps are not observed in the adaptive simulation. Figure 6.11 shows the non-dimensionalized z-component of the vorticity field at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. Comparing these confirms that the five-level adaptive simulation qualitatively provides satisfactory resolution in the boundary layer and the wake region while requiring fewer degrees of freedom. Additionally, based on $A/D = 1$ and $T_e/T_s = f_s/f_e \approx 1.23$ ¹, the vortex mode must be $P + S$ ² from Williamson's vortex map. We see that our results also show a $P + S$ vortex mode, which validates the ALE implementation and the computer code in capturing more complex shedding modes. Table 6.3 reports quantitative values of the p -refinement study and the adaptive simulation, which shows the adaptive simulation quantitatively agrees with the uniform $K = 5$ simulation within 0.04%, 0.01%, and 0.06% numerical error in the \bar{C}_D , C_{Drms} , and C_{Lrms} values respectively, while requiring 3.11 times fewer degrees of freedom. In order to better observe the required DOF, we can turn to Figure 6.12 which compares the mean number of degrees of freedom for the uniform and adaptive simulations for both oscillating cylinder cases. This shows that the adaptive simulation consistently requires significantly fewer degrees of freedom for comparable levels of accuracy.

¹Based on Williamson's assumption of $St = 0.2$ which gives $T_s \approx 42.26$ from $T_s = D/StU_\infty$.

²For $Re < 300$, $P + S$ vortex mode appears instead of the $2P$ mode.

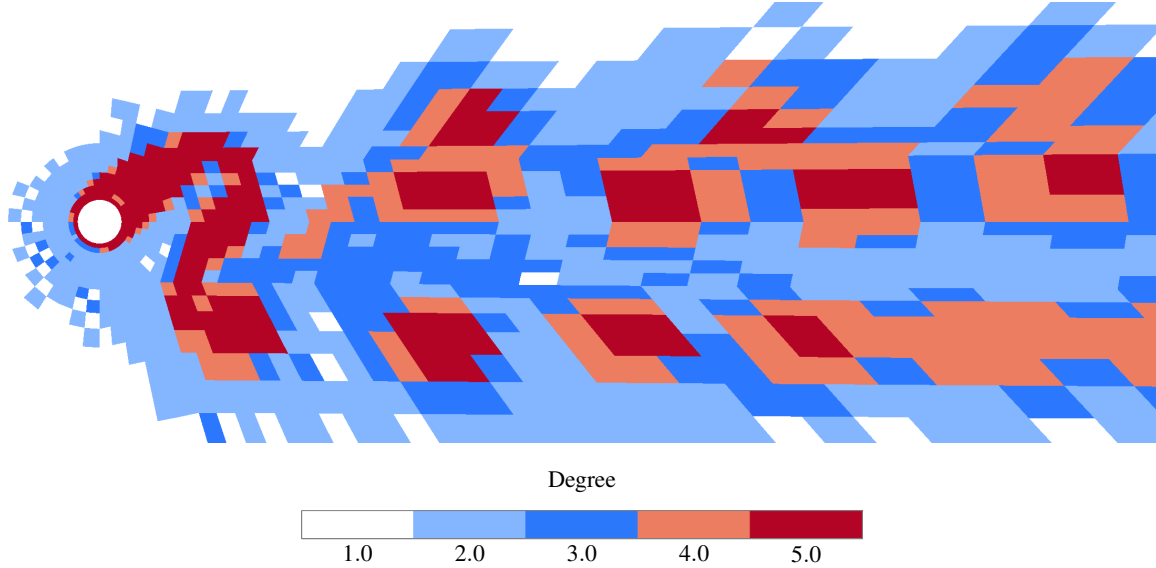


Figure 6.9. Solution polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the time of maximal lift for the oscillating circular cylinder with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$.

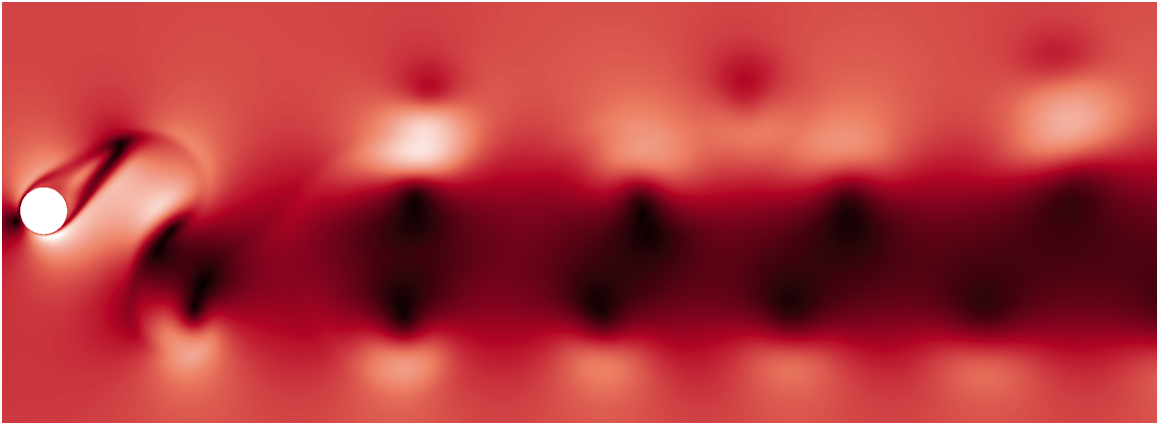
6.2 2D Dynamic Stall of a NACA 0012 Airfoil

6.2.1 Introduction

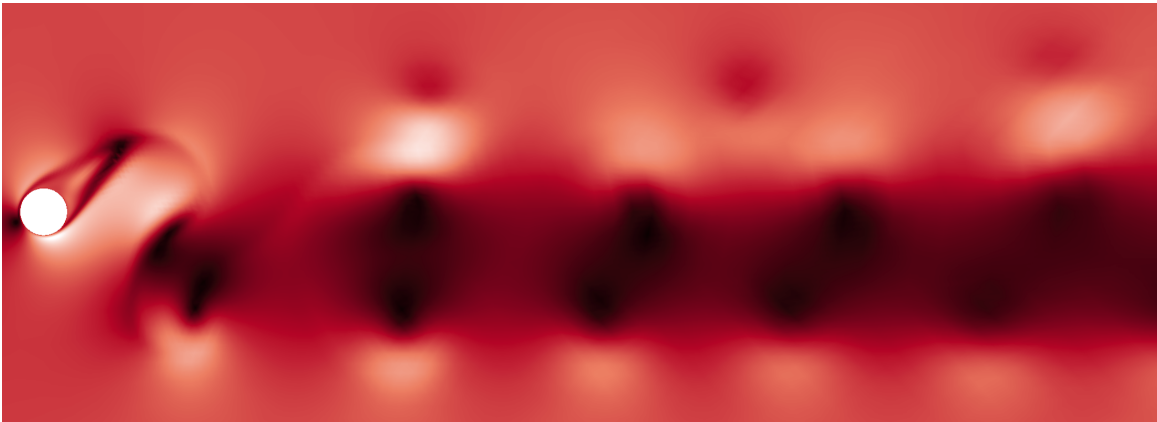
Static stall happens when the flow separates from the wing due to a high angle attack. This causes a sudden decrease in the lift, which in the case of an aircraft, means losing altitude with the nose up. The angle at which stall happens depends on the wing geometry and varies for different airfoils. Dynamic stall, on the other hand, occurs when an airfoil undergoes a rapid pitch manoeuvre and produces high lift for a short time interval, followed by a sudden drop [128]. The high lift, which is caused by the formation of a Leading-Edge Vortex (LEV) due to the pitch manoeuvre, plateaus as the LEV traverses along the airfoil. As soon as the vortex passes the trailing edge, the airfoil suffers a dramatic decrease in lift and a stall happens. This is typical of fixed-wing aircraft. However, flapping wings can create a second LEV before the first vortex passes the trailing edge. This way the high lift period continues which prevents the wing from stalling. In fact, the leading-edge vortex has been identified as the main lift enhancing mechanism of flapping wings, which is produced consecutively in each stroke and prevents the wing from stalling [129]. This phenomenon, with the recent development of micro air vehicles, has attracted numerous authors to study



(a) Velocity field for $K = 1$.



(b) Velocity field for $K = 5$.



(c) Velocity field for $K = 1 - 5$.

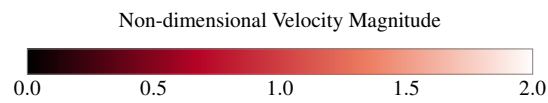


Figure 6.10. Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the oscillating circular cylinder with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$.



(a) Vorticity field for $K = 1$.



(b) Vorticity field for $K = 5$.



(c) Vorticity field for $K = 1 - 5$.

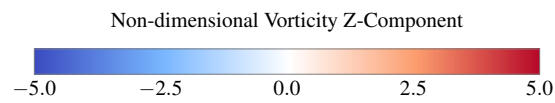


Figure 6.11. Non-dimensional z-component of the vorticity for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on vorticity magnitude indicator for the oscillating circular cylinder with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$.

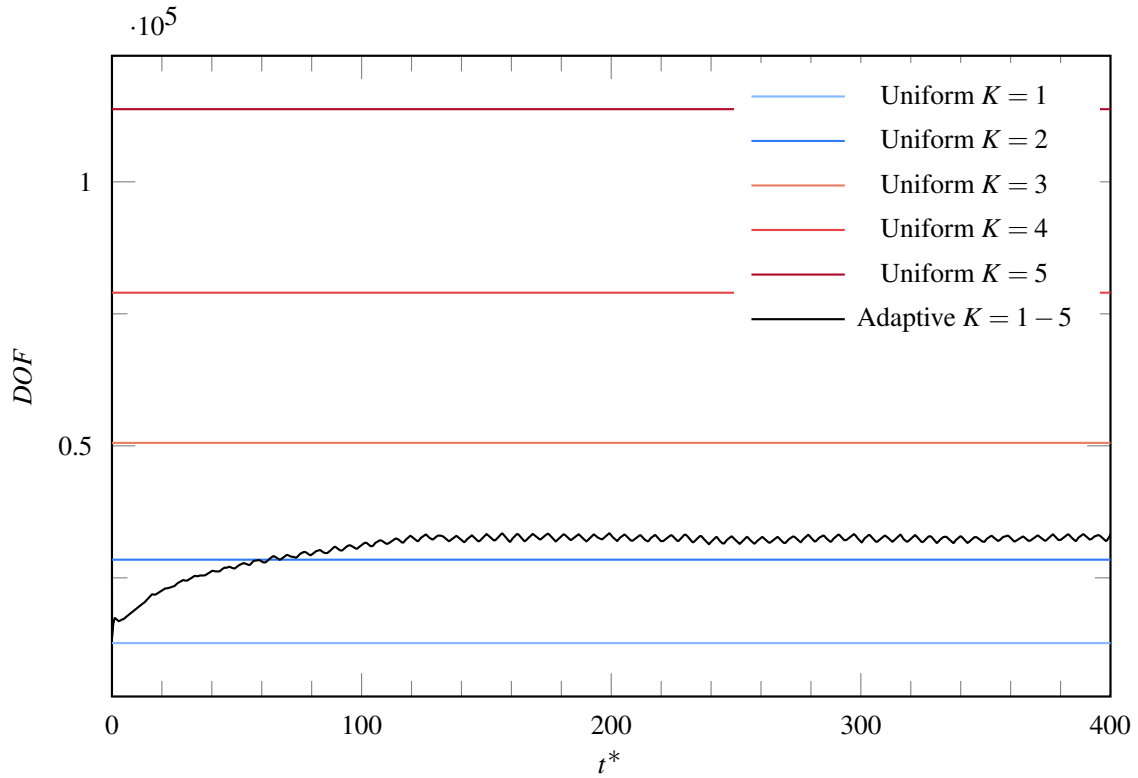
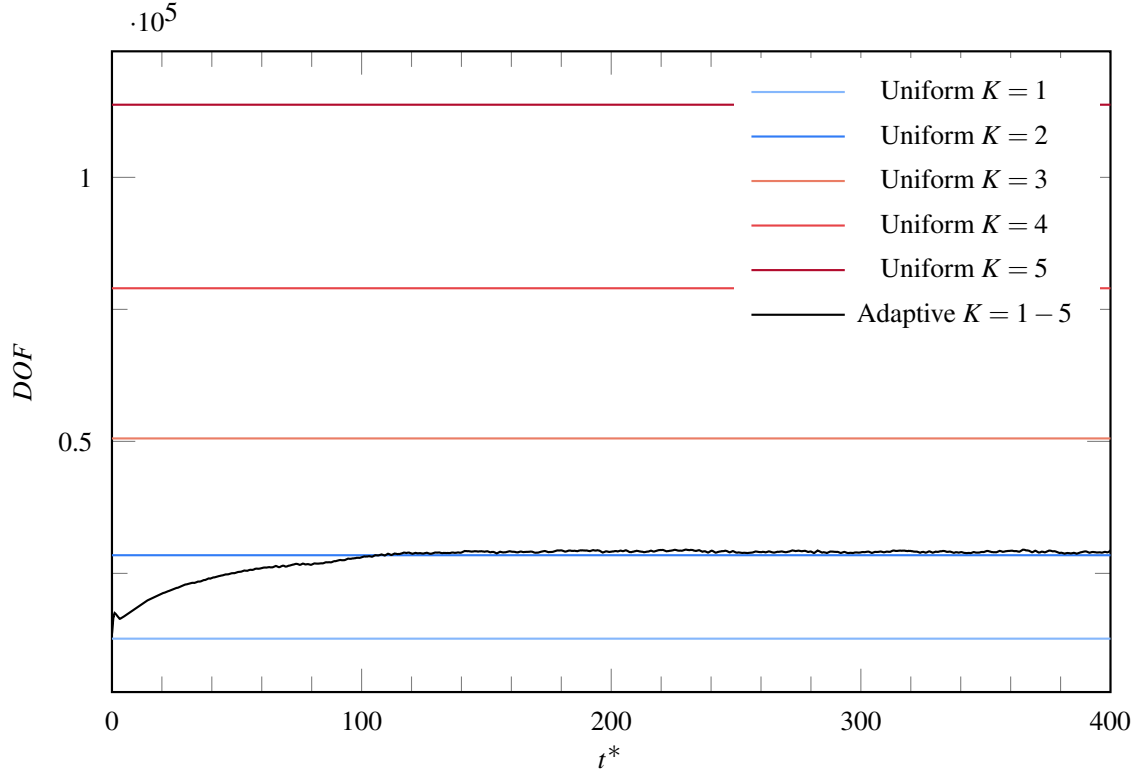


Figure 6.12. Degrees of freedom profiles for the uniform and adaptive simulations for the oscillating circular cylinder with $Re = 185$, $A/D = 0.2$, $f_e/f_s = 1.1$ on the top, and with $Re = 200$, $A/D = 1$, and $f_e = 0.0193$ on the bottom.

2D airfoils undergoing pure heaving or pitching motions [130, 131, 132, 133, 134, 135], where the effect of leading and trailing edge vortices are of interest.

6.2.2 Computational Details

In this section, a 2D NACA 0012 airfoil undergoing both heaving and pitching motions with a phase shift at a low Reynolds number is studied in order to validate the performance of the vorticity-based polynomial adaptation technique for moving and deforming domains at low Reynolds numbers. The translation of the coordinates of the center of oscillation and the pitching function are defined as

$$\begin{aligned} x(t) &= 0, \\ y(t) &= A \cos(2\pi f_e t), \\ \theta(t) &= \theta_0 + \theta_e \cos(2\pi f_e t + \phi_e), \end{aligned} \tag{6.6}$$

where A is the heaving oscillating amplitude, which is non dimensionalized by the airfoil chord length, f_e is the excitation frequency which is defined in terms of the reduced frequency $f_K = \pi f_e c / U_\infty$, θ_0 is the mean pitch angle value, θ_e is the pitch amplitude, and ϕ_e is the phase shift between the heaving and pitching motions. The airfoil effective angle of attack $\alpha_e(t)$ is defined as

$$\alpha_e(t) = \theta(t) - \tan^{-1} \left(\frac{y'(t)}{U_\infty} \right). \tag{6.7}$$

Figure 6.13 shows plots of vertical location $y(t)$, vertical velocity $y'(t)$, pitching angle $\theta(t)$, and the effective angle of attack for one cycle. The oscillation functions are defined as a cosine function to prevent a large velocity at the onset of simulations. The pitching motion used here is a rotation about the center of pressure, x_{cp} , which, for the NACA 0012, is located $0.25c$ from the leading edge. The Reynolds number is defined as $Re = \rho U_\infty c / \mu$, where c is the airfoil chord. A two-dimensional computational domain, with its origin at the leading edge, consisting of 2722 unstructured quadrilateral and triangular elements, moderately refined near the trailing and leading edges was used with Riemann invariant boundary condition at the far-field and a no-slip adiabatic wall boundary condition at the surface of the airfoil. The domain extends to $12.5c$ above, below, and upstream of the airfoil, and $25c$ downstream, as

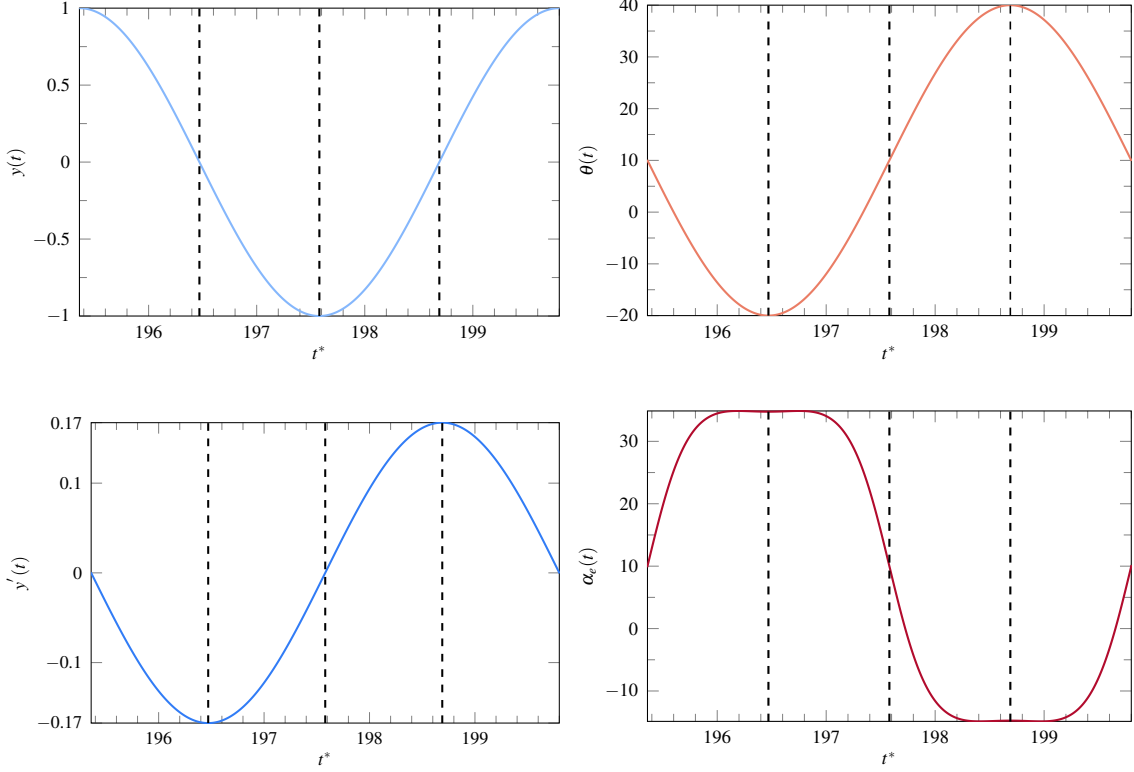


Figure 6.13. Plots of vertical location $y(t)$, vertical velocity $y'(t)$, oscillation pitch angle $\theta(t)$, and effective angle of attack $\alpha_e(t)$ for the pitching and heaving NACA 0012 airfoil with $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$. Dashed lines show the location of $t = 1T_e/4$, $t = 2T_e/4$, and $t = 3T_e/4$.

shown in Figure 6.14. Quadratically curved elements were used at the boundaries to match the airfoil geometry. The mesh is initially partitioned over 96 processors using METIS [102], and MPI is used for parallel communication [99].

Simulations were carried out at $Re = 1000$, $M = 0.1$, $A/c = 1$, $f_K = 0.708$, resulting in a period of oscillation of $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$, and $\phi_e = 90^\circ$ in order to reproduce the results of Moriche et al. [7], who studied the effect of the mean pitch angle and phase shift between the heaving and pitching motions on the aerodynamic forces. The total simulation time is set to $200t_c$ which was sufficient for subsequent cycles to show nearly identical behaviour where, in this case, t_c is the time required for the flow to traverse one airfoil chord. Time integration is carried out with the two-stage second order singly diagonally implicit Runge–Kutta scheme, SDIRK_{2,2}, with a non-dimensional time step of $\Delta t^* = \Delta t U_\infty / D = 2.5 \times 10^{-3}$. As in the previous section, six simulations, including five

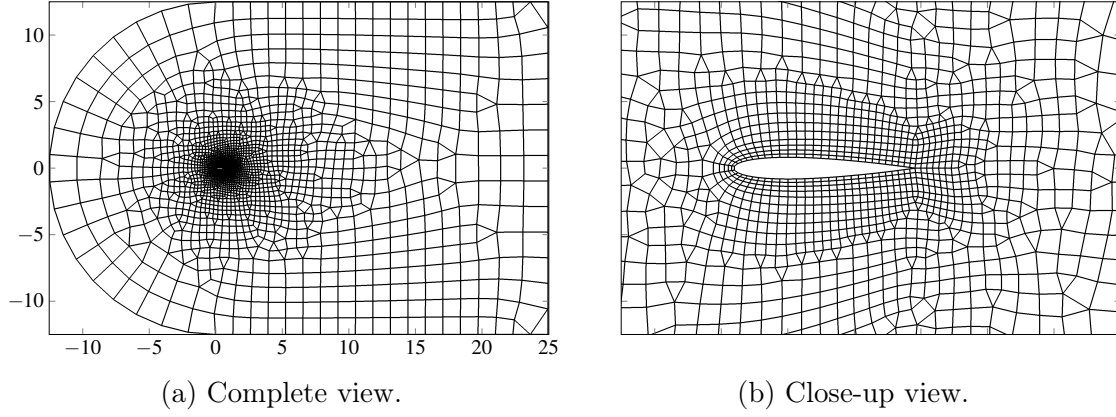


Figure 6.14. The NACA 0012 unstructured mesh with 2722 quadrilateral and triangular elements. Dimensions are given in multiples of the airfoil chord length.

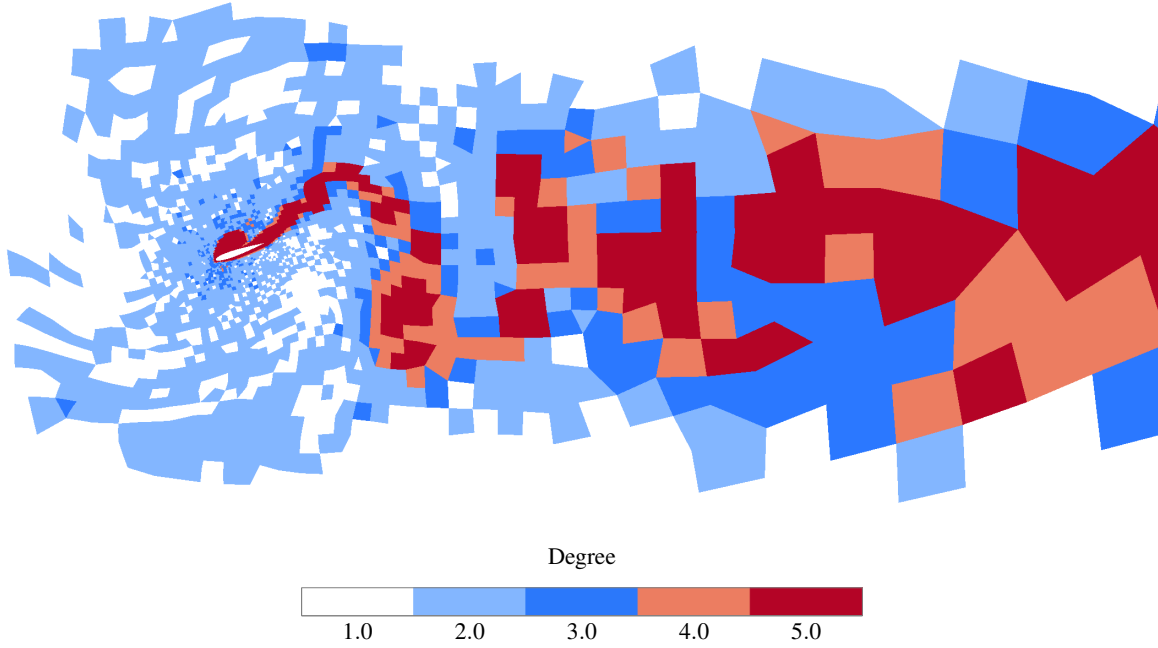


Figure 6.15. Polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the time corresponding to maximal lift for the pitching and heaving NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.

uniform simulations with solution polynomials of degrees $K = 1$ to $K = 5$, and a five-level adaptive simulation, have been performed to validate the utility of the adaptation algorithm. The adaptation parameters K_m , ϵ , and $\vartheta_{1:5}$ are set to 5, 6.25×10^{-3} , and $[0.25, 0.25, 1, 4, 16]$ respectively. The adaptation routine is called every 10 iterations.

6.2.3 Numerical Results

Results are compared in terms of \bar{C}_D , C_{Drms} , mean lift coefficient \bar{C}_L , C_{Lrms} , Strouhal number St , and the mean number of degrees of freedom \overline{DOF} , where the lift and drag coefficients and their mean and rms values are defined as in Equations 7.1 and 6.4. Figure 6.15 shows the polynomial degree distribution for the five-level adaptive simulation ($K = 1 - 5$). As in the previous cases, this verifies that the algorithm successfully tracks elements with large vorticity magnitude relative to their size. A blending function is defined, using Algorithm 3, to limit the mesh movement to within a maximal radius and avoid complete deformation of the domain while the airfoil heaves and pitches, where \mathbf{x} is the location of any point on the mesh, $r_{\mathbf{x}}$ is the distance of the point from the mesh origin, r_{min} and r_{max} are constants that control the mesh deformation, which are defined in a way to prevent zero element volume when the mesh is deformed to its maximum level, f_b is the blending function, \mathbf{v}_g is the local mesh velocity, and \mathbf{v}_f is the velocity induced by the oscillating function. The shape of the deformed mesh and contours of non-dimensional spanwise vorticity are shown in Figure 6.16 for four instants of $0T_e/4$, $1T_e/4$, $2T_e/4$, and $3T_e/4$, which compares qualitative results obtained by this study with the previous work of Moriche et al. [7]. Figure 6.17 shows contours of non-dimensional velocity magnitude at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. While we observe large inter-element jumps in the flow solution of the uniform $K = 1$ simulation, the flow solution is smooth for the uniform $K = 5$ and adaptive $K = 1 - 5$ simulations. Figure 6.18 shows contours of spanwise vorticity at the instant of maximal lift non-dimensionalized by the stream velocity for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. Comparing these plots confirms that the five-level adaptive simulation qualitatively provides satisfactory resolution in the trailing and leading edges, as well as the wake region while requiring fewer degrees of freedom. To further evaluate the accuracy of the adaptive method, we turn to Table 6.4 which reports the

numerical values of \bar{C}_D , C_{Drms} , \bar{C}_L , C_{Lrms} , St , and the mean number of degrees of freedom \overline{DOF} for all simulations, in comparison with the previous study of Moriche et al. [7]. These results show agreement within 0.46%, 0.03%, 0.04%, and 0.32% in the \bar{C}_D , C_{Drms} , \bar{C}_L , and C_{Lrms} values respectively between the adaptive and uniform $K = 5$ simulations, while the former requires 2.45 times fewer degrees of freedom.

```

Compute  $r_x$  for any  $\mathbf{x}$ ;
while  $r_x \neq null$  do
     $f_b = 0.5 \cos((r_x - r_{min})\pi/(r_{max} - r_{min})) + 0.5$ ;
    if  $r \leq r_{min}$  then
         $\mathbf{v}_g = \mathbf{v}_f$ ;
    else if  $r_{min} < r_x \leq r_{max}$  then
         $\mathbf{v}_g = f_b \mathbf{v}_f$ ;
    else
         $\mathbf{v}_g = 0$ ;
    end
end

```

Algorithm 3: Blending function for polynomial adaptation.

Figure 6.19 compares the lift and drag coefficient plots of our uniform $K = 5$ simulation with the B090 case produced by Moriche et al. [7], where the dashed lines show the instants of $t = 1T_e/4 = 196.47$, $t = 2T_e/4 = 197.58$, $t = 3T_e/4 = 198.69$. Comparing these results shows good agreement with the reference data. Furthermore, we observe two peaks of lift coefficient, one positive, and one negative, slightly after $t = 1T_e/4$ and $t = 3T_e/4$ accordingly. These are where the vertical velocity $y'(t)$ and effective angle of attack α_e are maximum.

Table 6.4. Numerical values of \bar{C}_D , C_{Drms} , \bar{C}_L , C_{Lrms} , St , \overline{DOF} , and error percent for the oscillating NACA 0012 at $Re = 1000$ with $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.

Degree	\bar{C}_D	C_{Drms}	\bar{C}_L	C_{Lrms}	St	$\varepsilon_{\bar{C}_D}$	$\varepsilon_{C_{Drms}}$	$\varepsilon_{\bar{C}_L}$	$\varepsilon_{C_{Lrms}}$	\overline{DOF}
$K = 1$	-0.7207	0.9659	1.7715	2.8428	0.2252	0.87	1.83	1.49	0.39	10609
$K = 2$	-0.7110	0.9528	1.7650	2.8610	0.2252	0.49	0.45	1.12	0.25	23661
$K = 3$	-0.7153	0.9509	1.7517	2.8578	0.2252	0.11	0.25	0.36	0.14	41878
$K = 4$	-0.7148	0.9482	1.7463	2.8545	0.2252	0.04	0.03	0.05	0.02	65260
$K = 5$	-0.7145	0.9485	1.7455	2.8538	0.2252	-	-	-	-	93807
$K = 1 - 5$	-0.7112	0.9482	1.7448	2.8446	0.2252	0.46	0.03	0.04	0.32	38198
Moriche B090 [7]	-0.7245	0.9224	1.5507	2.7743	0.2252	-	-	-	-	-

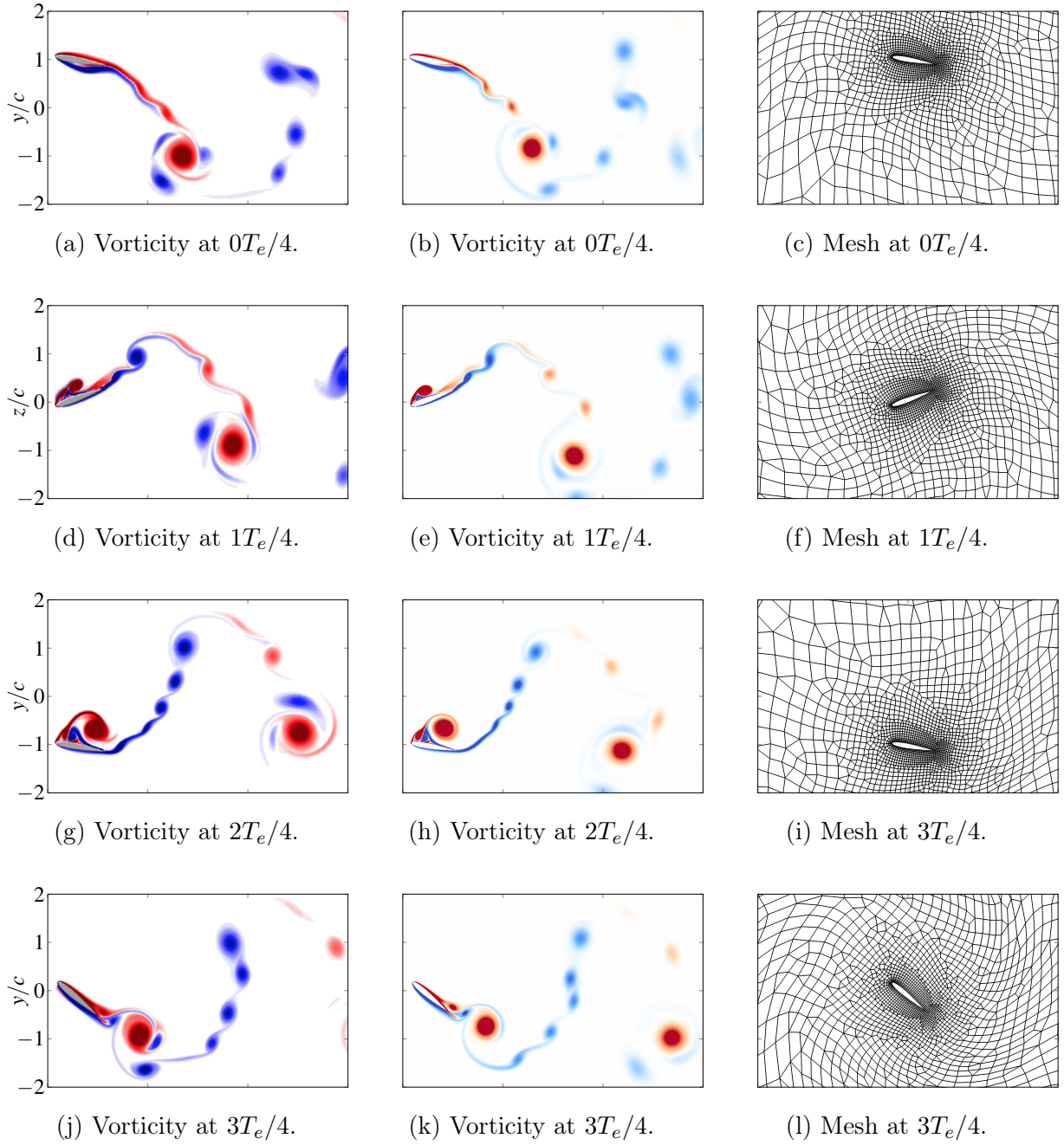
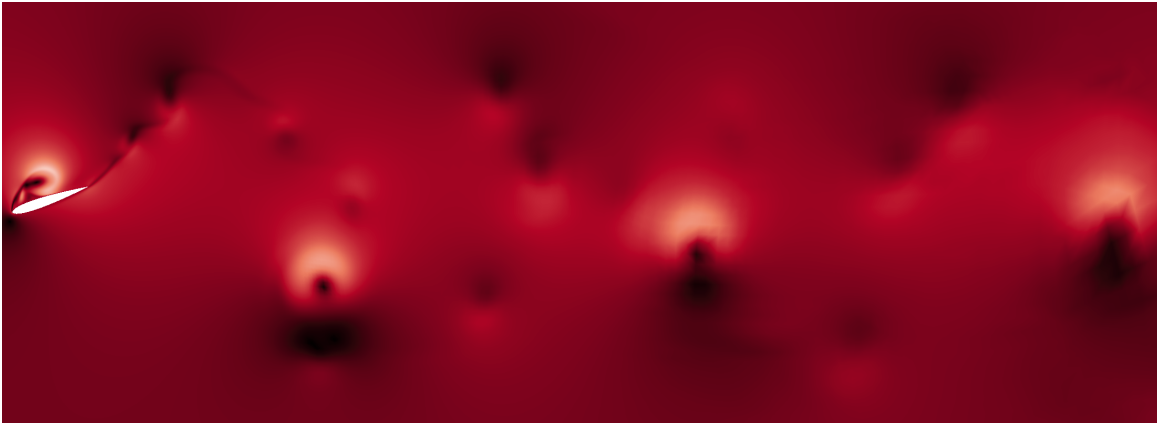


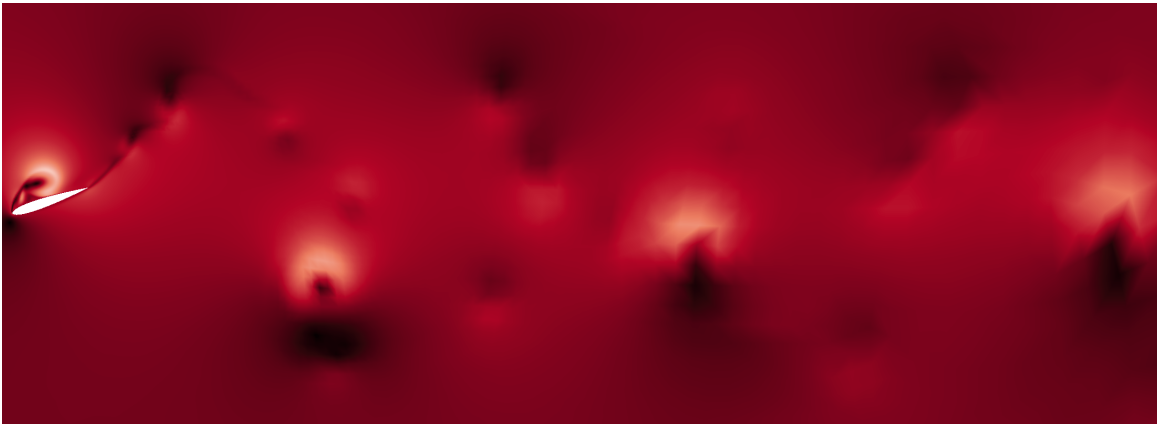
Figure 6.16. Contours of spanwise vorticity of the B090 case from Moriche et al. [7] (a, d, g, and j) [reproduced with permission from Cambridge University Press], of the $K = 5$ deforming mesh NACA 0012 case (b, e, h, and k), along with the mesh deformation (c, f, i, and l) at four time instants.



(a) velocityvelocity field for $K = 1$.



(b) Velocity field for $K = 5$.



(c) Velocity field for $K = 1 - 5$.

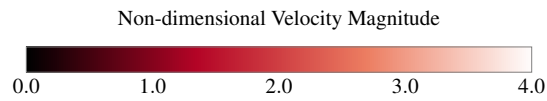


Figure 6.17. Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator for the heaving and pitching NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.



(a) Vorticity field for $K = 1$.



(b) Vorticity field for $K = 5$.



(c) Vorticity field for $K = 1 - 5$.

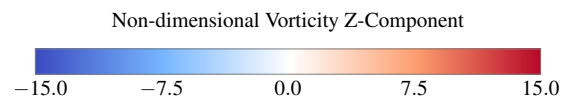


Figure 6.18. Non-dimensional z-component of the vorticity for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on vorticity magnitude indicator for the heaving and pitching NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.

Figure 6.20 compares the lift and drag coefficient plots of uniform and adaptive simulations. In order to understand the physics of dynamic stall and the effect of leading-edge vortex on aerodynamic forces, namely lift and drag, in Figure 6.21, we illustrate contours of spanwise vorticity at several instants of a cycle starting from a minimum lift value, continuing to the maximum lift value, and ending at the next minimum lift value. At the instant a , the lift coefficient is negative since the effective angle of attack is negative $\alpha_e = -14.77^\circ$. The airfoil is moving up to the highest plunge amplitude, which happens at instant d . Through a to d , the airfoil is in the pre-stall region where the lift increases as α_e increases. At instant e , flow reversal begins. Although $\alpha_e = 27.48$ has exceeded the static stall angle of attack, the airfoil has not stalled due to the formation of the LEV. Through instant f to i , where the maximum lift happens, we can observe the LEV contributing to delaying stall and increasing the lift. Furthermore, at instant g , we can also observe the formation of a vortex pair in red. This vortex continues to grow as the airfoil approaches instant i , where it cuts the LEV and causes it to detach from the airfoil. The separation of LEV causes the dynamic stall at instant i , and the lift starts to decrease. The lift decreases dramatically until instant m , where it plateaus because of formation of a second LEV. This vortex slows the lift decrease until instant o . At this moment, a trailing edge vortex starts to form, which causes the lift to decrease with a higher pace until it is shed at about instant r . The lift reaches its minimum value when the secondary LEV has passed the trailing edge at the final instant, which also causes the flow reattachment.

Figure 6.29 illustrates the required DOF by comparing the mean number of degrees of freedom for uniform and adaptive simulations for the oscillating NACA 0012 airfoil, which, again, shows the adaptive simulation consistently requires fewer degrees of freedom for comparable levels of accuracy.

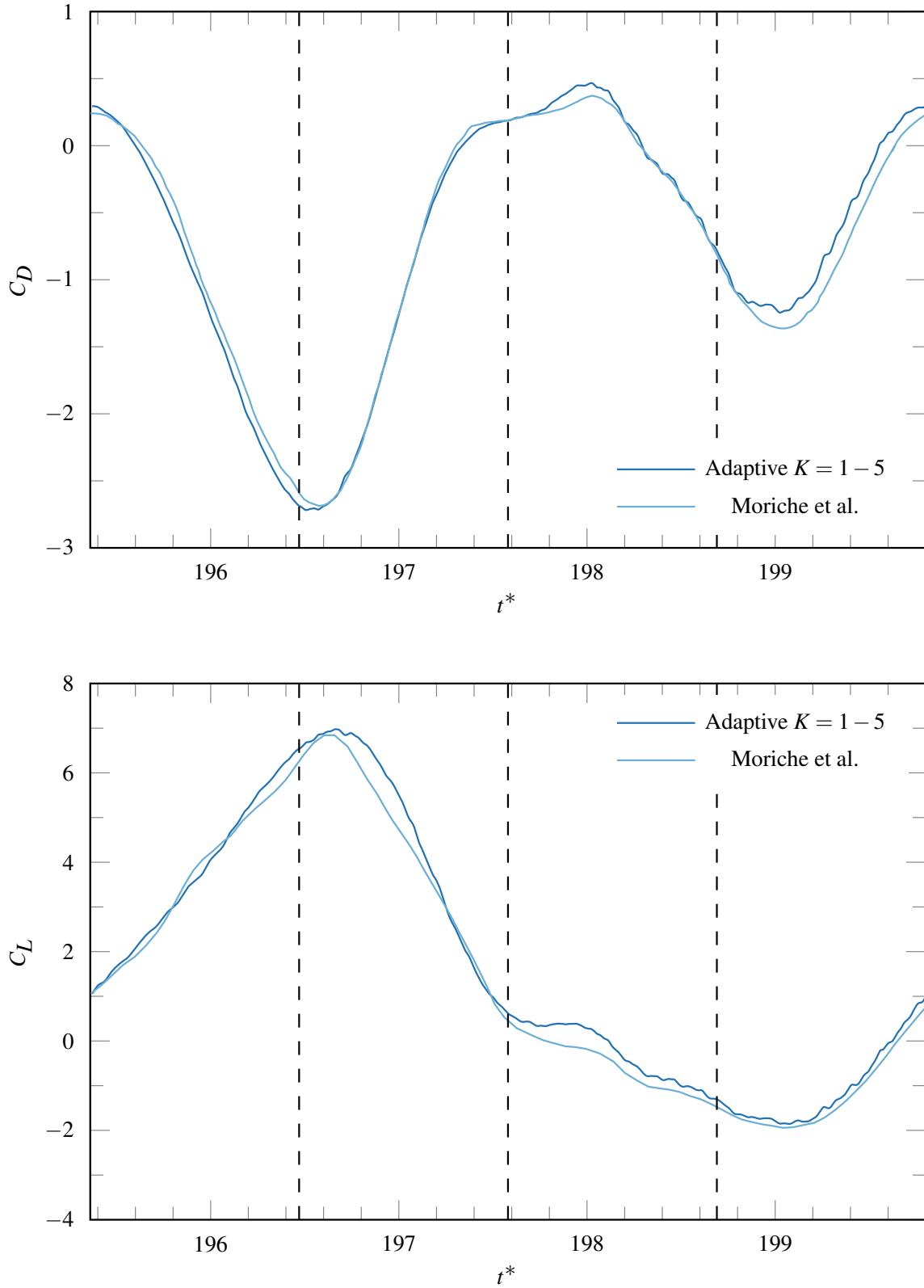


Figure 6.19. Drag and lift coefficient profiles of the uniform $K = 1 - 5$ and the B090 case performed by Moriche et al. [7] for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$. Dashed lines correspond to $t = 1T_e/4$, $t = 2T_e/4$, and $t = 3T_e/4$.

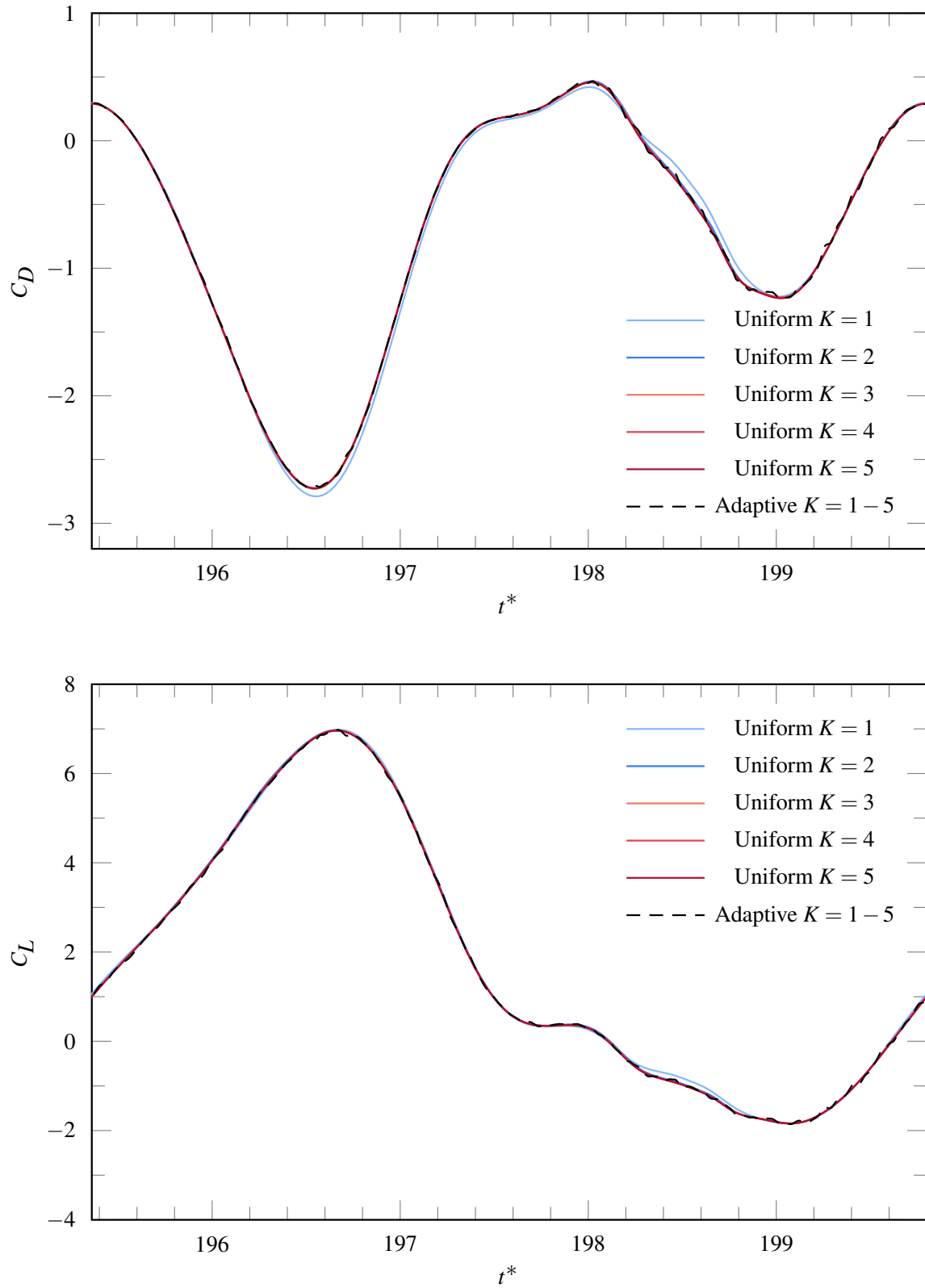
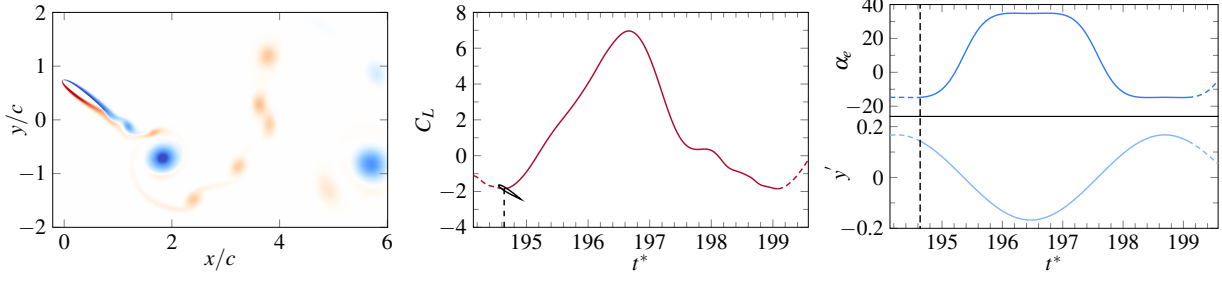
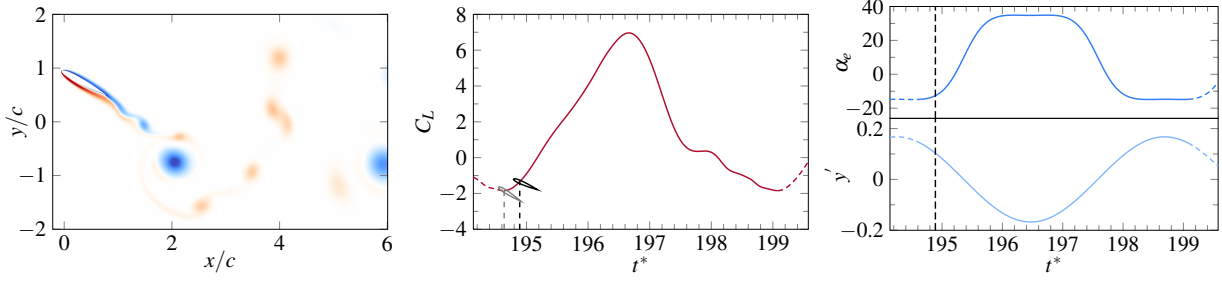


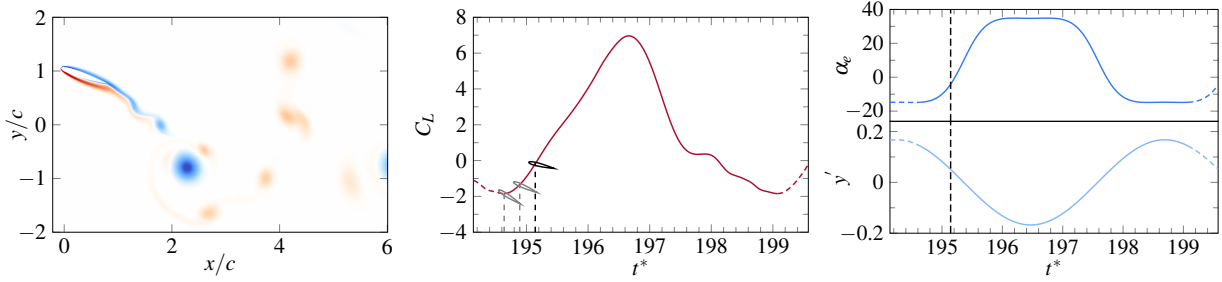
Figure 6.20. Drag and lift coefficient profiles of the uniform and adaptive simulations for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.



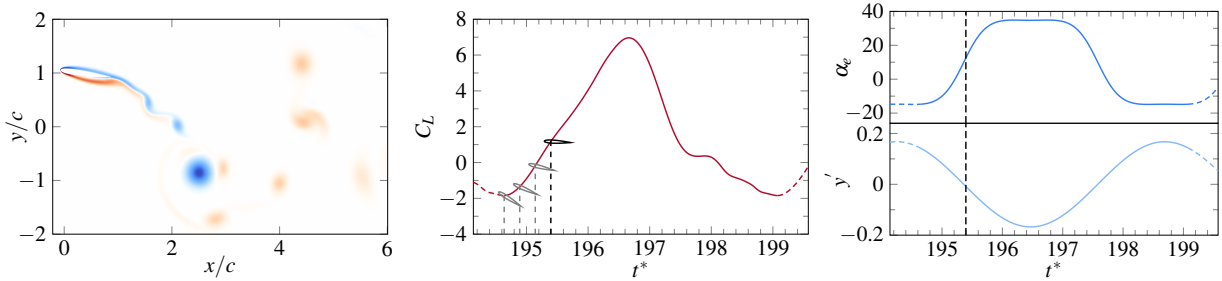
(a) Instant *a*.



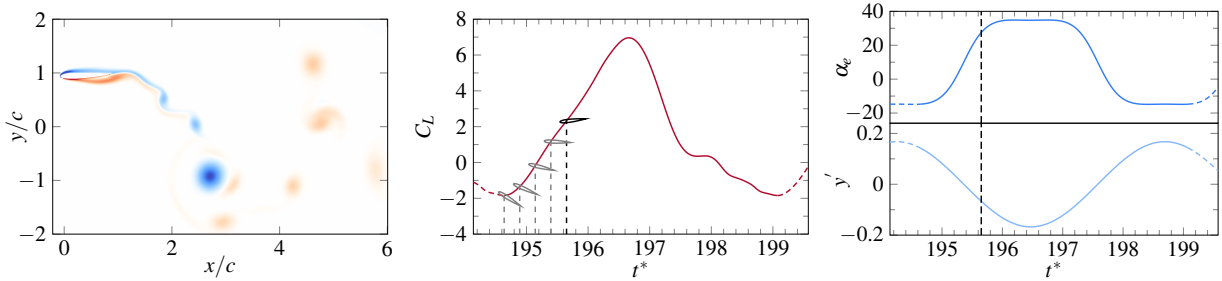
(b) Instant *b*.



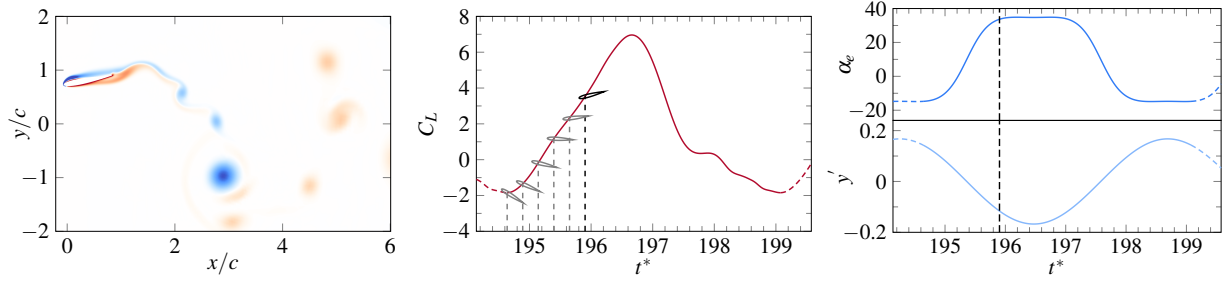
(c) Instant *c*.



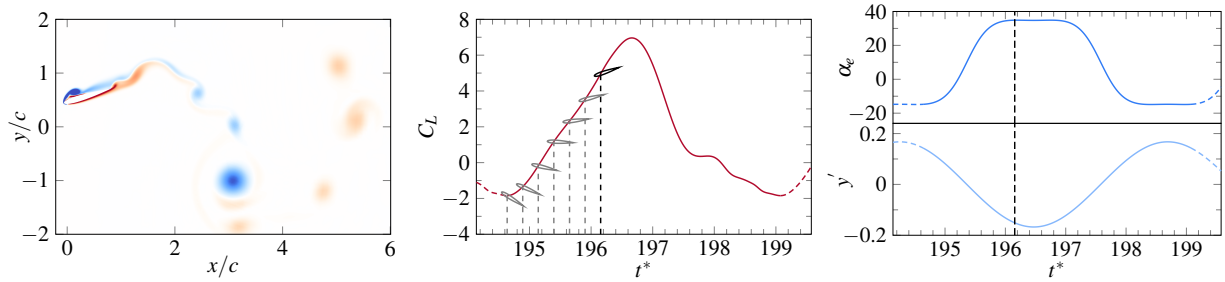
(d) Instant *d*.



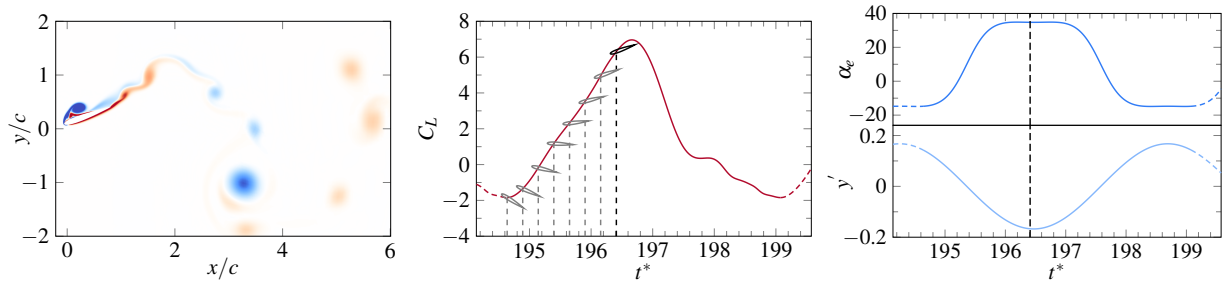
(e) Instant *e*.



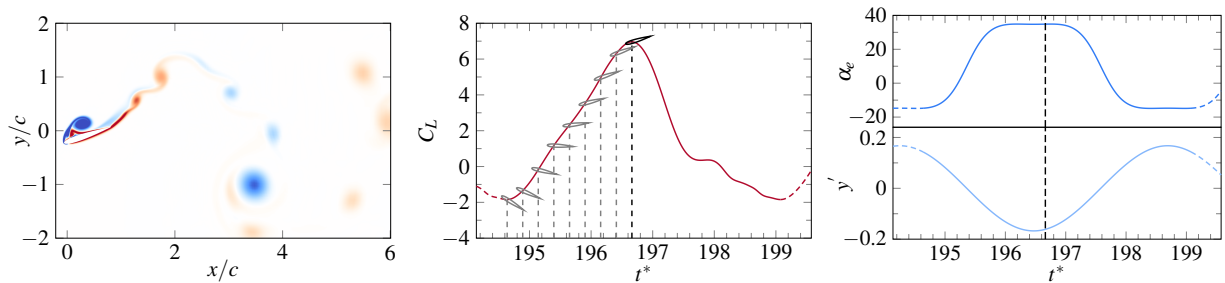
(f) Instant f .



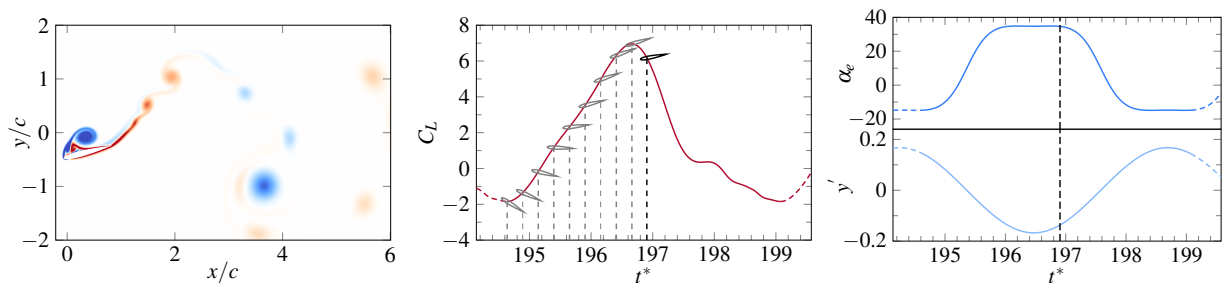
(g) Instant g .



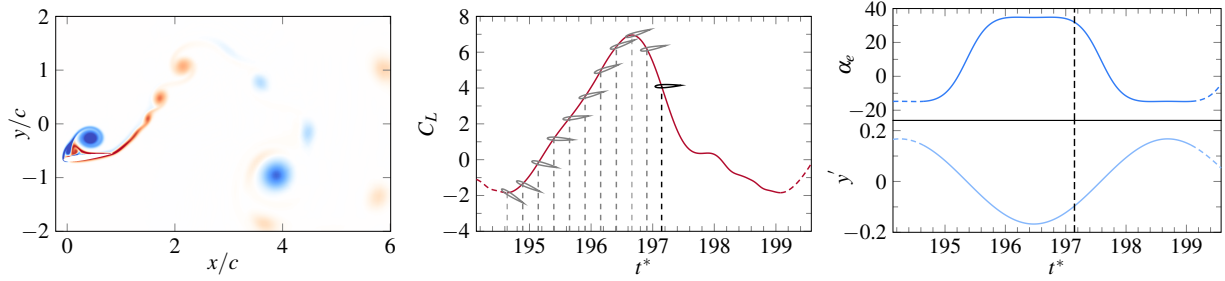
(h) Instant h .



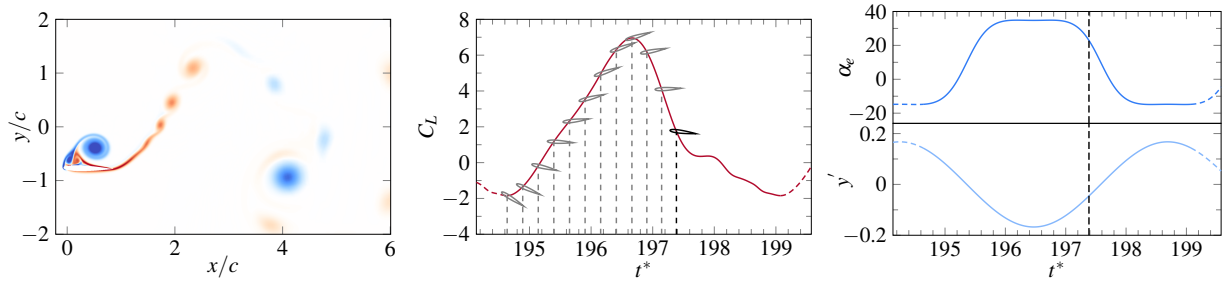
(i) Instant i .



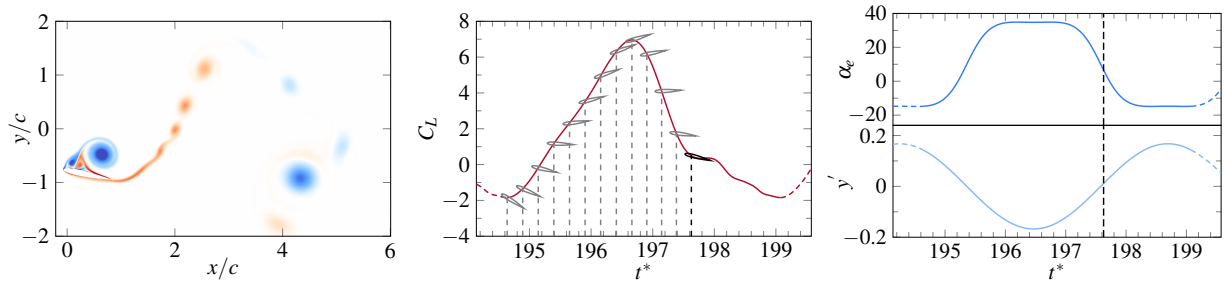
(j) Instant j .



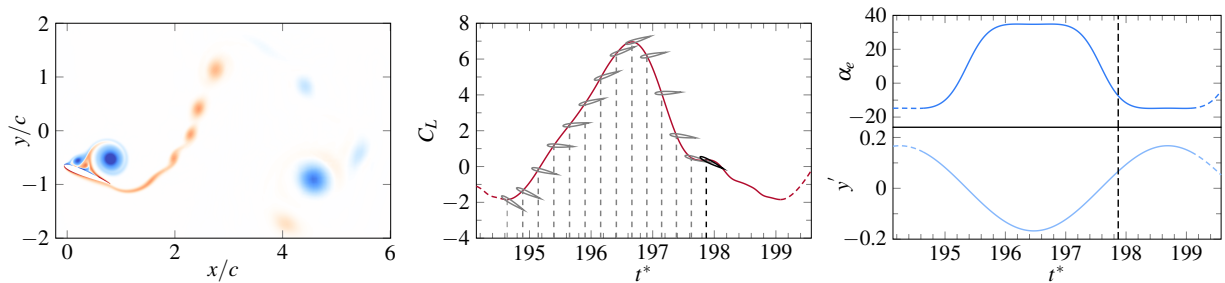
(k) Instant k .



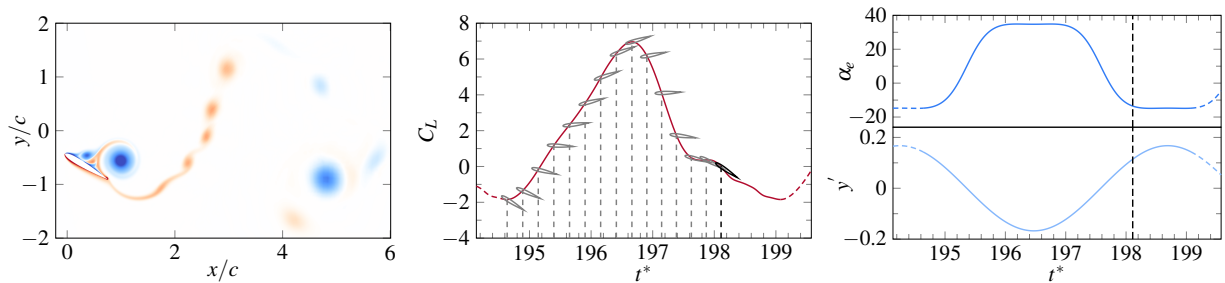
(l) Instant l .



(m) Instant m .



(n) Instant n .



(o) Instant o .

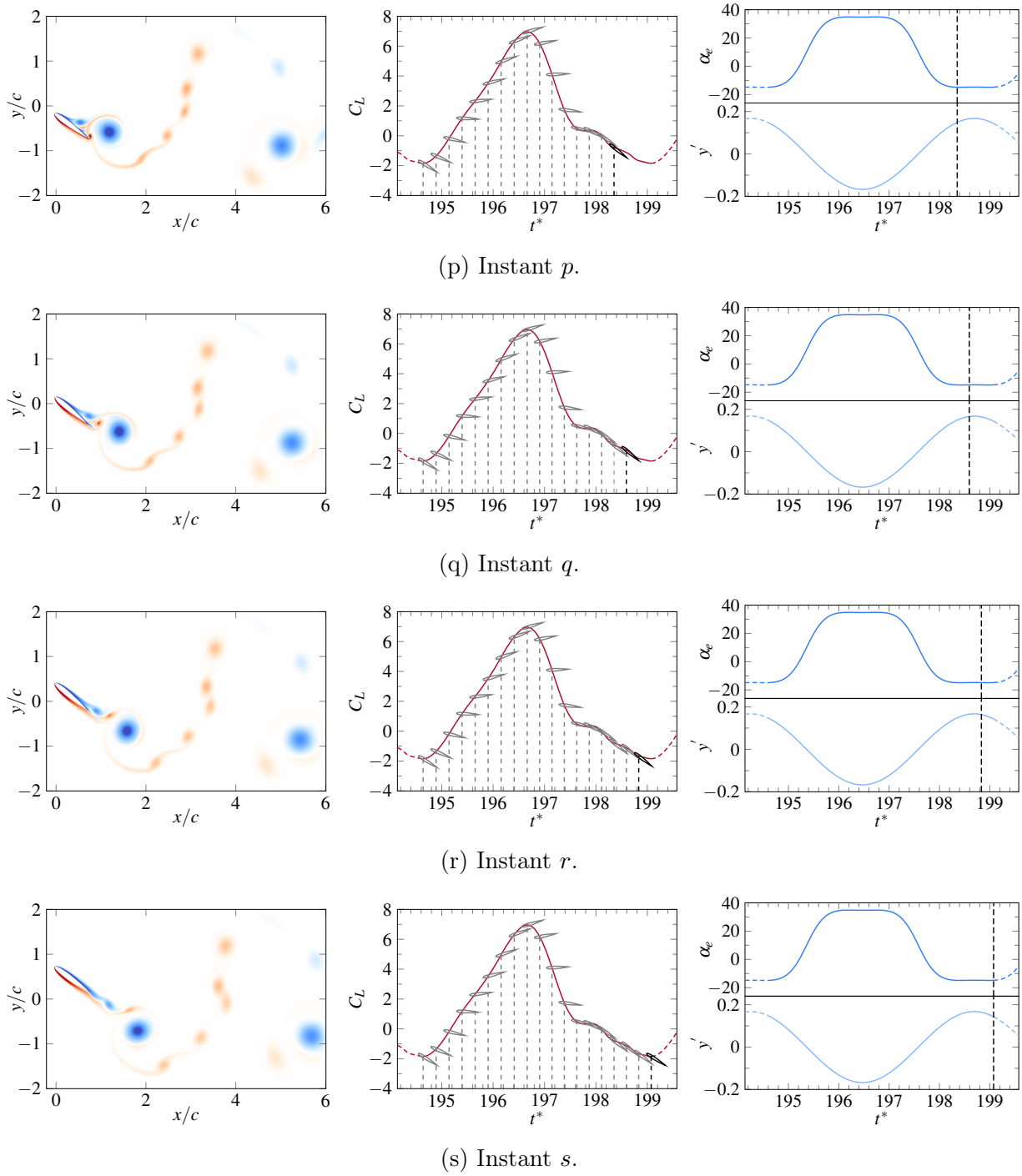


Figure 6.21. Contours of spanwise vorticity (left), lift coefficient (middle), effective angle of attack (top-right), and airfoil vertical velocity (bottom-right) at several time instants for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.

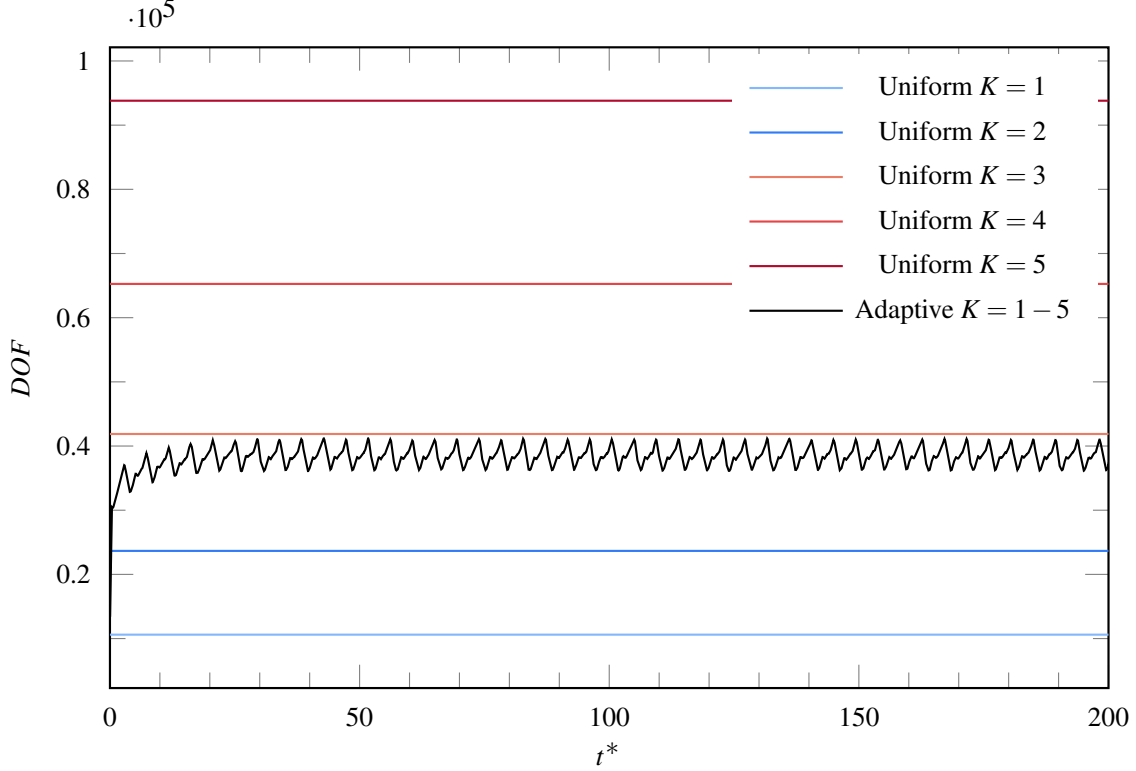


Figure 6.22. Degrees of freedom profile of the uniform and adaptive simulations for the oscillating NACA 0012 airfoil with $Re = 1000$, $A/D = 1$, $T_e = 4.44c/U_\infty$, $\theta_0 = 10^\circ$, $\theta_e = 30^\circ$ and $\phi_e = 90^\circ$.

6.3 2D Vertical Axis Wind Turbine

6.3.1 Introduction

While the previous validation cases demonstrated the ability of the non-dimensional vorticity indicator for relatively simple applications, real-life applications often deal with multiple moving objects. Thus, the vortices shed from one object can interact with others. Consequently, dynamically adapting the entire domain would be vital for accurate numerical results. To demonstrate this, a two dimensional VAWT composed of two NACA 0012 airfoils is considered. This straight-bladed vertical-axis wind turbine was previously built and tested in a tow tank by Strickland et al. [1, 9, 136]. The airfoil chord length and blade tip speed were chosen to yield a blade Reynolds number of $Re_b = 40,000$. Kanner et al. [8] have performed a numerical study based on the same VAWT model. They concluded that their numerical results can accurately approximate the experimental data at higher Tip Speed

Ratios (TSR) [8], λ_b , where λ_b is defined as

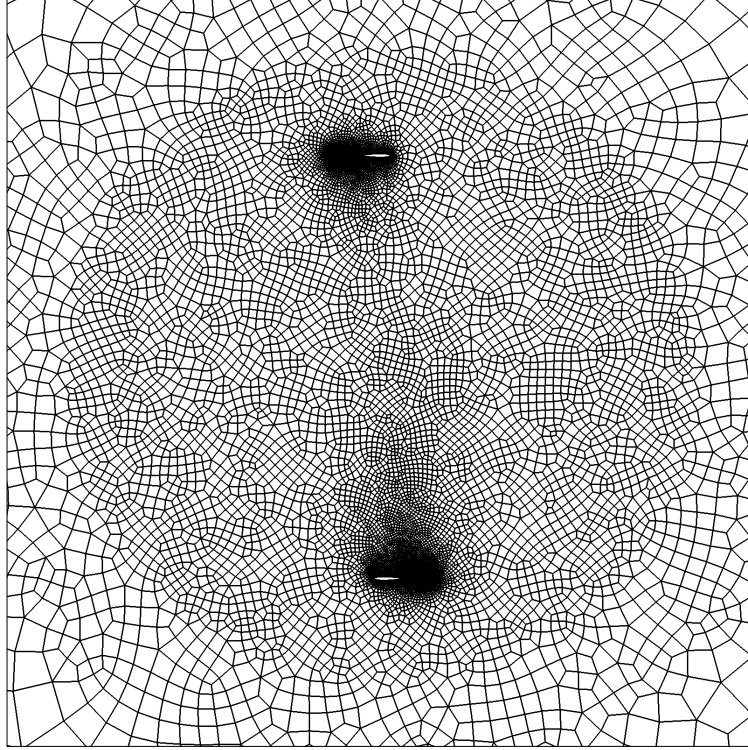
$$\lambda_b = \frac{U_{blade}}{U_\infty}. \quad (6.8)$$

Consequently, among the three different TSR of 2.5, 5, and 7.5 in the experimental study, the TSR of 7.5 is chosen here for our numerical simulation. In the experimental study, the center of pressure x_{cp} of the blades, which is at the quarter-chord of the NACA 0012 airfoils, was located at a radius of $R_b = 0.61m$ from the center of the wind turbine, the chord length of each blade was $c = 9.14cm$, and the blade offset pitch angle was set to $\alpha_0 = 0^\circ$. Kanner et al. [8] used a similar attachment angle of $\alpha_0 = -2$ for their numerical results based on possible alignment uncertainty in the experiments. To compare against the original experimental study of Strickland et al. [1, 9, 136] and the numerical results of Kanner et al. [8], we run two sets of simulations using $\alpha_0 = 0$ and $\alpha_0 = -2$.

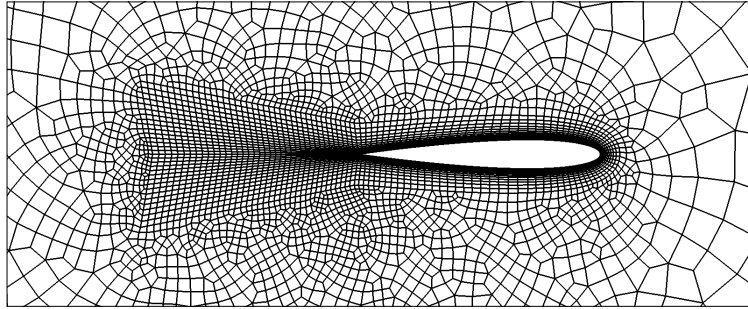
6.3.2 Computational Details

All simulations are carried out at a free-stream Reynolds number of $Re = 5333$, in order to produce a blade Reynolds number of 40,000, when $\lambda_b = 7.5$, and the Mach number is set to $M = 0.025$. A two-dimensional computational domain with a radius of $50c$, centred at the origin of VAWT, consisting of 16081 unstructured quadrilateral elements moderately refined near the blades was used, as shown in Figure 6.23, with Riemann invariant boundary condition at the far-field, periodic boundary condition in the blades span-wise direction, and a no-slip adiabatic wall boundary condition at the surface of blades. Quadratically curved elements were used at the boundaries to match the airfoil geometry. The mesh is initially partitioned over 120 processors using METIS [102], and MPI is used for parallel communication [99].

A ramp-up motion is used at the onset of simulations to increase the angular velocity gradually. The ramp-up angular displacement and angular velocity are defined as $\theta_r(t) = 0.5\hat{\omega}t^2/1000\Delta t^*$ and $\omega_r(t) = \hat{\omega}t/1000\Delta t^*$, respectively. Hence, the angular velocity incrementally reaches the required value in 1000 simulation iterations, where $\hat{\omega}$ is the required VAWT angular velocity, t is time, Δt^* is the non-dimensional time step. Since the experi-



(a) Complete view of the first blade, on the bottom, and the second blade on the top.



(b) Close-up view of the second blade

Figure 6.23. Straight-bladed, NACA0012, Vertical-axis wind turbine unstructured mesh with 16081 quadrilateral elements and $\alpha_0 = 0^\circ$ captured at its first-quarter turn at the time corresponding to $\theta = 90^\circ$.

mental results are reported for a blade at its fourth revolution, the total simulation time is set as the time required for the VAWT to complete 4 cycles. The first cycle starts when the second blade passes $\theta = 270^\circ$ after starting from $\theta_0 = 0^\circ$, and the fourth cycle ends when it passes $\theta = 1710^\circ$. The first three-quarter revolution of the simulation is not considered to avoid errors due to initial transients and the ramp-up motion. Time integration is carried out

with the three-stage third order singly diagonally implicit Runge–Kutta scheme, SDRIK_{3,3}, with a non-dimensional time step of $\Delta t^* = \Delta t U_\infty / c = 2.5 \times 10^{-3}$. As in the previous cases, six simulations, including five uniform simulations with solution polynomials of degrees $k = 1$ to $k = 5$, and a five-level adaptive simulation, have been performed to validate the utility of the adaptation algorithm. The adaptation parameters K_m , ϵ , and $\vartheta_{1:5}$ are set to 3, 6.25×10^{-3} , and $[0.25, 0.25, 1, 4, 16]$ respectively. The adaptation routine is called every 20 iterations.

6.3.3 Numerical Results

Results are compared in terms of sectional tangential force coefficient C_T , power coefficient C_P and the mean number of degrees of freedom \overline{DOF} . The tangential force and power coefficients are defined as

$$\begin{aligned} C_T &= \frac{2F_T}{\rho U_\infty^2 c}, \\ C_P &= N_b \frac{\lambda_b 2T_q}{\rho R_b A_f U_\infty^2} = N_b \frac{\lambda_b \bar{C}_T c}{2R_b}, \end{aligned} \tag{6.9}$$

where $F_T = T_q/R_b$ is the sectional tangential force on the blade, T_q is the torque contribution, $A_f = 2R$ is the total projected frontal area of the rotor, and \bar{C}_T is the mean sectional tangential force coefficient.

Figure 6.24 shows the polynomial degree distribution for the five-level adaptive simulation ($K = 1 - 5$) with $\alpha_0 = 0^\circ$. As in the previous cases, this verifies that the p -adaptation algorithm successfully tracks elements with large vorticity magnitude relative to their element size in the boundary layer and wake region of each blade, as well as the downwind zone, where the shed vortices from the blade in the upwind zone will interact with the downstream blade. Figure 6.25 shows contours of non-dimensional spanwise vorticity in the fourth cycle, for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. Comparing these results confirms that the five-level adaptive simulation qualitatively provides satisfactory resolution both in the upwind and downwind zones, as well as the wake region of each blade, while requiring fewer degrees of freedom.

Figure 6.26 compares the tangential force coefficient of the uniform $K = 5$ simulation with $\alpha_0 = -2^\circ$ while the second blade makes its first revolution from $270^\circ \leq \theta \leq 630^\circ$ after

starting from $\theta_0 = 0^\circ$, against the numerical results of Kanner et al. [8], while Figure 6.27 compares the tangential force coefficient of the uniform $K = 5$ simulation with $\alpha_0 = 0^\circ$ when one of the blades completed its fourth revolution, against results from the experimental data [1, 9, 136]. Based on phase correction reported by Strickland et al. [1] and Kanner et al. [8], the experimental data has been shifted accordingly. The numerical results at $\alpha_0 = -2^\circ$ are almost identical, while Figure 6.27 shows a good agreement between the numerical and experimental results $\alpha_0 = 0^\circ$. Strickland performed three experimental measurements of the two-bladed wind turbine at a TSR of 7.5 [1], where the plots of tangential force coefficient versus the rotor angle are included in Appendix B for all three runs. These plots show that due to the complexity of the flow, a small perturbation in the initial conditions could result in a slightly different force prediction in the subsequent revolutions.

Table 6.5 reports the numerical values of \bar{C}_T , C_P , and \overline{DOF} averaged over the fourth cycle for all uniform simulations and the five-level adaptive simulation alongside analytical results of Strickland et al. [136]. These results quantitatively show agreement within 5.89% and 5.90% in the C_T and C_P values respectively between the adaptive and uniform $K = 5$ simulations, while the adaptive simulation requires 1.75 times fewer degrees of freedom. Furthermore, the adaptive simulation quantitatively agrees with the analytical results within 8.57% in the C_P value. Figure 6.28 compares the tangential force coefficient plots of the uniform and adaptive simulations, which shows good agreement between the uniform $K = 5$ and adaptive $K = 1 - 5$ simulations illustrating the capability of the adaptive simulations to converge to a full high order result while requiring fewer degrees of freedom. To clarify the difference between different polynomial degrees, the high-frequency noise is filtered from the plots shown in Figure 6.28. Finally, Figure 6.29 compares the number of degrees of freedom of the uniform and adaptive simulations for the vertical-axis wind turbine over four cycles, which shows the adaptive simulation consistently requires fewer degrees of freedom for comparable levels of accuracy.

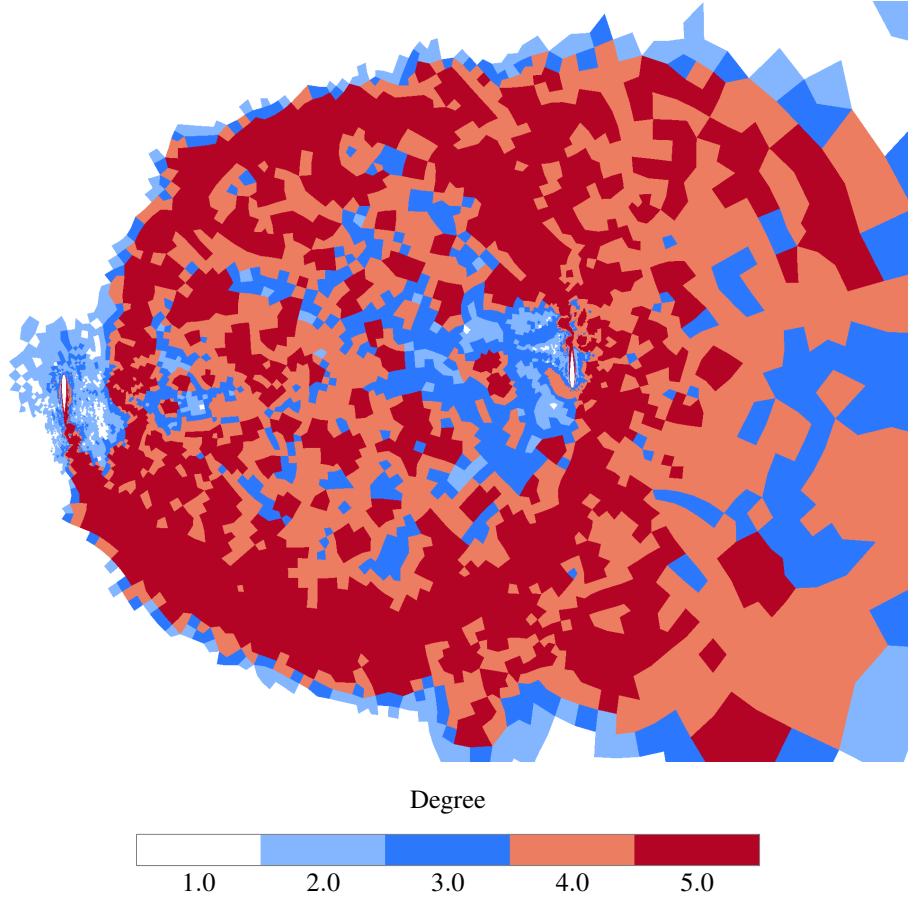
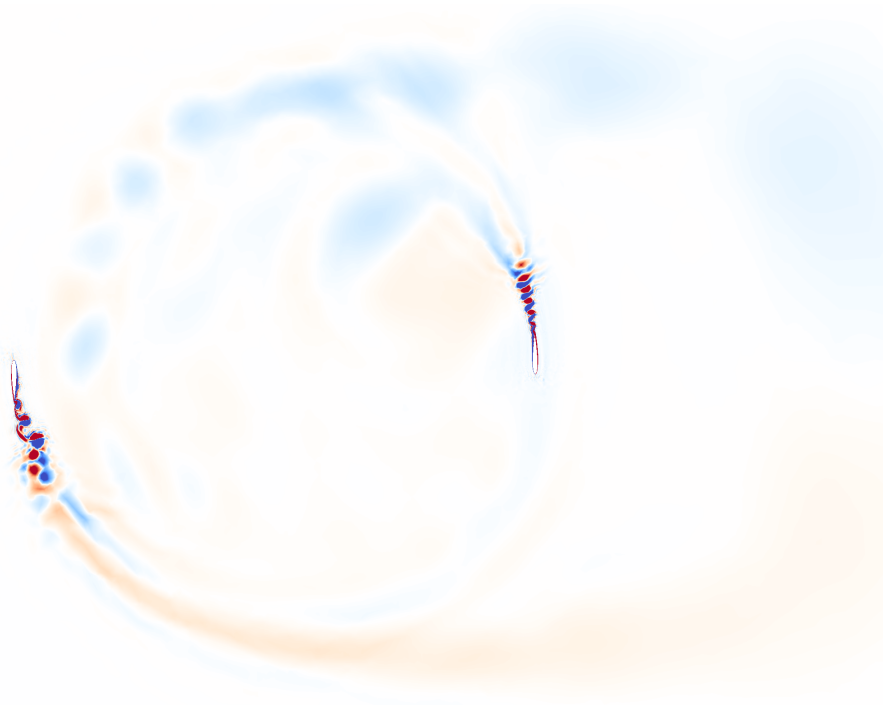


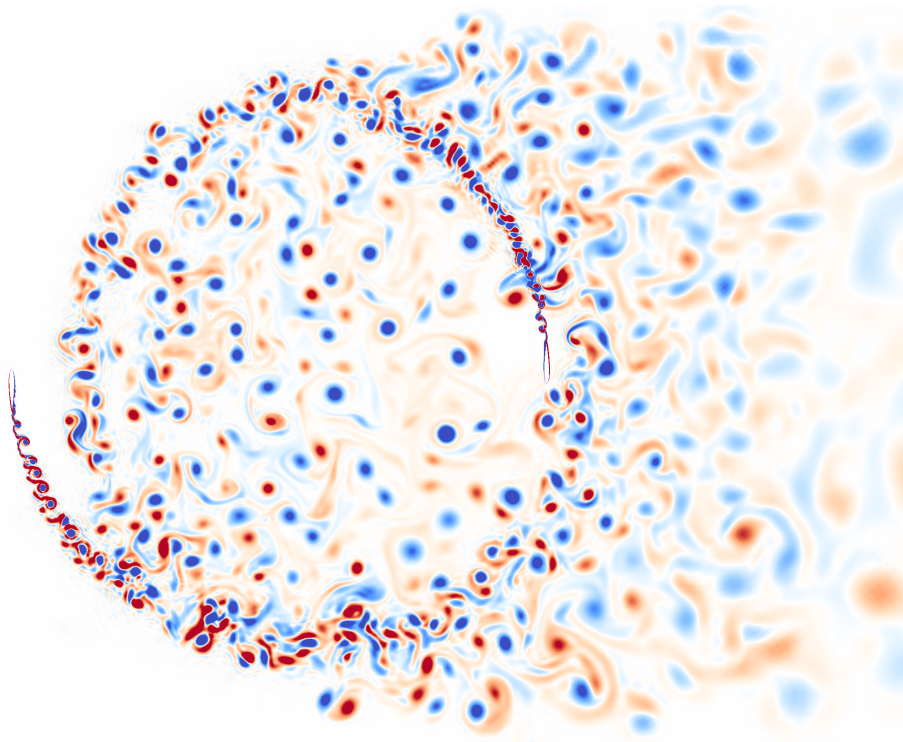
Figure 6.24. Polynomial distribution for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator in the 4th revolution at the time corresponding to $\theta = 1620^\circ$ for the vertical-axis wind turbine composed of two NACA 0012 blades.

Table 6.5. Numerical values of C_T , C_P , and \overline{DOF} averaged over the fourth cycle with related error percent for the vertical-axis wind turbine composed of two NACA 0012 blades and $\alpha_0 = 0$.

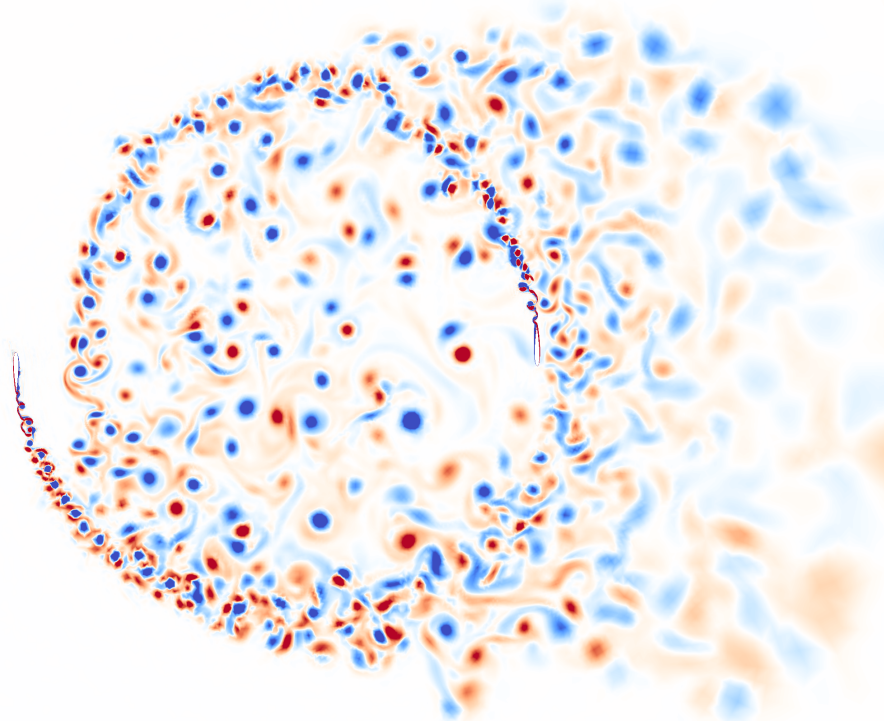
Degree	C_T	C_P	$\varepsilon_{\bar{C}_T}$	ε_{C_P}	\overline{DOF}
$K = 1$	-1.1095	-1.2468	56.73%	56.73%	64324
$K = 2$	-0.8586	-0.9648	21.29%	21.28%	144729
$K = 3$	-0.6845	-0.7692	3.31%	3.31%	257296
$K = 4$	-0.6960	-0.7821	1.68%	1.68%	402025
$K = 5$	-0.7079	-0.7955	-	-	578916
$K = 1 - 5$	-0.7496	-0.8424	5.89%	5.90%	331648
Strickland et al. [136]	-	-0.7759	-	-	-



(a) Vorticity field for $K = 1$.



(b) Vorticity field for $K = 5$.



(c) Vorticity field for $K = 1 - 5$.

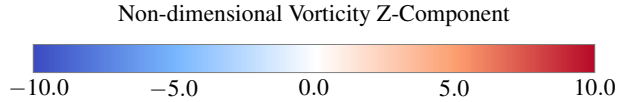


Figure 6.25. Non-dimensional z-component of the vorticity for the uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$ computations based on the vorticity magnitude indicator in the 4th revolution at the time corresponding to $\theta = 1620^\circ$ for the vertical-axis wind turbine composed of two NACA 0012 blades.

6.4 Conclusion

We have explored the utility of our novel non-dimensional vorticity-based indicator for p -adaptation on moving and deforming domains. This algorithm was validated for two classical problems, specifically, a pair of oscillating circular cylinder configurations and a dynamic stall of a heaving and pitching NACA 0012 airfoil. Results demonstrated that the adaptation algorithm is capable of tracking salient flow features on deforming domains, such as the boundary layer and unsteady wake regions. Qualitative results showed equivalent levels of accuracy between the adaptive and high-order solutions, with a significant reduction in the total number of degrees of freedom.

Finally, a practical test case of flow over a VAWT was considered. As in other test cases

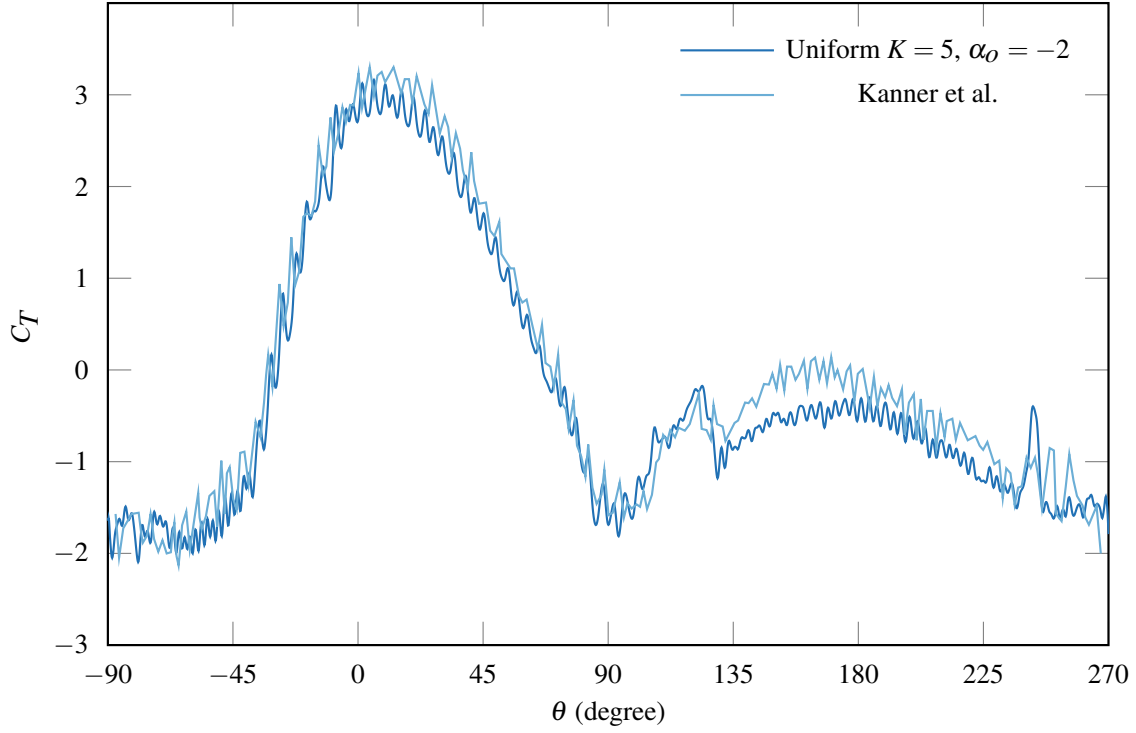


Figure 6.26. Tangential force coefficient profiles corresponding to the 1st cycle of the uniform $K = 5$ simulation and the previous numerical study performed by Kanner et al. [8], for the second blade of the vertical-axis wind turbine with $\alpha_0 = -2$.

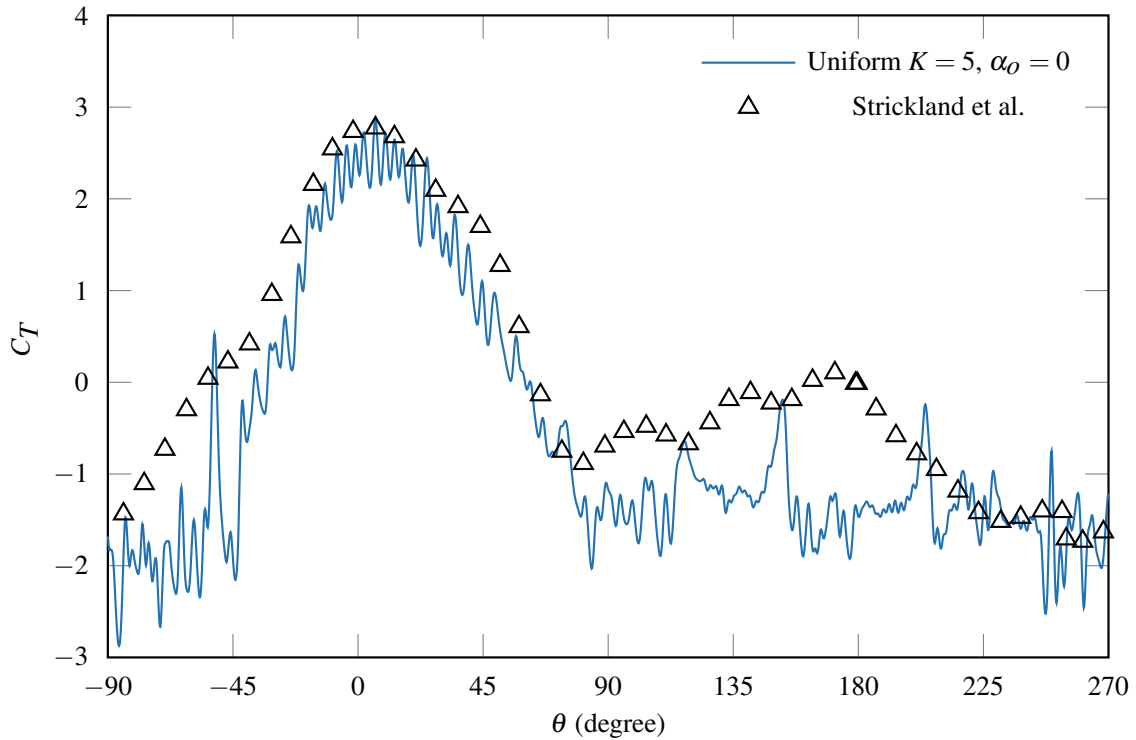


Figure 6.27. Tangential force coefficient profiles corresponding to the 4th cycle of the uniform $K = 5$ simulation and the previous experimental study performed by Strickland et al. [1, 9], for a blade of the vertical-axis wind turbine with $\alpha_0 = 0$.

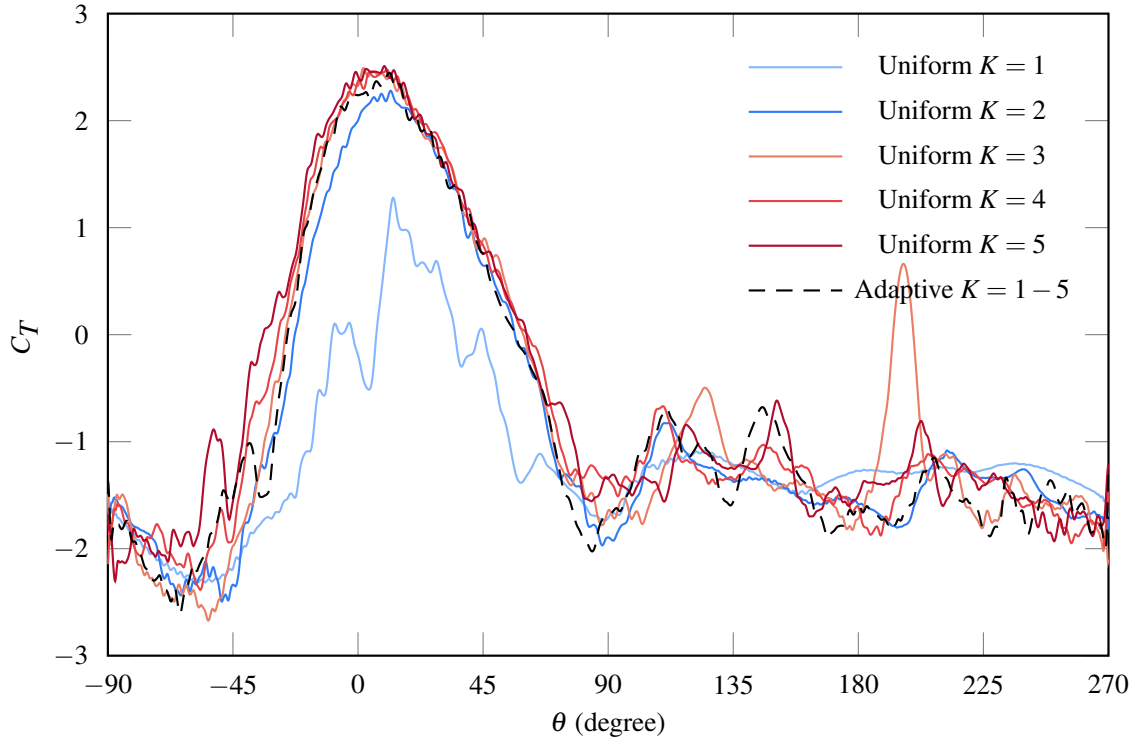


Figure 6.28. Tangential force coefficient profiles of the 4th cycle of the uniform and adaptive simulations for the vertical-axis wind turbine composed of two NACA 0012 blades.

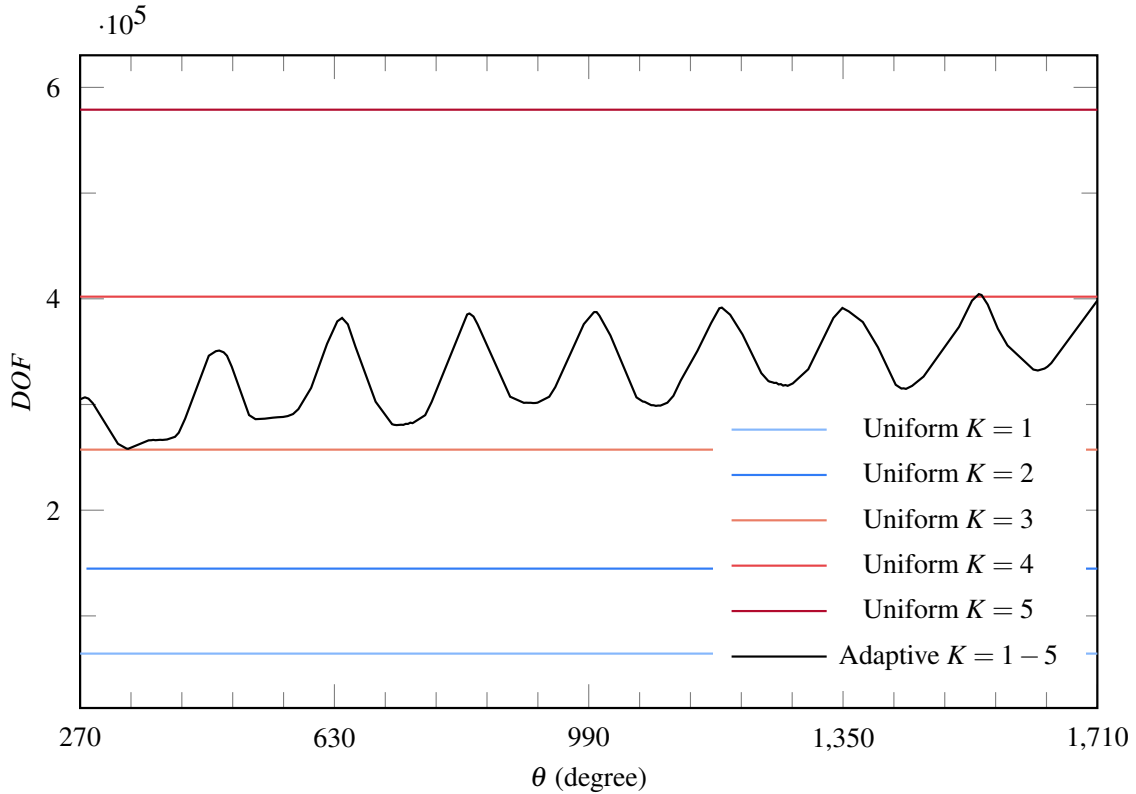


Figure 6.29. Degrees of freedom profile over four cycles of the uniform and adaptive simulations for the vertical-axis wind turbine composed of two NACA 0012 blades.

it was observed that the adaptive simulations were able to track the boundary layer and wake regions of each blade, and their subsequent interactions. Results from the adaptive simulations also showed good quantitative agreement with parallel high-order simulations and reference numerical, experimental, and analytical results.

Hence, the proposed non-dimensional vorticity-based adaptation indicator is a simple, effective, and accurate approach for performing p -adaptation on moving and deforming domains.

The next chapter will explore the extension of this indicator to three-dimensional problems using LES, by performing dynamically load-balanced polynomial adaptation of transitional and turbulent flows.

Chapter 7

Dynamically Load Balanced Polynomial Adaptation of Turbulent Flows

The objective of this chapter is to further validate the efficiency of the load-balanced vorticity-based adaptation algorithm when applied to three-dimensional unsteady transitional and turbulent flows. Three-dimensional simulations typically require relatively a larger number of degrees of freedom increasing the overhead caused by polynomial adaptation, hence p -adaptation performance will be significantly decreased unless a DLB algorithm is applied to repartition the mesh at run time. In this chapter, we use ILES without an explicit SGS model, since, as explained earlier, it relies on the dissipation error of the numerical scheme to dissipate the kinetic energy from the smallest turbulent scale in the flow. The rest of this chapter is organized as follows. In Section 7.1, we validate the algorithm by studying shallow dynamic stall of a three-dimensional SD 7003 airfoil undergoing heaving and pitching motions. We then continue validation of the algorithm in the existence of turbulent transition and turbulent wake flow by considering a three-dimensional circular cylinder at the onset of the shear-layer transition regime in Section 7.2. Finally, we present conclusions in Section 7.3.

7.1 3D Dynamic Stall of a SD 7003 Airfoil

7.1.1 Computational Details

In order to further verify the performance of the adaptation algorithm, a 3D SD 7003 airfoil undergoing heaving and pitching motions at a moderately high Reynolds number is studied. A three-dimensional computational domain, with its origin at the leading edge, consisting of 56000 structured hexahedral elements, moderately refined near the trailing and leading edges, was used with Riemann invariant boundary condition at the far-field, periodic boundary condition in the span-wise direction, and a no-slip adiabatic wall boundary condition at the surface of the airfoil. The domain extends to $19c$ upstream, $20c$ above, below, and downstream of the airfoil, and $0.4c$ in the span-wise direction, as shown in Figure 7.1, where c is the airfoil chord. Quadratically curved elements were used at the boundaries to match the airfoil geometry. The mesh is initially partitioned over 120 processors using METIS [102], and MPI is used for parallel communication [99].

The translation of the coordinates of the center of oscillation and the pitching function are defined as in Equation 6.6. Simulation were carried out at $Re = 10000$, $M = 0.1$, $A/c = 0.5$, $f_K = 0.25$ resulting in a period of oscillation of $T_e = 4\pi/U_\infty$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$, and $\phi_e = 90^\circ$ in order to match the experimental study of Baik et al. [10] and Ol et al. [137], who studied shallow and deep stall of SD 7003 airfoil in nominally two-dimensional conditions for a range of Reynolds numbers. Shallow dynamic stall happens when the airfoil undergoes pitching and heaving motions. On the other hand, deep dynamic stall happens when the airfoil undergoes a pure heaving motion resulting in a relatively stronger leading-edge vortex. The total simulation time is set to $8T_e$. This corresponds to $\approx 125t_c$, where, as in the previous section, t_c is the time required for the flow to traverse one airfoil chord. Time integration was with the classical $RK_{4,4}$ scheme, with a non-dimensional time step of $\Delta t^* = 6.14 \times 10^{-5}$, which is a fraction of the oscillation period. A three-level adaptive simulation was performed to verify the utility of the adaptation algorithm and load balancing, which was achieved via the adaptive repartitioning algorithm in ParMETIS [95]. The adaptation parameters K_m , ϵ , and $\vartheta_{1.5}$ are set to 3, 2.5×10^{-1} , and $[0.25, 0.25, 1]$, respectively. The adaptation and the DLB routines are called every 100 and 1000 iterations respectively.

7.1.2 Numerical Results

We are interested in showing the efficiency of the load-balanced adaptation algorithm when applied to three-dimensional test cases. Contours of non-dimensional velocity magnitude are shown in Figure 7.5 for four instants of $0T_e/4$, $1T_e/4$, $2T_e/4$, and $3T_e/4$. Results show that the adaptive simulation provides satisfactory resolution in the trailing and leading edges, as well as the wake region. Figure 7.9 shows three-dimensional contours of the Q-criterion colored by the velocity magnitude at the time instant of $1T_e/4$. The Q-criterion is the second invariant of the velocity gradient tensor $\nabla \mathbf{v}$ used to visualize vortical structure and is defined as $Q = 1/2(\|\bar{\bar{\boldsymbol{\Omega}}}\|^2 - \|\bar{\bar{\mathbf{S}}}\|^2)$ [138], where $\|\mathbf{S}\| = \text{tr}(\mathbf{S}\mathbf{S}^T)^{1/2}$, $\|\boldsymbol{\Omega}\| = \text{tr}(\boldsymbol{\Omega}\boldsymbol{\Omega}^T)^{1/2}$, $\bar{\bar{\boldsymbol{\Omega}}} = 1/2(\partial v_i/\partial x_j - \partial v_j/\partial x_i)$ is the antisymmetric component of $\nabla \mathbf{v}$, known as the vorticity tensor, and $\bar{\bar{\mathbf{S}}} = 1/2(\partial v_i/\partial x_j + \partial v_j/\partial x_i)$ is the symmetric component of $\nabla \mathbf{v}$, known as the strain rate tensor. Hence it represents a balance between the strain rate and the vorticity magnitude in the flow. Figure 7.4 shows contours of Q-criterion colored by the velocity magnitude and the polynomial degree distribution for the three-level load-balanced simulation ($K = 1 - 3$) for four instants of $0T_e/4$, $1T_e/4$, $2T_e/4$, and $3T_e/4$. As in the previous cases, this verifies that the algorithm successfully tracks elements with large vorticity magnitude relative to their size. It is evident that most of the $K = 3$ elements are employed where the vorticity magnitude is maximal relative to the element size, which illustrates high-order elements track vorticity behind the airfoil. The $K = 2$ elements occur further downstream, out of the wake region, leaving the far-field region for $K = 1$ elements. Figure 7.5 shows the non-dimensional x-component of the velocity profile averaged through the span and four last cycles for three instants of $0T_e/4$, $1T_e/4$, and $2T_e/4$ at chord locations of $x/c = 0.125$, $x/c = 0.25$, $x/c = 0.5$, $x/c = 0.75$, $x/c = 1.0$, and $x/c = 1.25$, where $x/c = 0$ corresponds to the initial x-coordinate of the leading edge, which is the mesh origin. $y/c=0$ corresponds to the y-coordinate of the trailing edge when $x/c = 1$ and $x/c = 1.25$, while corresponds to the airfoil surface for all other chord locations. Results are compared with the phase-averaged Particle Image Velocimetry (PIV) velocity field data obtained by Baik et al. [10]. Comparing these plots confirms that the three-level adaptive simulation correlates closely with the experiment results.

As mentioned in Chapter 4, p -adaptation leaves the computational domain fairly unbal-

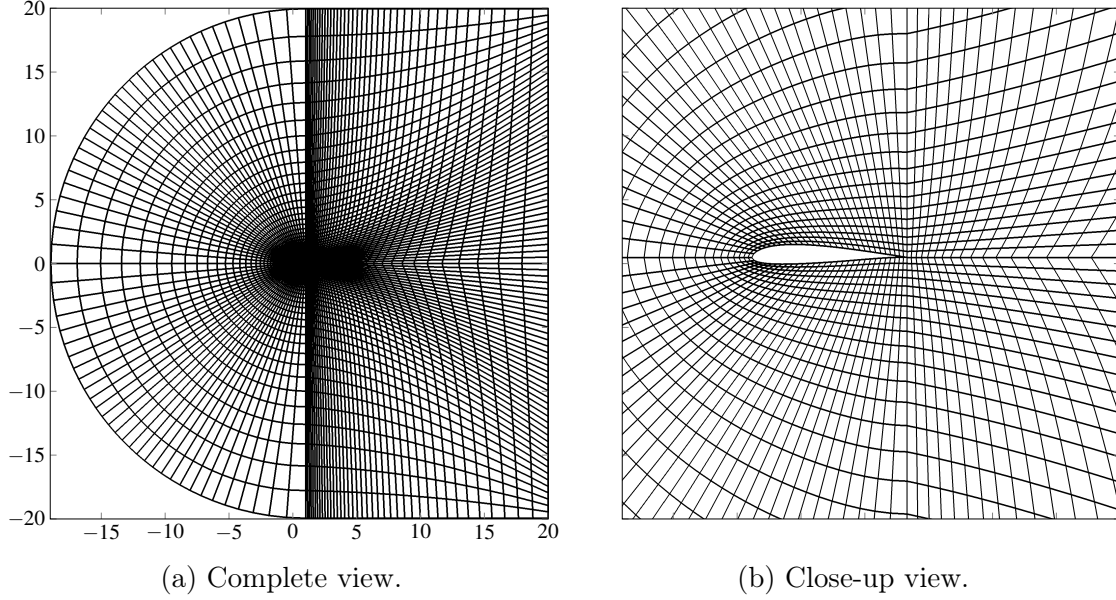


Figure 7.1. The SD 7003 mesh with 56000 hexahedral elements. Dimensions are given in multiples of the airfoil chord length.

anced causing a significant overhead. This overhead is typically larger for a three-dimensional simulations since they require relatively larger numbers of degrees of freedom, necessitating DLB. To validate the efficiency of the DLB algorithm, a set of uniform and parallel simulations were carried out on one node of Niagara cluster consisting 2 sockets with 20 Intel Skylake cores (2.4GHz, AVX512), for a total of 40 cores and a total of 202 GB of RAM. Table 7.1 shows the simulation time per one iteration T_{itr} and the speed-up factor $S_A = T_U/T_A$ for uniform and the three-level adaptive simulations. The speed-up factor is calculated based on the uniform $K = 3$ simulation with the same mesh resolution, where T_A is the compute time of an adaptive $K = 1 - 3$ simulation featured with DLB, and T_U is the compute time of a uniform $K = 3$ simulation. The speed-up factor is 2.73 which makes the adaptive simulation even faster than a uniform $K = 2$ simulation. It is worth mentioning that the adaptation speed-up has a lower limit of 1 and an upper limit of $T_{U,K_m}/T_{U,K_{min}}$, where T_{U,K_m} is the compute time of a uniform $K = K_m$ simulation, and $T_{U,K_{min}}$ is the compute time of a uniform $K = K_{min}$ simulation. Hence considering that the uniform $K = 1$ simulation is only 4.57 faster than the uniform $K = 3$ simulation, the speed-up factor of 2.73 is quite reasonable.

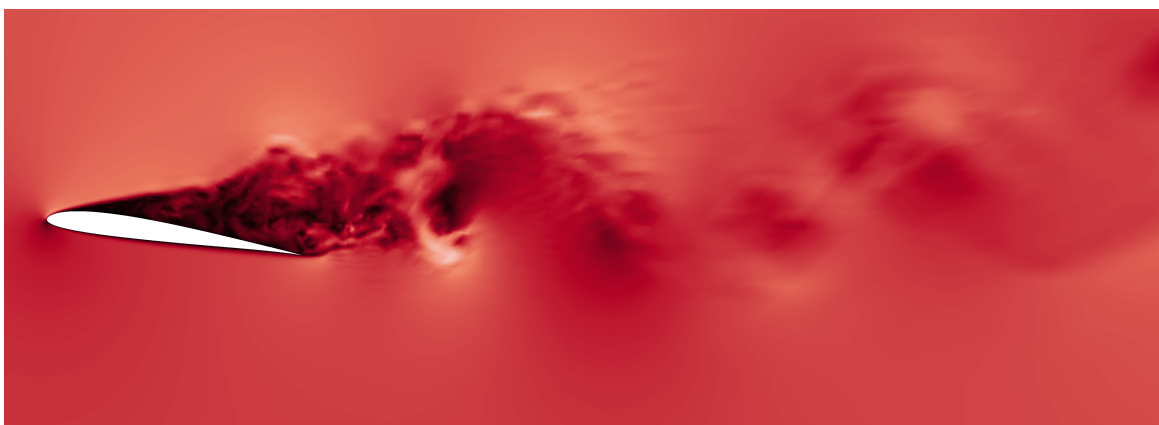
To validate the scalability of the DLB algorithm, a set of parallel simulations were carried out on Niagara with different numbers of nodes. Each node consists of 2 sockets with 20 Intel



(a) Velocity at $0T_e/4$.



(b) Velocity at $1T_e/4$.



(c) Velocity at $2T_e/4$.



(d) Velocity at $3T_e/4$.

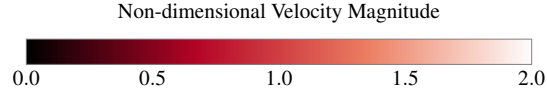


Figure 7.2. Non-dimensional velocity magnitude at different time instants for the adaptive dynamic load-balanced heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.

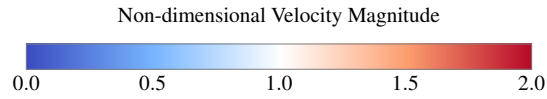
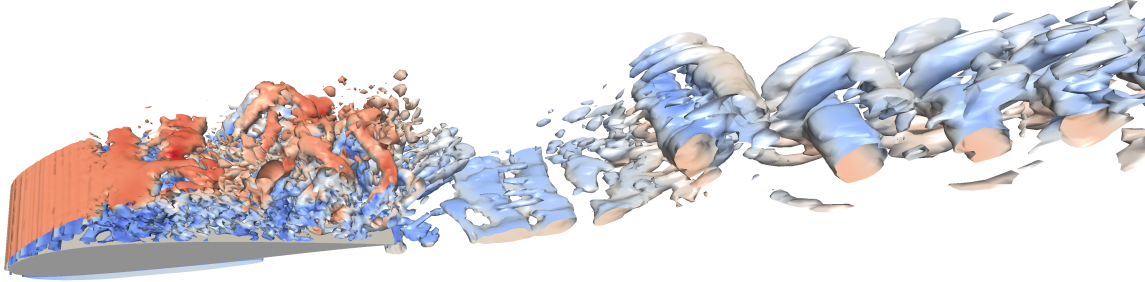
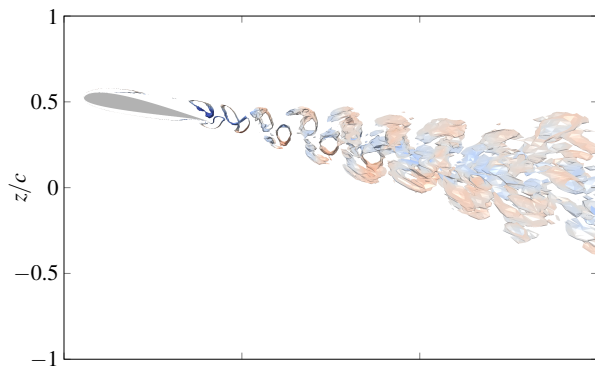
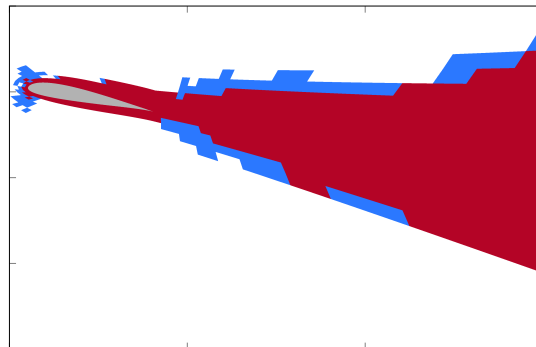


Figure 7.3. Contours of Q-criterion colored by velocity magnitude at the time instant of $1T_e/4$ for the adaptive dynamic load-balanced heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.

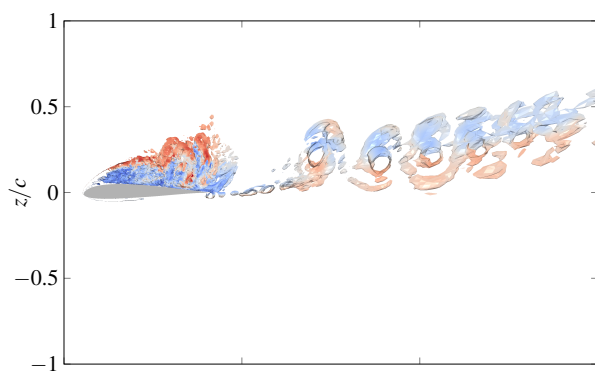
Skylake cores (2.4GHz, AVX512), for a total of 40 cores per node, and a total of 202 GB of RAM. Table 7.2 shows the scalability of the parallel algorithm for different numbers of cores, where N_n is the number of nodes, C is the number of cores, $S_N = T_N/T_C$ is the speed-up factor based on a one node simulation, $E_s = S_N/C$ is the efficiency, T_N is the compute time of a parallel simulation on one node, and T_C is the compute time of a parallel simulation on



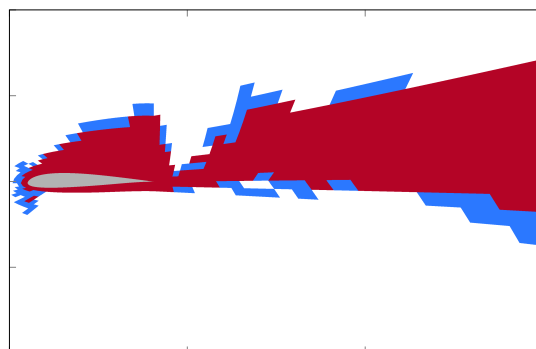
(a) Vorticity at $0T_e/4$.



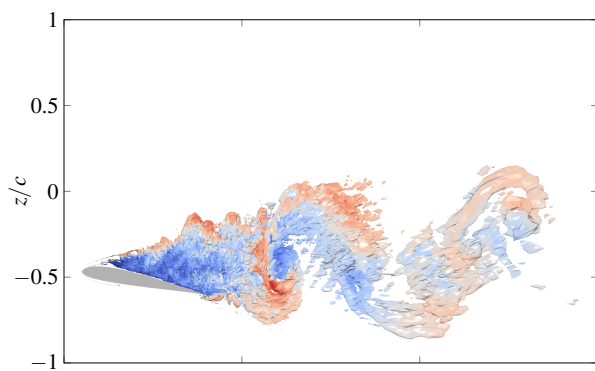
(b) Polynomial degree at $0T_e/4$.



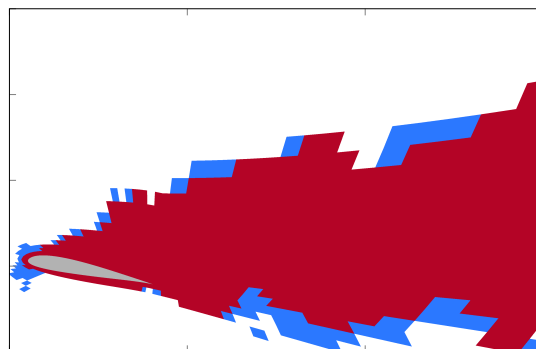
(c) Vorticity at $1T_e/4$.



(d) Polynomial degree at $1T_e/4$.



(e) Vorticity at $2T_e/4$.



(f) Polynomial degree at $2T_e/4$.

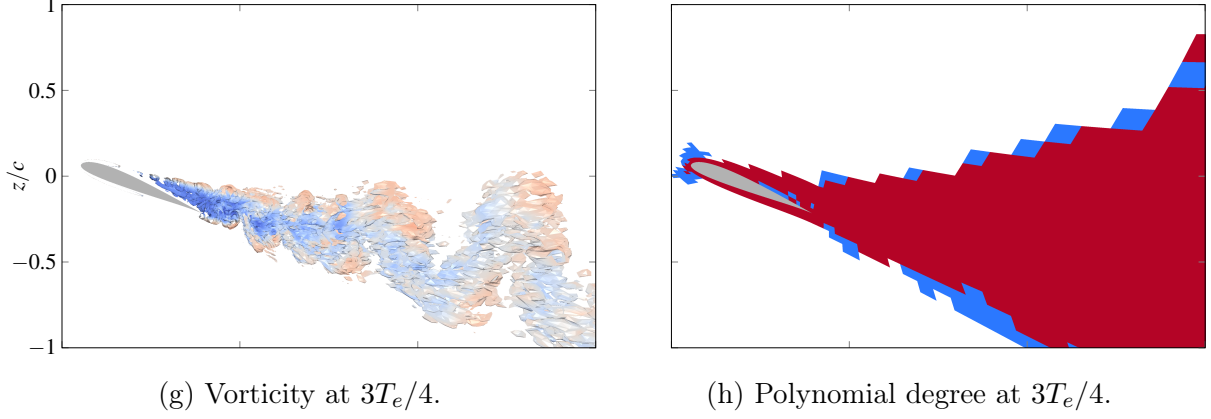


Figure 7.4. Contours of Q-criterion colored by velocity magnitude, and polynomial distribution for the adaptive dynamic load-balanced heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.

C cores. These results verify the scalability of the algorithm and the capability of the DLB to distribute uniform computation load among processors.

Table 7.1. Computation time of the dynamic load-balanced adaptive algorithm for the heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.

Scheme	Number of Nodes	Number of Cores	T_{itr}	S_A
$K = 1$	1	40	3.832×10^2	-
$K = 2$			8.629×10^2	-
$K = 3$			1.750×10^3	-
$K = 1 - 3$			6.416×10^2	2.73

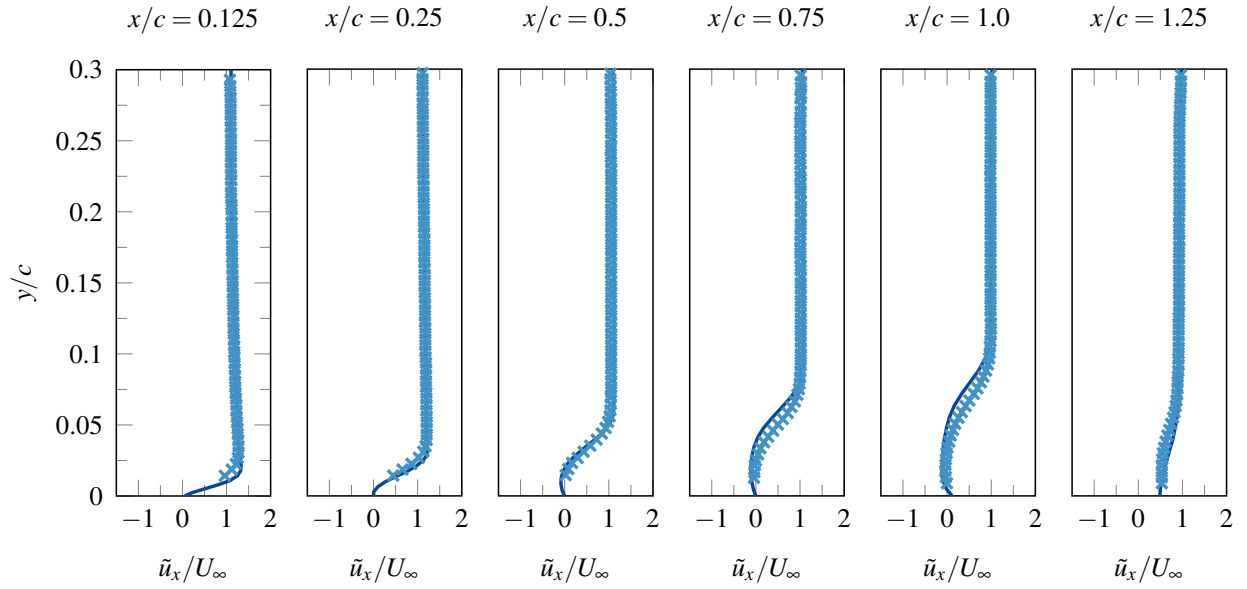
Table 7.2. Scalability metrics of the dynamic load-balanced adaptive algorithm for the heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.

Number of nodes	Number of cores	S_N	E_s
$N_n = 1$	40	1.000	1.000
$N_n = 2$	80	1.979	0.989
$N_n = 4$	160	3.946	0.986
$N_n = 8$	320	7.669	0.959
$N_n = 16$	640	13.722	0.861
$N_n = 32$	1280	24.258	0.758

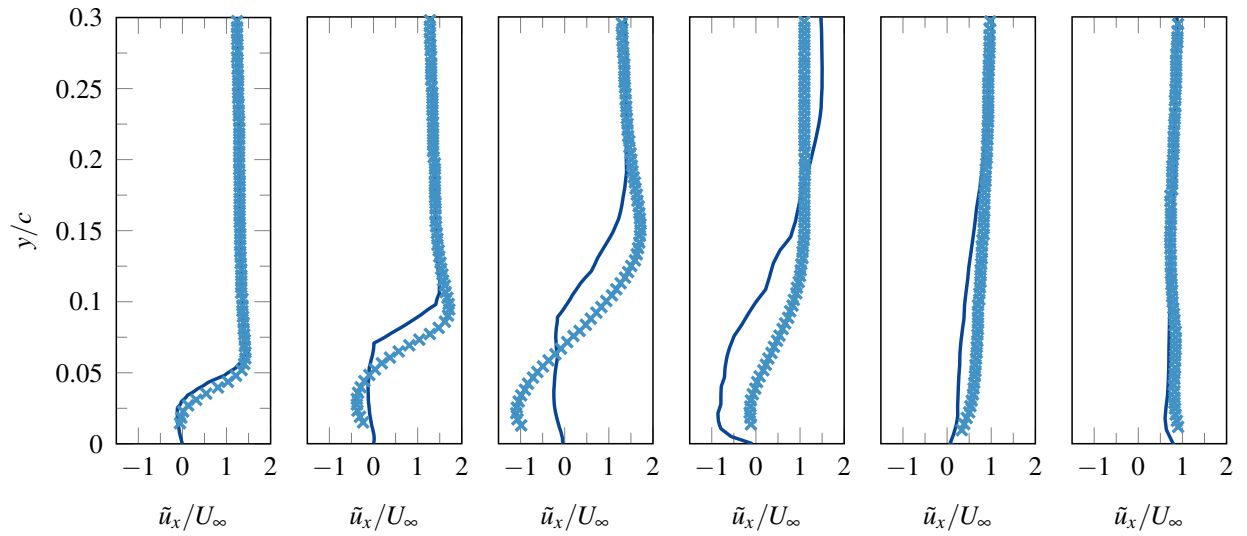
7.2 Turbulent Flow Over a 3D Circular Cylinder

7.2.1 Introduction

Cross-flow over a 3D circular cylinder has been the focus of several previous studies both due to its physical significance and its arrangement simplicity in experiments. As mentioned in Chapter 5, its characteristics are known to be highly dependent on the Reynolds number Re . The flow is laminar with symmetric vortices for Reynolds numbers below 49. Increasing Re



(a) $t = 0T_e/4$



(b) $t = 1T_e/4$

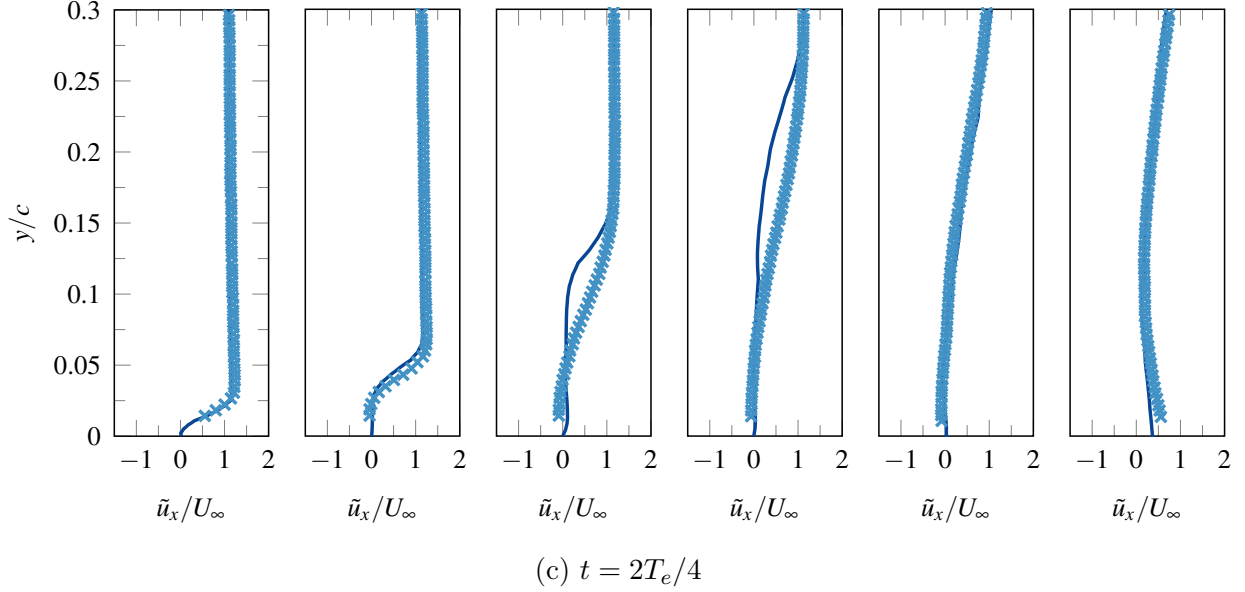


Figure 7.5. Phase averaged x-component of the velocity profile at different chord locations and time instants (solid lines correspond to this study, and the crosses correspond to the experiment [10]) for the heaving and pitching SD 7003 airfoil with $Re = 10000$, $A = 0.5$, $f_K = 0.25$, $\theta_0 = 8^\circ$, $\theta_e = -8.42^\circ$ and $\phi_e = 90^\circ$.

further, vortex shedding will be observed, but the wake flow remains laminar for $Re < 170$. At Re larger than about 170, the wake structure becomes 3D, even though the wake flow is in the transition regime between about $170 < Re < 300$, and is not yet turbulent. As the Re increases above about 300, there would be increasing disorder in the fine-scale three-dimensionality and the wake flow becomes fully turbulent at Re larger than about 400 [100, 139, 140, 141], which is the case in real-life applications. Hence, in this section, flow over a three-dimensional circular cylinder at $Re = 1000$, which sits on the onset of the shear-layer transition regime identified by Williamson [100], is studied to verify the effectiveness of the vorticity-based polynomial adaptation technique coupled with dynamic load balancing in the existence of turbulent transition and turbulent wake flow behind a bluff body. It has been studied previously by, for example, Braza et al. [142] using the finite volume method, Zhao et al. [143] using Petrov–Galerkin finite element method, where the effects of different yaw angle on the turbulent features is studied, and Pereira [96] using the flux reconstruction method where the effect of different element types on the numerical results are investigated.

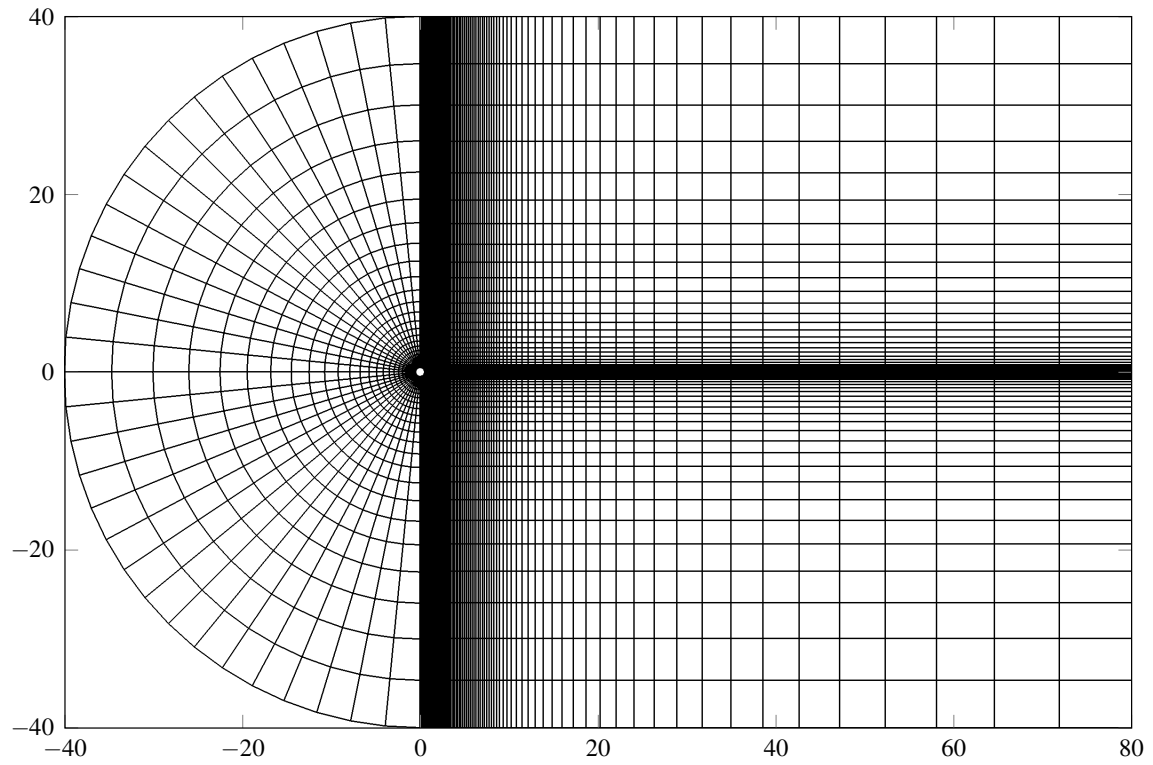
7.2.2 Computational Details

A three-dimensional computational domain, with its origin located at the centre of the cylinder, containing a total of 78176 hexahedral elements with 14 elements in the span-wise direction was used with Riemann invariant boundary conditions at the far-field, periodic boundary condition in the span-wise direction, and a no-slip adiabatic wall boundary condition at the surface of the cylinder. The domain extends to $40D$ above, below, and upstream of the cylinder, $80D$ downstream, and $2\pi D$ in the span-wise direction, where D is the cylinder diameter, shown in Figure 7.6. The mesh is moderately refined near the wall and in the wake region, and uses quadratically curved elements at the boundaries to match the cylinder geometry. The mesh was initially partitioned over 320 processors using METIS [102], and MPI is used for parallel communication [99].

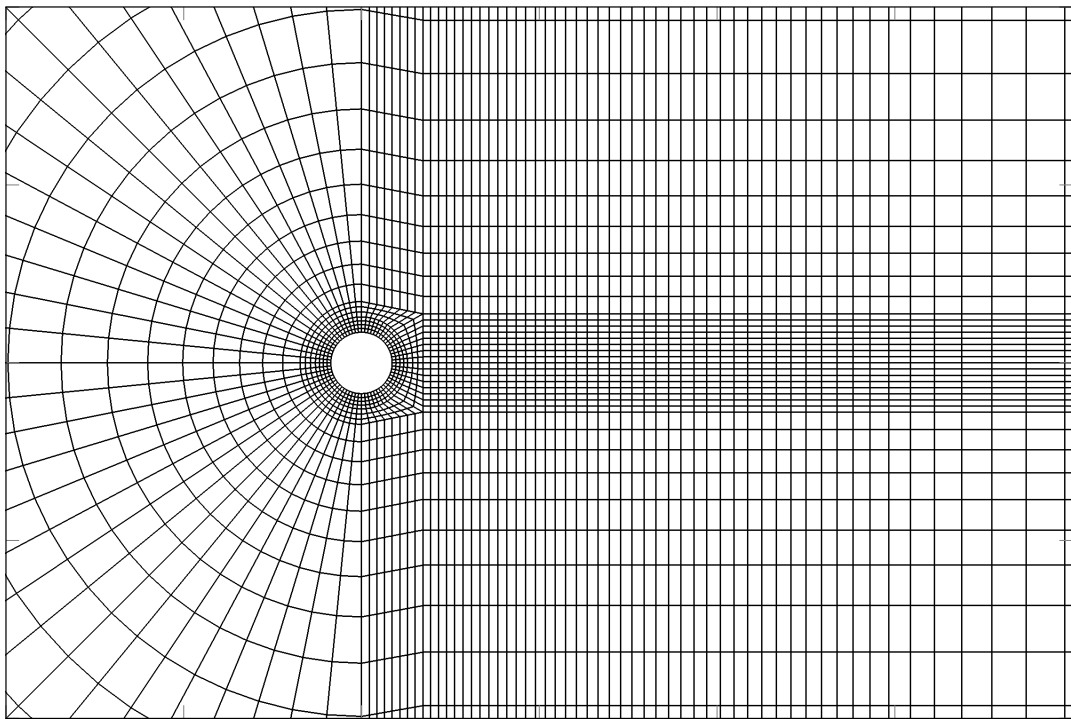
Simulations were run at Mach number $M = 0.2$, for a total of $400t_c$, where $t_c = D/U_\infty$ is the time required for the flow to traverse one cylinder diameter. Time integration was carried out with the fourth-order twelve-stage optimized Runge-Kutta scheme [70] mentioned in Chapter 3, and the non-dimensional time step is set to $\Delta t^* = \Delta t U_\infty / D = 8.0 \times 10^{-4}$. As mentioned before, by having more stages, this Runge-Kutta scheme allows for a larger time step size which relatively reduces the computational cost. A total of six simulations were carried out to verify the utility of the dynamically load-balanced adaptation algorithm, including five uniform simulations with solution polynomials of degree $K = 1$ to $K = 5$ and a five-level adaptive simulation. The adaptation parameters K_m , ϵ , and $\vartheta_{1:5}$ are set to 5, 1.25×10^{-1} , and $[0.25, 0.25, 1, 4, 16]$ respectively. The adaptation and the DLB routines are called every 100 and 1000 iterations, respectively. Simulations were carried out on eight nodes of the Niagara cluster, each node consists of 2 sockets with 20 Intel Skylake cores (2.4 GHz, AVX512), for a total of 40 cores per node, and a total of 202 GB of RAM.

7.2.3 Numerical Results

We are interested in achieving speed-up factors relative to the higher-polynomial degree solutions, while still maintaining accuracy with respect to the reference data. Results are compared in terms of mean drag coefficient \bar{C}_D , root mean square of the lift coefficient C_{Lrms} ,

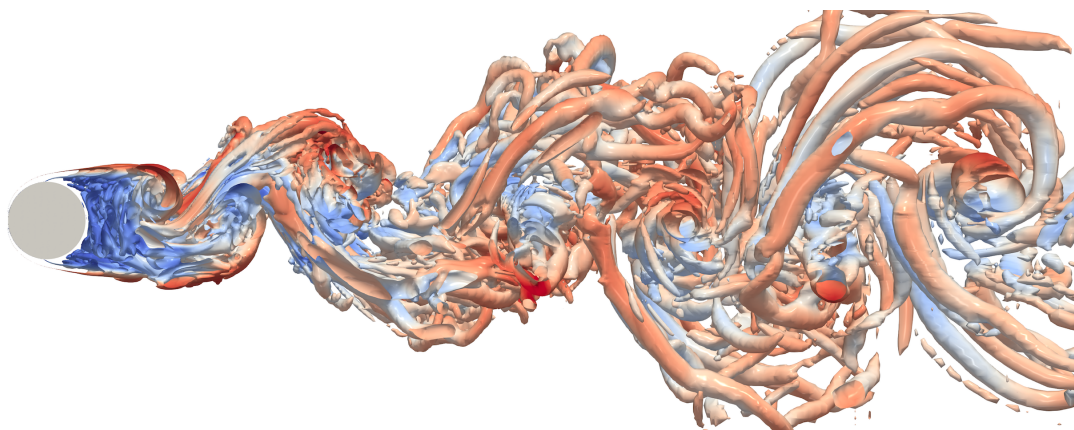


(a) Complete view. Dimensions are given in terms of cylinder diameter D .

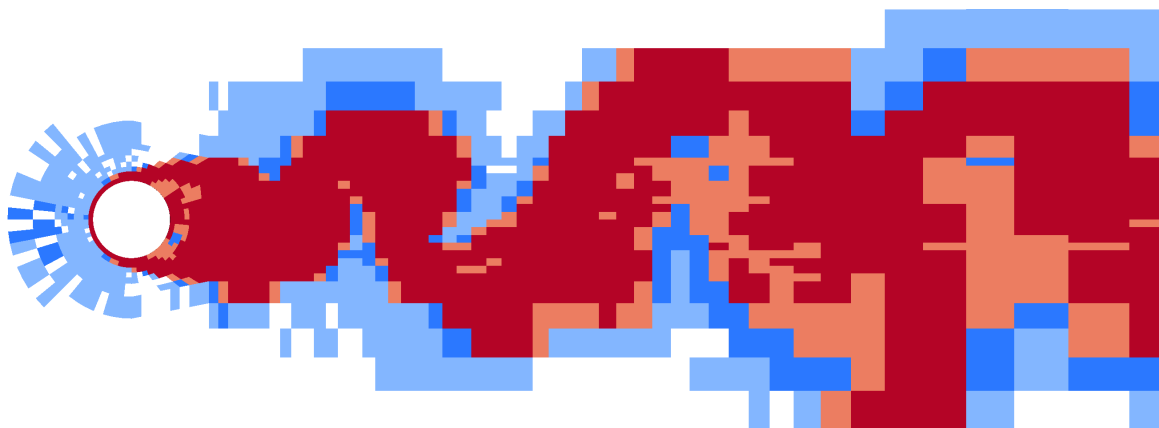


(b) Close-up view.

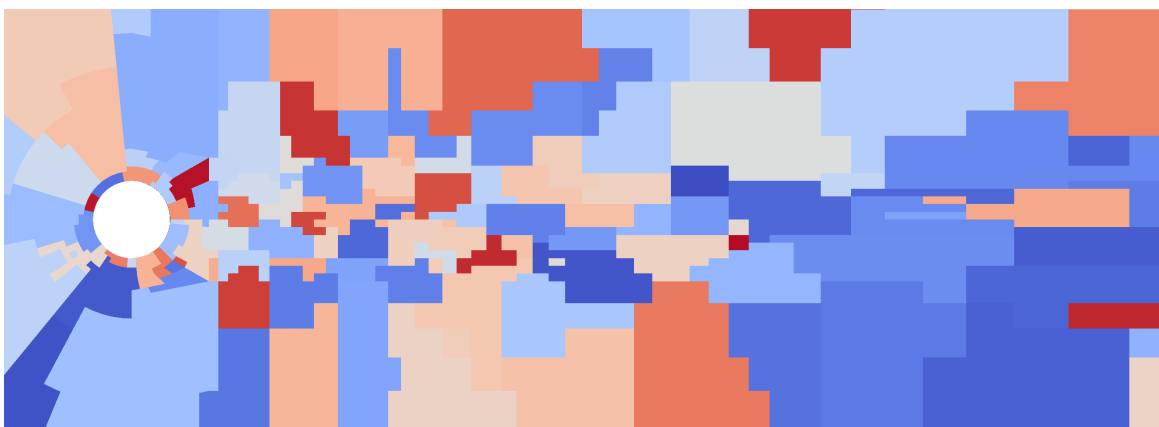
Figure 7.6. Three-dimensional circular cylinder structured mesh with 78176 hexahedral elements.



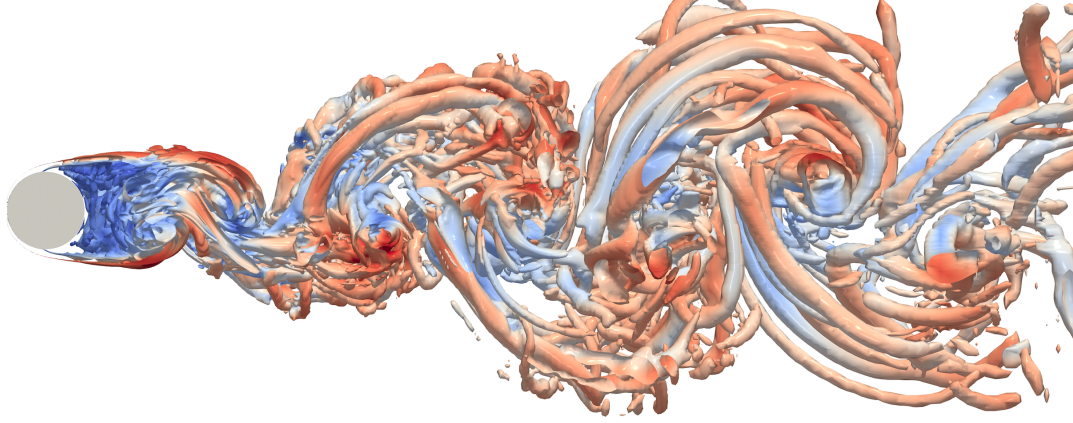
(a) Q-criterion at a maximum lift instant.



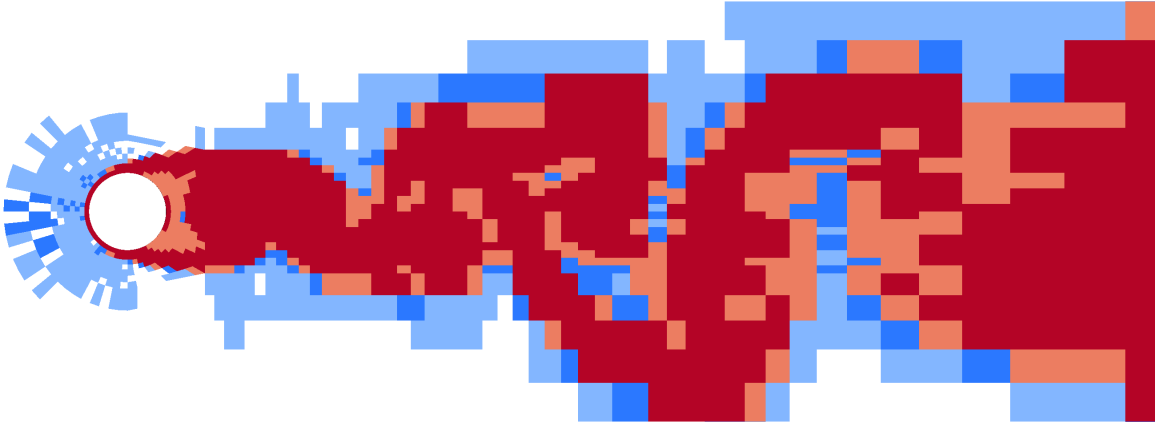
(b) Polynomial degree a maximum lift instant.



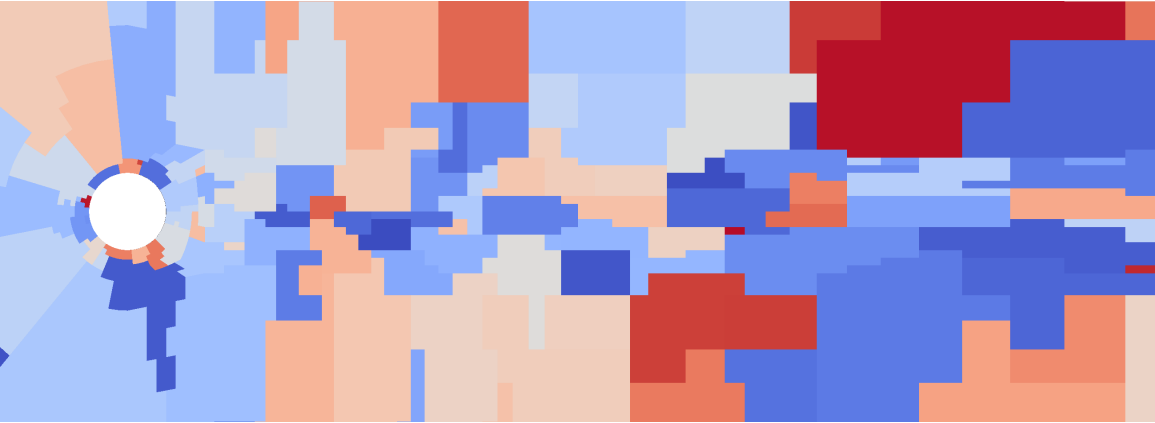
(c) Mesh partition at a maximum lift instant.



(d) Q-criterion at a minimum lift instant.



(e) Polynomial degree a minimum lift instant.



(f) Mesh partition at a minimum lift instant.

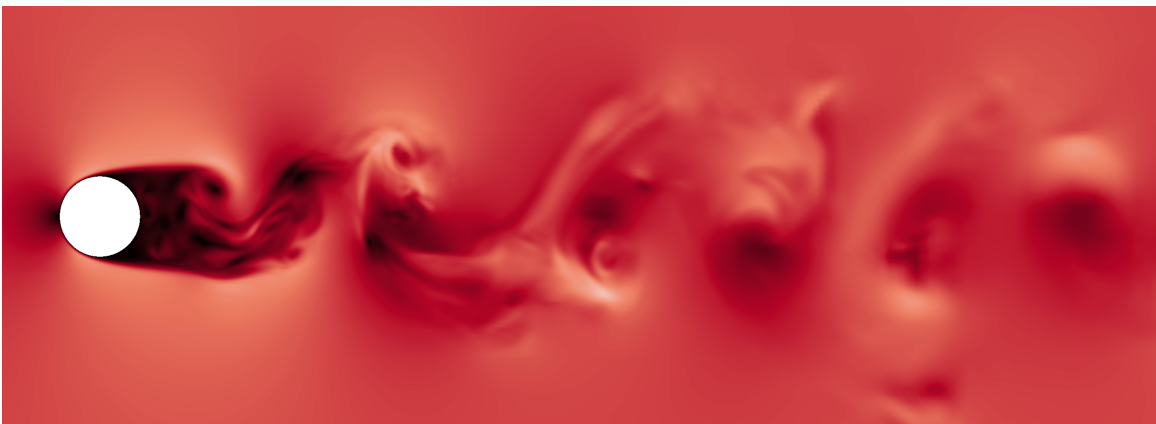
Figure 7.7. Contours of Q-criterion, polynomial distribution, and dynamically balanced mesh partition for the adaptive computation ($K = 1 - 5$) based on the vorticity magnitude indicator at the times corresponding to maximal and minimal lift for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.



(a) Velocity field for $K = 1$.



(b) Velocity field for $K = 5$.



(c) Velocity field for $K = 1 - 5$.

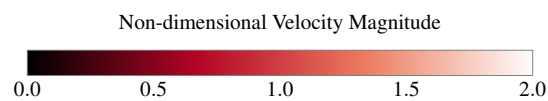
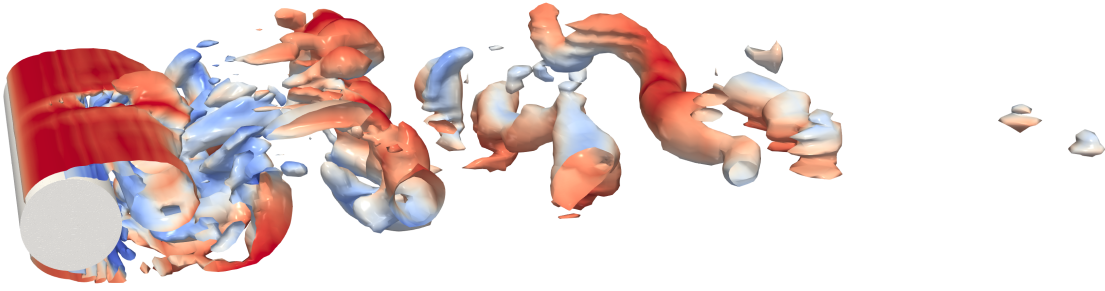
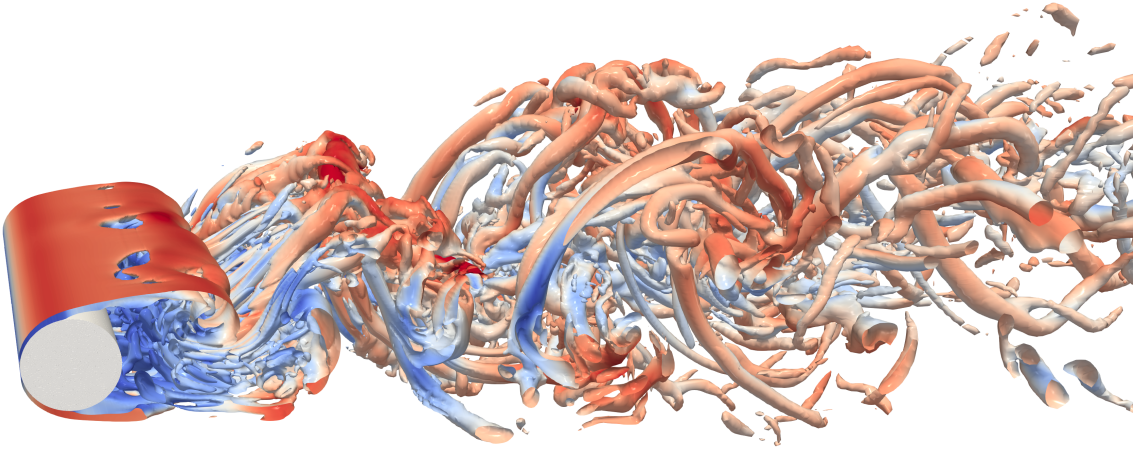


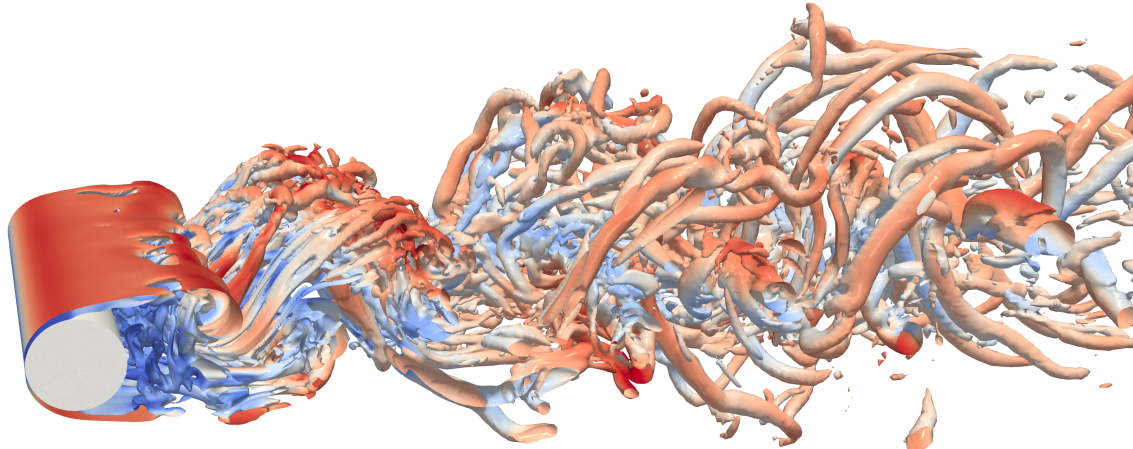
Figure 7.8. Non-dimensional velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive load balanced $K = 1 - 5$ computations based on the vorticity magnitude indicator at the times corresponding to maximal lift for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.



(a) Contours of Q-criterion for $K = 1$.



(b) Contours of Q-criterion for $K = 5$.



(c) Contours of Q-criterion for $K = 1 - 5$.

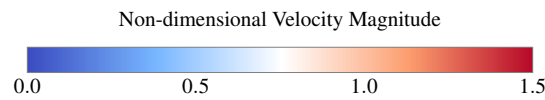


Figure 7.9. Contours of Q-criterion colored by velocity magnitude for the uniform $K = 1$, uniform $K = 5$, and adaptive load-balanced $K = 1 - 5$ computations based on the vorticity magnitude indicator at the times corresponding to maximal lift for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.

root mean square of the drag coefficient C_{Drms} , Strouhal number St , the average total number of degrees of freedom \overline{DOF} , compute time T_{itr} , which is the simulation time of one iteration of a parallel simulation on one nodes of the Niagara cluster, and the adaptive speed-up factor $S_A = T_A/T_U$ defined based on a uniform $K = 5$ simulation with the same mesh resolution, where T_A is the compute time of an adaptive $K = 1 - 5$ simulation featured with DLB, and T_U is the compute time of a uniform $K = 5$ simulation. The lift and drag coefficients are defined as

$$\begin{aligned} C_L &= \frac{2F_L}{\rho_\infty U_\infty^2 A_f}, \\ C_D &= \frac{2F_D}{\rho_\infty U_\infty^2 A_f}, \end{aligned} \tag{7.1}$$

where A_f is the frontal area which is equal to the area of the projected surface normal to the flow, taken to be the cylinder diameter D multiplied by the span length. The average and the root mean square value of a time-dependant variable $v(t)$ over a time interval $T_{int} = t_n - t_1$ is calculated using Equation 6.4. Figure 7.7 shows the contours of the Q-criterion colored by the velocity magnitude, the polynomial degree distribution, and the repartitioned mesh for the five-level adaptive simulation ($K = 1 - 5$) for two instants of minimal and maximal lift. This verifies that the adaptation algorithm successfully tracks elements with large vorticity magnitude relative to their size. We observe that a relatively fewer number of high-order elements are allocated to processors near the cylinder and in the wake region, while processors located farther away from the cylinder and the wake contain a larger number of lower-order elements. This illustrates the ability of the DLB algorithm to maintain the load balance. Figure 7.8 shows contours of non-dimensional velocity magnitude at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. Although there are large inter-elements jumps and unresolved regions in the flow solution of the uniform $K = 1$ simulation, the smoothness of the flow solution is comparable for the uniform high-order $K = 5$ and adaptive $K = 1 - 5$ simulations. Figure 7.9 depicts three-dimensional contours of Q-criterion colored by the velocity magnitude at the time of maximal lift for uniform $K = 1$, uniform $K = 5$, and adaptive $K = 1 - 5$. While the uniform $K = 1$ simulation did not capture all vortical structures in the flow, the uniform $K = 5$ and adaptive $K = 1 - 5$ simulations

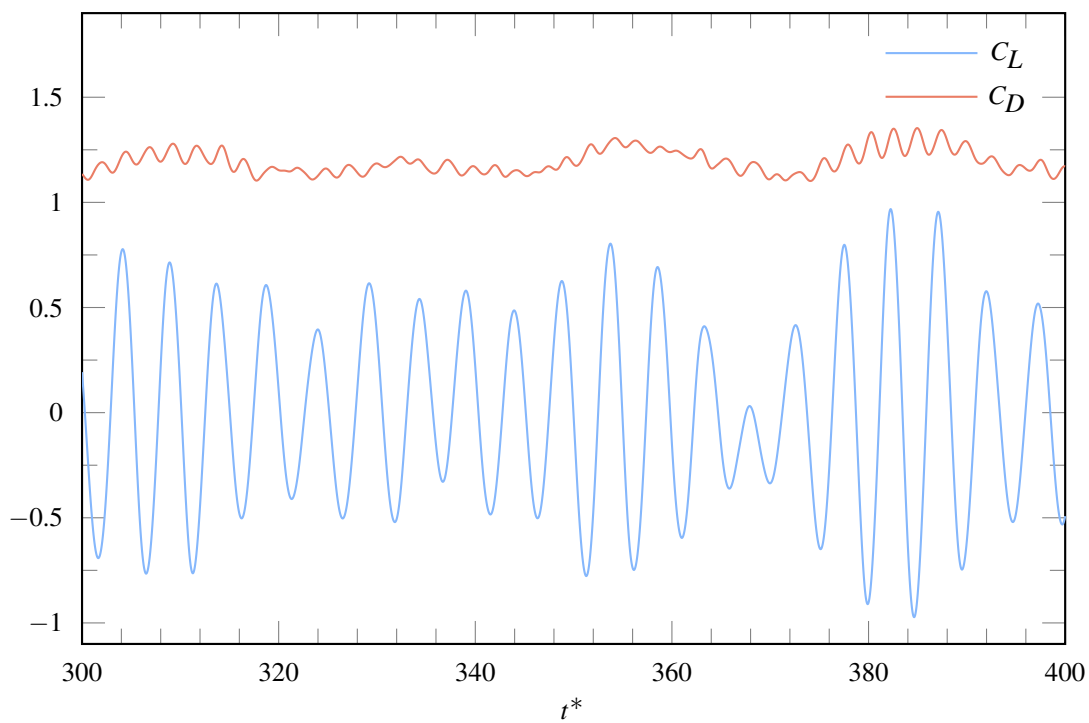
Table 7.3. Numerical values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , T_{itr} and S_A averaged over the 100 convective times for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.

Degree	\bar{C}_D	C_{Lrms}	C_{Drms}	St	\overline{DOF}	T_{itr}	S_A
$K = 1$	1.1946	0.4494	0.0557	0.2037	6.254×10^5	7.418×10^2	-
$K = 2$	1.0828	0.2371	0.0333	0.2080	2.111×10^6	2.137×10^3	-
$K = 3$	1.0084	0.1004	0.0221	0.2099	5.003×10^6	4.744×10^3	-
$K = 4$	1.0180	0.1036	0.0267	0.2047	9.772×10^6	9.746×10^3	-
$K = 5$	1.0141	0.0986	0.0193	0.2053	1.689×10^7	1.895×10^4	-
$K = 1 - 5$	1.0549	0.1587	0.0218	0.2084	3.075×10^6	4.095×10^3	4.63
Pereira [96]	1.064	-	-	-	-	-	-
Zhao et al. [143]	1.092	0.310	-	0.202	-	-	-

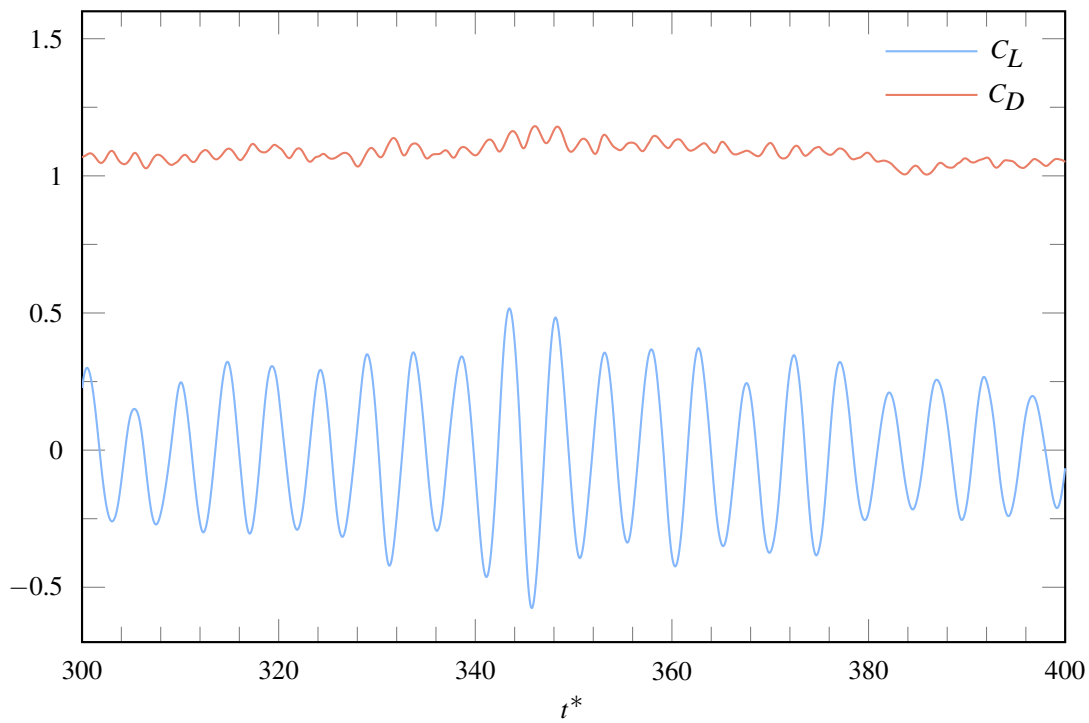
both captured turbulent structures indistinguishably, confirming that the five-level adaptive simulation qualitatively provides satisfactory resolution near the cylinder and the wake region, while reducing the final computational cost by requiring fewer degrees of freedom. To further evaluate the accuracy and speed-up factor of the adaptive method, we turn to Table 7.3 which reports the quantitative values of \bar{C}_D , C_{Lrms} , C_{Drms} , St , \overline{DOF} , T_{itr} , and S_A averaged during $100t_c$ and compares them against the reference data. These results quantitatively show agreement within 4.02%, 60.95%, and 12.95% in the \bar{C}_D , C_{Lrms} , and C_{Drms} values respectively between the adaptive and uniform $K = 5$ simulations, while the former is 4.63 times faster. Figure 7.10 shows the lift and drag coefficient plots for all simulations plotted within the last $100t_c$.

7.3 Conclusion

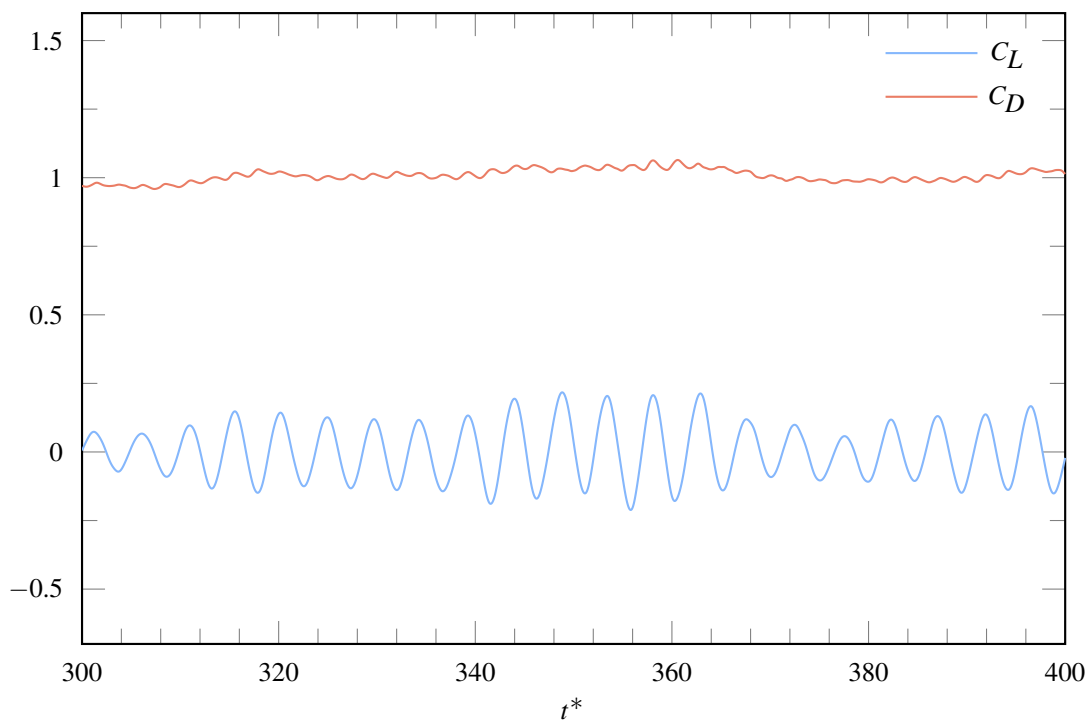
We have further explored the utility of the load-balanced vorticity-based adaptation algorithm for turbulent flows by performing simulations for a shallow dynamic stall of a three-dimensional heaving and pitching SD 7003 airfoil, and a three-dimensional circular cylinder. In both test studies, the adaptation algorithm was featured with a DLB technique to circumvent the overhead caused by adaptation. Results demonstrated that the algorithm is capable of tracking salient turbulent features. Qualitative results showed equivalent levels of accuracy between the adaptive and high-order solutions, with a significant reduction in simulation time. Results from the adaptive simulations also showed good quantitative agreement with



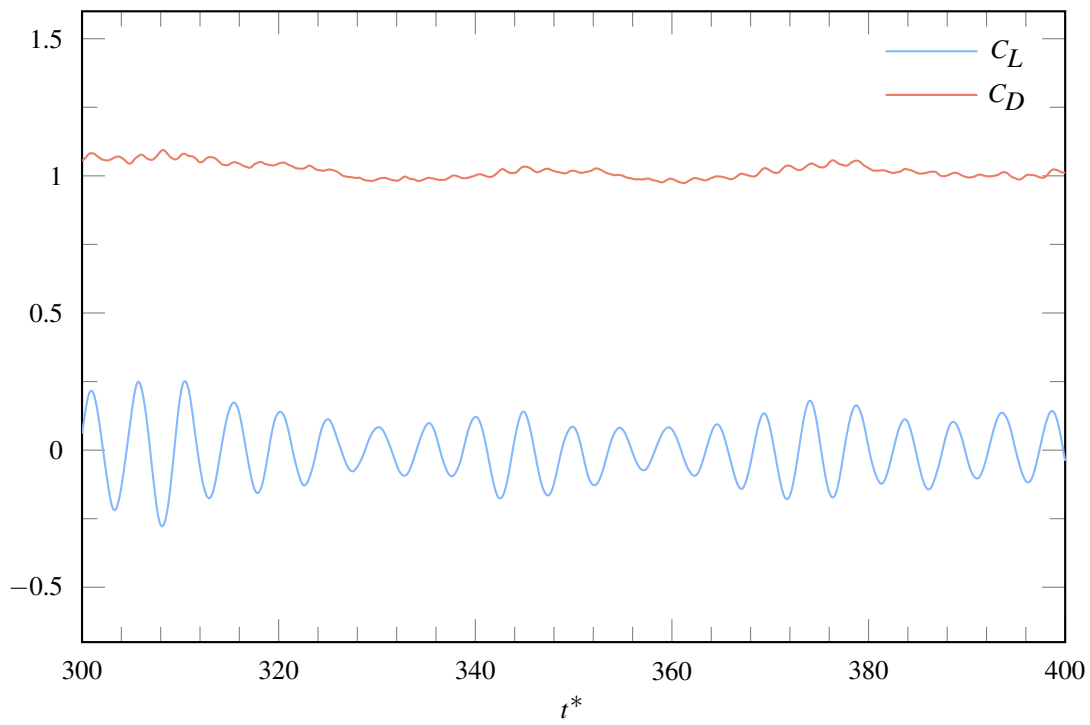
(a) Uniform $K = 1$.



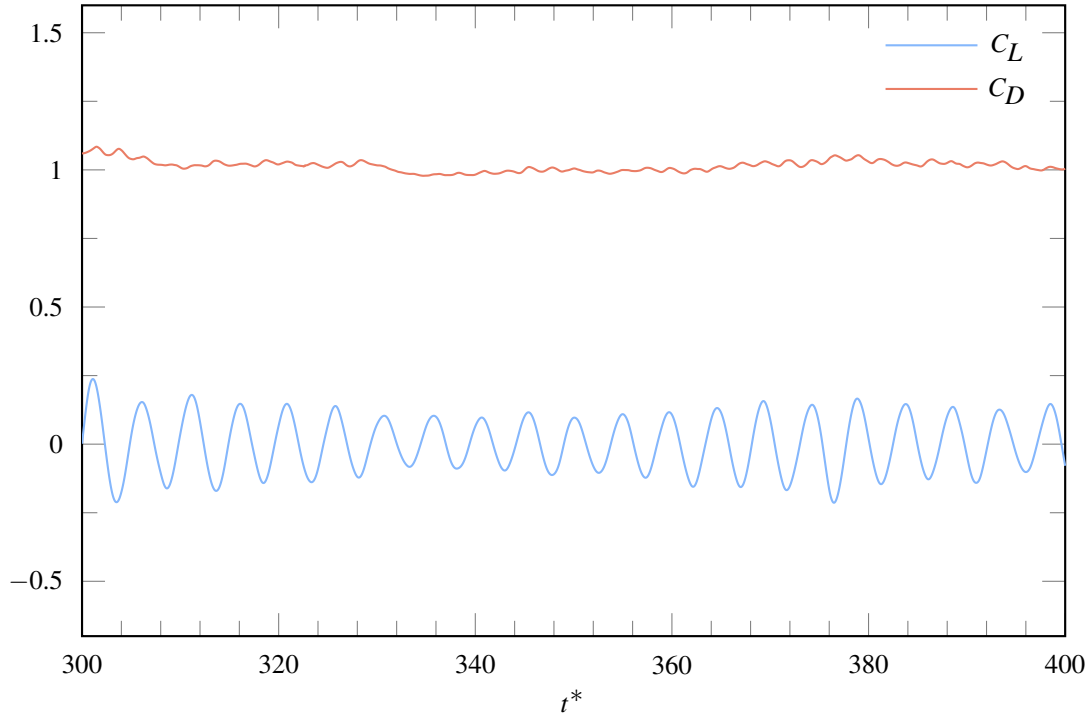
(b) Uniform $K = 2$.



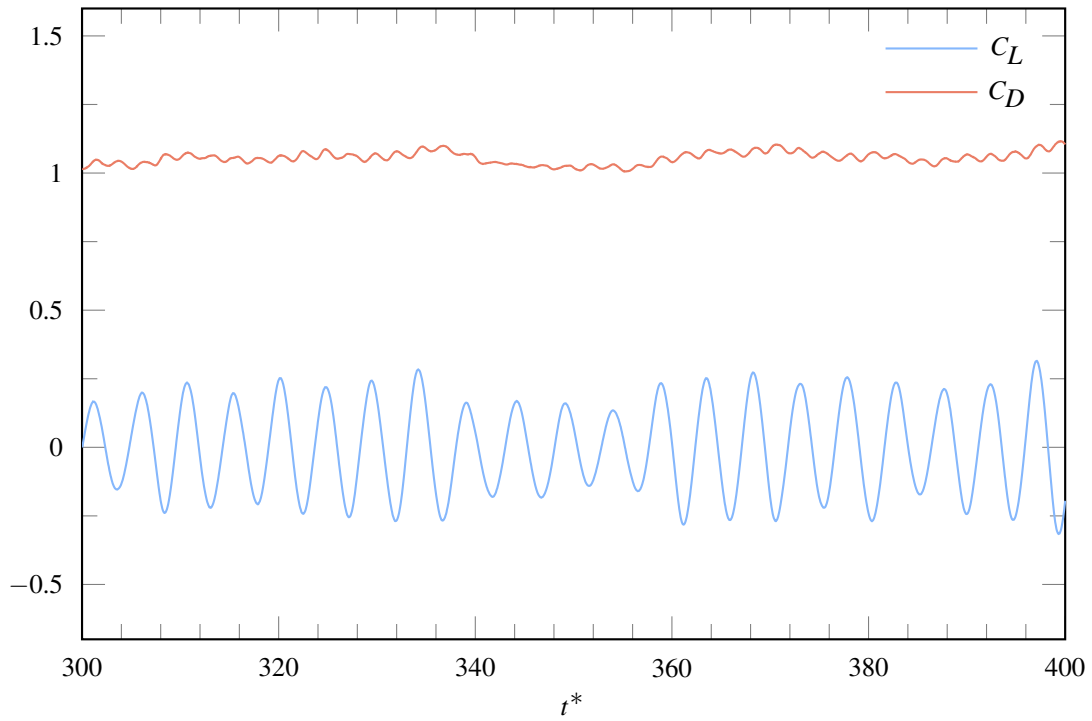
(c) Uniform $K = 3$.



(d) Uniform $K = 4$.



(e) Uniform $K = 5$.



(f) Adaptive $K = 1 - 5$.

Figure 7.10. Drag and lift coefficient profiles of the uniform and adaptive simulations for the three-dimensional circular cylinder with $Re = 1000$ and $M = 0.2$.

parallel high-order simulations and reference numerical, experimental, and analytical results. In addition, results showed that the DLB algorithm was able to keep the domain balanced by redistributing elements at runtime. Hence, the proposed non-dimensional load-balanced vorticity-based adaptation algorithm is an effective and accurate approach for performing p -adaptation for unsteady transitional and turbulent flows.

The following chapter will summarize the conclusions of this study and include recommendations for future work.

Chapter 8

Conclusions and Future work

In this work, we have introduced a novel dynamically load-balanced vorticity-based polynomial adaptation algorithm for simulations of unsteady flows on both fixed and moving or deforming domains. In this chapter, we summarize the results obtained throughout this study.

We made use of the implicit large eddy simulation turbulence modelling approach for the three-dimensional simulations. The flux construction scheme was used for spatial discretization due to its suitability for scale-resolving simulations of unsteady turbulent flows, as well as modern many-core hardware architectures. Furthermore, the numerical dissipation error of FR acts as the subgrid-scale turbulence model befitted for ILES. Among other advantages of FR, it was particularly appealing for this study as it allows for dynamic adaptation of degree of solution polynomials owing to its element-wise approach. To circumvent the overhead caused by adaptation when combined with parallel computing, we borrowed and customized the adaptive repartitioning algorithm from the ParMETIS library [95].

We first verified the FR scheme and the computer code when applied to the ALE form of the compressible Navier–Stokes equations. Results showed that the deforming mesh cases achieved their designed high-order accuracy. We then verified the utility of the dynamic load-balanced p -adaptation algorithm by performing simulations of an isentropic vortex. We observed that the algorithm achieved a significant speed-up while maintaining high-order accuracy. The adaptation algorithm was validated on fixed domains by solving different two-dimensional test cases, including simulation of flow over a circular cylinder and an SD 7003 airfoil. Results demonstrated that the adaptation algorithm is capable of tracking

elements with large vorticity magnitude relative to their size. Furthermore, results of the adaptive simulation were in excellent agreement with high-order solutions, both qualitatively and quantitatively, while requiring a relatively fewer total number of degrees of freedom.

On moving and deforming domains, the adaptation algorithm was validated for two classical problems, specifically, a pair of oscillating circular cylinder configurations and dynamic stall of heaving and pitching NACA 0012 airfoil. Results demonstrated that the adaptation algorithm is capable of tracking salient flow features on deforming domains, such as the boundary layer and unsteady wake regions. Qualitative results showed equivalent levels of accuracy between the adaptive and high-order solutions, with a significant reduction in the total number of degrees of freedom. We then considered a practical test case of flow over a VAWT. Similar to the validation test cases, it was observed that the adaptive simulations were able to track the boundary layer and wake regions of each blade, and their subsequent interactions. Results from the adaptive simulations also showed good quantitative agreement with parallel high-order simulations and reference numerical, experimental, and analytical results.

Finally, three-dimensional simulations of transitional and turbulent flows were performed to validate the dynamically load-balanced adaptation algorithm, specifically, for a shallow dynamic stall of a three-dimensional heaving and pitching SD 7003 airfoil, and turbulent flows over a three-dimensional circular cylinder. Results verified the scalability of the algorithm and also showed that the dynamically load-balanced adaptation algorithm is capable to circumvent the adaptation overhead when applied to unsteady turbulent flows by repartitioning the mesh at runtime. Qualitative and quantitative results showed equivalent levels of accuracy between the adaptive and high-order solutions, with a significant speed-up.

The proposed non-dimensional vorticity-based adaptation indicator is a simple, effective, and accurate approach for performing p -adaptation of unsteady flows. Furthermore, the DLB algorithm is capable of increasing the efficiency of the parallel adaptive simulations by dynamically distributing computation load among processors. It also increases the granularity of adaptive simulations by increasing the scalability. As mentioned earlier, industrial deployment of scale-resolving techniques such as LES are avoided due to their inherently expensive nature. However, the current work could serve toward the industrialization of high-fidelity

scale-resolving techniques. Future works should focus primarily on applying the proposed adaptation technique to more complex three-dimensional geometries as well as on deployment of this algorithm to industrial problems, including but not limited to aerodynamic shape optimization to achieve a better fuel efficiency, aeroacoustic optimization to reduce aircraft noise, and studying new proposed alternative designs.

Bibliography

- [1] J. H. Strickland, T. Smith, and K. Sun. A vortex model of the Darrieus turbine: an analytical and experimental study, Sandia National Laboratories. Technical report, SAND 81-7017, 1981.
- [2] Boeing’s sub-scale X-48B Blended Wing Body aircraft. <https://www.nasa.gov/centers/dryden/multimedia/imagegallery/X-48B/ED07-0192-06.html>. Editor: Yvonne Gibbs.
- [3] J. L. Felder. Nasa N3-X with turboelectric distributed propulsion. Technical report, 2014.
- [4] J. Banke. Reduce fuel burn with a dose of BLI. <https://www.nasa.gov/aero/reduce-fuel-burn-with-a-dose-of-bli>. Editor: Lillian Gipson.
- [5] The Double Bubble D8. <https://www.nasa.gov/content/the-double-bubble-d8-0>. Editor: Lillian Gipson.
- [6] C. H. Williamson and A. Roshko. Vortex formation in the wake of an oscillating cylinder. *Journal of fluids and structures*, 2(4):355–381, 1988.
- [7] M. Moriche, O. Flores, and M. García-Villalba. On the aerodynamic forces on heaving and pitching airfoils at low Reynolds number. *Journal of Fluid Mechanics*, 828:395–423, 2017.
- [8] S. Kanner and P. Persson. Validation of a high-order large-eddy simulation solver using a vertical-axis wind turbine. *AIAA Journal*, 54(1):101–112, 2016.

- [9] J. H. Strickland, B. T. Webster, and T. Nguyen. A vortex model of the Darrieus turbine: an analytical and experimental study. 1979.
- [10] Y. S. Baik, J. Rausch, L. Bernal, and M. Ol. Experimental investigation of pitching and plunging airfoils at Reynolds number between 1×10^4 and 6×10^4 . In *39th AIAA fluid dynamics conference*, page 4030, 2009.
- [11] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer, January 2008.
- [12] International Air Transit Association. Air passenger market analysis - november 2021. <https://www.iata.org/en/iata-repository/publications/economic-reports/air-passenger-monthly-analysis---november-2021>, 2021.
- [13] International Civil Aviation Organization. Annual report 2019. The world of air transport in 2019. <https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx>, ICAO, 2019.
- [14] M. Cames, J. Graichen, A. Siemons, and V. Cook. Emission reduction targets for international aviation and shipping. *Policy Department A: Economic and Scientific Policy, European Parliament, B-1047 Brussels*, 2015.
- [15] International Civil Aviation Organization. Resolutions adopted at the 37th session of the assembly. ICAO, 2010. Printed Montreal, QC.
- [16] Council of European Union. Council regulation (EU) no 71/2008: Setting up the Clean Sky Joint Undertaking, 2007.
- [17] F. Collier. Overview of NASA’s Environmentally Responsible Aviation (ERA) project. In *NASA Environmentally Responsible Aviation Project Pre-Proposal Meeting. Washington, DC*, 2010.
- [18] Green Aviation Research & Development Network. Annual report 2016-2017, 2017.
- [19] Air Transport Action Group (ATAG). The right flightpath to reduce aviation emissions. *UNFCCC Climate Talks*, 2011.

- [20] R. H. Liebeck. Design of the blended wing body subsonic transport. *Journal of aircraft*, 41(1):10–25, 2004.
- [21] R. T. Kawai, D. M Friedman, and L. Serrano. Blended Wing Body (BWB) Boundary Layer Ingestion (BLI) inlet configuration and system studies. Technical report, 2006.
- [22] J. Felder, H. Kim, G. Brown, and J. Kummer. An examination of the effect of boundary layer ingestion on turboelectric distributed propulsion systems. In *49th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 300, 2011.
- [23] A. T. Wick, J. R. Hooker, and C. H. Zeune. Integrated aerodynamic benefits of distributed propulsion. In *53rd AIAA Aerospace Sciences Meeting*, page 1500, 2015.
- [24] E. M. Greitzer, P. A. Bonnefoy, E. DelaRosaBlanco, C. S. Dorbian, M. Drela, D. K. Hall, R. J. Hansman, J. I. Hileman, R. H. Liebeck, J. Lovegren, et al. N+3 aircraft concept designs and trade studies. Technical report, 2010.
- [25] A. Uranga, M. Drela, E. Greitzer, N. Titchener, M. Lieu, N. Siu, A. Huang, G. M. Gatlin, and J. Hannon. Preliminary experimental assessment of the boundary layer ingestion benefit for the D8 aircraft. In *52nd Aerospace Sciences Meeting*, page 0906, 2014.
- [26] L. Euler. Principes généraux de l’état d’équilibre des fluides. *Mémoires de l’académie des sciences de Berlin*, pages 217–273, 1757.
- [27] C. J. M. H. Navier. Mémoire sur les lois du mouvement des fluides. *Mémoires de l’Académie Royale des Sciences de l’Institut de France*, 6:389–440, 1823.
- [28] G. G. Stokes. On the Theories of the Internal Friction of Fluids in Motion, and of the Equilibrium and Motion of Elastic Solids. In *Mathematical and Physical Papers vol.1*, volume 1 of *Cambridge Library Collection - Mathematics*. 2007.
- [29] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences. Technical Report NASA/CR–2014-218178, 2014.

- [30] P. A. Davidson. *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, 2015.
- [31] C. D. Argyropoulos and N. C. Markatos. Recent advances on the numerical modelling of turbulent flows. *Applied Mathematical Modelling*, 39(2):693–732, 2015.
- [32] A. N. Kolmogorov. The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds’ Numbers. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 434(1890):9–13, 1991.
- [33] O. Reynolds. On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. *Philosophical Transactions of the Royal Society of London. A*, 186:123–164, January 1895.
- [34] J. Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.
- [35] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.
- [36] P. Vincent, F. Witherden, B. Vermeire, J. S. Park, and A. Iyer. Towards Green Aviation with Python at Petascale. In *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 11, 2016.
- [37] J. S. Park, F. D. Witherden, and P. E. Vincent. High-order implicit large-eddy simulations of flow over a NACA0021 aerofoil. *AIAA Journal*, pages 2186–2197, 2017.
- [38] B. C. Vermeire and P. E. Vincent. On the behaviour of fully-discrete flux reconstruction schemes. *Computer Methods in Applied Mechanics and Engineering*, 315:1053–1079, 2017.
- [39] C. A. Pereira and B. C. Vermeire. Fully-discrete analysis of high-order spatial discretizations with optimal explicit runge–kutta methods. *Journal of Scientific Computing*, 83(3):1–35, 2020.

- [40] B. Vermeire, J. Cagnone, and S. Nadarajah. ILES using the correction procedure via reconstruction scheme. In *51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 1001, 2013.
- [41] B. C. Vermeire, S. Nadarajah, and P. G. Tucker. Implicit large eddy simulation using the high-order correction procedure via reconstruction scheme. *International Journal for Numerical Methods in Fluids*, 82(5):231–260, 2016.
- [42] D. Drikakis, C. Fureby, F. F. Grinstein, and D. Youngs. Simulation of transition and turbulence decay in the Taylor–Green vortex. *Journal of Turbulence*, (8):N20, 2007.
- [43] M. Hamed and B. C. Vermeire. Optimized filters for stabilizing high-order large eddy simulation. *Computers & Fluids*, page 105301, 2022.
- [44] C. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, August 1988.
- [45] G. Jiang and C. Shu. Efficient Implementation of Weighted ENO Schemes. *Journal of Computational Physics*, 126(1):202–228, June 1996.
- [46] R. Borges, M. Carmona, B. Costa, and W. Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, 227(6):3191–3211, 2008.
- [47] B. C. Vermeire, F. D. Witherden, and P. E. Vincent. On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools. *Journal of Computational Physics*, 334:497–521, 2017.
- [48] B. Cockburn, S. Lin, and C. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems. *Journal of Computational Physics*, 84(1):90–113, September 1989.
- [49] B. Cockburn, S. Hou, and C. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. The multidimensional case. *Mathematics of Computation*, 54(190):545–581, 1990.

- [50] Z. J. Wang. Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids. Basic Formulation:. *Journal of Computational Physics*, 178(1):210–251, May 2002.
- [51] Y. Liu, M. Vinokur, and Z. J. Wang. Discontinuous Spectral Difference Method for Conservation Laws on Unstructured Grids. In C. Groth and D. W. Zingg, editors, *Computational Fluid Dynamics 2004*, pages 449–454. Springer Berlin Heidelberg, January 2006.
- [52] D. A. Kopriva. A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows. II. A Semi-Structured Method. *Journal of Computational Physics*, 128(2):475–488, October 1996.
- [53] H. T. Huynh. A Flux Reconstruction Approach to High-Order Schemes Including discontinuous Galerkin Methods. In *18th AIAA Computational Fluid Dynamics Conference*, volume AIAA 2007-4079. American Institute of Aeronautics and Astronautics, 2007.
- [54] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing*, volume 110. Benjamin/Cummings Redwood City, CA, 1994.
- [55] A. W. Brightmore O. Reynolds and W. H. Moorby. Papers on mechanical and physical subjects: The sub-mechanics of the universe. 1903.
- [56] K. Liu, Y. Lu, and C. You. High-order ALE method for the Navier–Stokes equations on a moving hybrid unstructured mesh using flux reconstruction method. *International Journal of Computational Fluid Dynamics*, 27(6-7):251–267, 2013.
- [57] H. Gao and Z. J. Wang. A High-order Lifting Collocation Penalty Formulation for the Navier-Stokes Equations on 2d Mixed Grids. In *19th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, June 2009.
- [58] H. T. Huynh. A Reconstruction Approach to High-Order Schemes Including discontinuous Galerkin for Diffusion. In *47th AIAA Aerospace Sciences Meeting including The*

New Horizons Forum and Aerospace Exposition. American Institute of Aeronautics and Astronautics, January 2009.

- [59] B. Cockburn and C. W. Shu. The Local discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, December 1998.
- [60] F. Bassi and S. Rebay. A High-Order Accurate discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations. *Journal of Computational Physics*, 131(2):267–279, March 1997.
- [61] T. Haga, H. Gao, and Z. J. Wang. A High-Order Unifying discontinuous Formulation for the Navier-Stokes Equations on 3d Mixed Grids. *Mathematical Modelling of Natural Phenomena*, 6(03):28–56, 2011.
- [62] J. S. Cagnone, B. C. Vermeire, and S. Nadarajah. A p-adaptive LCP formulation for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 233(1):324–338, 2013.
- [63] Z. J. Wang and H. Gao. A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids. *J. Comput. Phys.*, 228(21):8161–8186, November 2009.
- [64] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin Heidelberg, 2 edition, 1993.
- [65] C. A. Kennedy and M. H. Carpenter. Diagonally implicit Runge-Kutta methods for ordinary differential equations. a review. 2016.
- [66] R. Alexander. Diagonally implicit Runge-Kutta methods for stiff ODE’s. *SIAM Journal on Numerical Analysis*, 14(6):1006–1021, 1977.
- [67] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A.

- May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. <https://www.mcs.anl.gov/petsc>, 2019.
- [68] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.13, Argonne National Laboratory, 2020.
- [69] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.
- [70] S. Hedayati Nasab, C. A. Pereira, and B. C. Vermeire. Optimal runge-kutta stability polynomials for multidimensional high-order methods. *Journal of Scientific Computing*, 89(1):1–30, 2021.
- [71] E. Ching, Y. Lv, and M. Ihme. Entropy residual as a feature-based adaptation indicator for simulations of unsteady flow. In *54th AIAA Aerospace Sciences Meeting*, page 0837, 2016.
- [72] L. Shi and Z. J. Wang. Adjoint-based error estimation and mesh adaptation for the correction procedure via reconstruction method. *Journal of Computational Physics*, 295:261–284, 2015.
- [73] L. Wang and D. J. Mavriplis. Adjoint-based h-p adaptive discontinuous Galerkin methods for the 2D compressible Euler equations. *Journal of Computational Physics*, 228(20):7643–7661, 2009.
- [74] K. J. Fidkowski and P. L. Roe. An entropy adjoint approach to mesh refinement. *SIAM Journal on Scientific Computing*, 32(3):1261–1287, 2010.
- [75] C. Mavriplis. A posteriori error estimators for adaptive spectral element techniques. In

- Proceedings of the Eighth GAMM-Conference on Numerical Methods in Fluid Mechanics*, pages 333–342. Springer, 1990.
- [76] M. Kompenhans, G. Rubio, E. Ferrer, and E. Valero. Adaptation strategies for high order discontinuous Galerkin methods based on Tau-estimation. *Journal of Computational Physics*, 306:216–236, 2016.
 - [77] M. Kompenhans, G. Rubio, E. Ferrer, and E. Valero. Comparisons of p-adaptation strategies based on truncation-and discretisation-errors for high order discontinuous Galerkin methods. *Computers & Fluids*, 139:36–46, 2016.
 - [78] F. Naddei, M. de la Llave Plata, and V. Couaillier. A comparison of refinement indicators for p-adaptive discontinuous Galerkin methods for the Euler and Navier-Stokes equations. In *2018 AIAA Aerospace Sciences Meeting*, page 0368, 2018.
 - [79] N. P. Weatherill, M. J. Marchant, O. Hassan, and D. L. Marcum. Grid adaptation using a distribution of sources applied to inviscid compressible flow simulations. *International Journal for Numerical Methods in Fluids*, 19(9):739–764, 1994.
 - [80] S. J. Kamkar, A. M. Wissink, V. Sankaran, and A. Jameson. Combined feature-driven Richardson-based adaptive mesh refinement for unsteady vortical flows. *AIAA journal*, 50(12):2834–2847, 2012.
 - [81] R. Balasubramanian and J. C. Newman III. Comparison of adjoint-based and feature-based grid adaptation for functional outputs. *International journal for numerical methods in fluids*, 53(10):1541–1569, 2007.
 - [82] A. Jameson. Aerodynamic shape optimization using the adjoint method. *Lectures at the Von Karman Institute, Brussels*, 2003.
 - [83] H. R. Karbasian and B. C. Vermeire. Sensitivity analysis of chaotic dynamical systems using a physics-constrained data-driven approach. *Physics of Fluids*, 34(1):014101, 2022.

- [84] P. J. Blonigan and Q. Wang. Multiple shooting shadowing for sensitivity analysis of chaotic dynamical systems. *Journal of Computational Physics*, 354:447–475, 2018.
- [85] A. Ni and Q. Wang. Sensitivity analysis on chaotic dynamical systems by non-intrusive least squares shadowing (nilss). *Journal of Computational Physics*, 347:56–77, 2017.
- [86] Q. Wang, R. Hu, and P. Blonigan. Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. *Journal of Computational Physics*, 267:210–224, 2014.
- [87] H. R. Karbasian and B. C. Vermeire. Gradient-free aerodynamic shape optimization using large eddy simulation. *Computers & Fluids*, 232:105185, 2022.
- [88] H. R. Karbasian and B. C. Vermeire. Application of physics-constrained data-driven reduced-order models to shape optimization. *Journal of Fluid Mechanics*, 934, 2022.
- [89] T. Phillips and C. Roy. Residual methods for discretization error estimation. In *20th AIAA Computational Fluid Dynamics Conference*, page 3870, 2011.
- [90] G. Rubio, F. Frayssé, D. A. Kopriva, and E. Valero. Quasi-a priori truncation error estimation in the dgsem. *Journal of Scientific Computing*, 64(2):425–455, 2015.
- [91] Y. Lv, Y. See, and M. Ihme. An entropy-residual shock detector for solving conservation laws using high-order discontinuous galerkin methods. *Journal of Computational Physics*, 322:448–472, 2016.
- [92] F. Bassi, A. Colombo, A. Crivellini, K. J. Fidkowski, M. Franciolini, A. Ghidoni, and G. Noventa. Entropy-adjoint p-adaptive discontinuous Galerkin method for the under-resolved simulation of turbulent flows. *AIAA Journal*, 58(9):3963–3977, 2020.
- [93] L. Wang, M. K. Gobbert, and M. Yu. A dynamically load-balanced parallel p-adaptive implicit high-order flux reconstruction method for under-resolved turbulence simulation. *Journal of Computational Physics*, 417:109581, 2020.
- [94] R. Ghoreishi and B. C. Vermeire. Vorticity-based polynomial adaptation for moving and deforming domains. *Computers & Fluids*, 231:105160, 2021.

- [95] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. 1997.
- [96] C. A. Pereira. *Analysis of High-Order Element Types for Implicit Large Eddy Simulation*. PhD thesis, Concordia University, 2019.
- [97] Z. J. Wang, K. Fidkowski, R. Abgrall, D. Caraeni, F. Bassi, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P. Persson, B. van Leer, and M. Visbal. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.
- [98] F. D. Witherden, A. M. Farrington, and P. E. Vincent. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the Flux Reconstruction approach. *Computer Physics Communications*, 185(11):3028–3040, 2014.
- [99] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *European Parallel Virtual Machine/Message Passing Interface Users’ Group Meeting*, pages 97–104. Springer, 2004.
- [100] C. H. K. Williamson. Vortex dynamics in the cylinder wake. *Annual review of fluid mechanics*, 28(1):477–539, 1996.
- [101] O. Inoue and N. Hatakeyama. Sound generation by a two-dimensional circular cylinder in a uniform flow. *Journal of Fluid Mechanics*, 471, November 2002.
- [102] K. George and K. Vipin. METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. <http://www.cs.umn.edu/~metis>, 2009.
- [103] P. E. Vincent, P. Castonguay, and A. Jameson. Insights from von Neumann analysis of high-order flux reconstruction schemes. *Journal of Computational Physics*, 230(22):8134–8154, September 2011.

- [104] W. Trojak, R. Watson, A. Scillitoe, and P. G. Tucker. Effect of mesh quality on flux reconstruction in multi-dimensions. *Journal of Scientific Computing*, 82(3):1–36, 2020.
- [105] C. A. Pereira and B. C. Vermeire. Spectral properties of high-order element types for implicit large eddy simulation. *Journal of Scientific Computing*, 85(2):1–38, 2020.
- [106] A. Uranga, P. Persson, M. Drela, and J. Peraire. Implicit Large Eddy Simulation of transition to turbulence at low Reynolds numbers using a discontinuous Galerkin method. *International Journal for Numerical Methods in Engineering*, 87(1-5):232–261, 2011.
- [107] M. R. López, A. Sheshadri, J. R. Bull, T. D. Economon, J. Romero, J. E. Watkins, D. M. Williams, F. Palacios, A. Jameson, and D. E. Manosalvas. Verification and Validation of HiFiLES: a High-Order LES unstructured solver on multi-GPU platforms. In *32nd AIAA Applied Aerodynamics Conference*, number AIAA 2014-3168. American Institute of Aeronautics and Astronautics, 2014.
- [108] B. C. Vermeire. Paired explicit Runge-Kutta schemes for stiff systems of equations. *Journal of Computational Physics*, 393:465–483, 2019.
- [109] B. Cockburn, S. Y. Lin, and C. W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems. 1988.
- [110] B. Cockburn and C. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Mathematics of Computation*, 52(186):411–435, 1989.
- [111] B. Cockburn, G. E. Karniadakis, and C. W. Shu. The development of discontinuous Galerkin methods. In *Discontinuous Galerkin Methods*, pages 3–50. Springer, 2000.
- [112] Y. Liu, M. Vinokur, and Z. J. Wang. Spectral difference method for unstructured grids I: basic formulation. *Journal of Computational Physics*, 216(2):780–801, 2006.

- [113] C. Carton de Wiart and K. Hillewaert. A discontinuous Galerkin method for implicit LES of moderate Reynolds number flows. In *53rd AIAA Aerospace Sciences Meeting*, page 0055, 2015.
- [114] C. Carton de Wiart, K. Hillewaert, L. Bricteux, and G. Winckelmans. Implicit LES of turbulent flows using a discontinuous Galerkin method. In *6th European Conference on Computational Fluid Dynamics (ECFD VI)*, 2014.
- [115] S. Gremmo, C. Carton de Wiart, B. Gorissen, K. Hillewaert, G. Winckelmans, G. Coussement, and L. Bricteux. Implicit LES of turbulent flows with a high order discontinuous Galerkin method. In *APS Meeting Abstracts*, 2012.
- [116] T. Bolemann, A. Beck, D. Flad, H. Frank, V. Mayer, and C.D. Munz. High-order discontinuous Galerkin schemes for large-eddy simulations of moderate Reynolds number flows. In *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, pages 435–456. Springer, 2015.
- [117] B. J. Zimmerman, Z. J. Wang, and M. R. Visbal. High-order spectral difference: verification and acceleration using GPU computing. In *21st AIAA Computational Fluid Dynamics Conference*, page 2941, 2013.
- [118] F. D. Witherden, A. M. Farrington, and P. E. Vincent. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications*, 185(11):3028–3040, 2014.
- [119] C. Liang, K. Miyaji, and B. Zhang. An efficient correction procedure via reconstruction for simulation of viscous flow on moving and deforming domains. *Journal of Computational Physics*, 256:55–68, 2014.
- [120] M. Yu and L. Wang. A high-order flux reconstruction/correction procedure via reconstruction formulation for unsteady incompressible flow on unstructured moving grids. *Computers & Fluids*, 139:161–173, 2016.
- [121] T. Sarpkaya. Vortex-induced oscillations: a selective review. 1979.

- [122] P. W. Bearman. Vortex shedding from oscillating bluff bodies. *Annual review of fluid mechanics*, 16(1):195–222, 1984.
- [123] R. E. D. Bishop and A. Hassan. The lift and drag forces on a circular cylinder oscillating in a flowing fluid. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 277(1368):51–75, 1964.
- [124] H. M. Blackburn and R. D. Henderson. A study of two-dimensional flow past an oscillating cylinder. *Journal of Fluid Mechanics*, 385:255–286, 1999.
- [125] E. Guilmineau and P. Queutey. A numerical simulation of vortex shedding from an oscillating circular cylinder. *Journal of Fluids and Structures*, 16(6):773–794, 2002.
- [126] C. H. Williamson. Defining a universal and continuous Strouhal-Reynolds number relationship for the laminar vortex shedding of a circular cylinder. *The Physics of fluids*, 31(10):2742–2744, 1988.
- [127] X. Y. Lu and C. Dalton. Calculation of the timing of vortex formation from an oscillating cylinder. *Journal of Fluids and Structures*, 10(5):527–541, 1996.
- [128] L. W. Carr. Progress in analysis and prediction of dynamic stall. *Journal of aircraft*, 25(1):6–17, 1988.
- [129] C. P. Ellington, C. Van Den Berg, A. P. Willmott, and A. L. Thomas. Leading-edge vortices in insect flight. *Nature*, 384(6610):626, 1996.
- [130] K. Jones, M. Platzer, K. Jones, and M. Platzer. Numerical computation of flapping-wing propulsion and power extraction. In *35th Aerospace Sciences Meeting and Exhibit*, page 826, 1997.
- [131] S. Wang, X. Zhang, G. He, and T. Liu. Evaluation of lift formulas applied to low-Reynolds-number unsteady flows. *AIAA Journal*, 53(1):161–175, 2014.
- [132] J. Choi, T. Colonius, and D. R. Williams. Surging and plunging oscillations of an airfoil at low Reynolds number. *Journal of Fluid Mechanics*, 763:237–253, 2015.

- [133] H. R. Karbasian, J. A. Esfahani, and E. Barati. Effect of acceleration on dynamic stall of airfoil in unsteady operating conditions. *Wind Energy*, 19(1):17–33, 2016.
- [134] H. R. Karbasian and K. C. Kim. Numerical investigations on flow structure and behavior of vortices in the dynamic stall of an oscillating pitching hydrofoil. *Ocean Engineering*, 127:200–211, 2016.
- [135] J. A. Esfahani, E. Barati, and H. R. Karbasian. Fluid structures of flapping airfoil with elliptical motion trajectory. *Computers & Fluids*, 108:142–155, 2015.
- [136] J. H. Strickland, B. T. Webster, and T. Nguyen. A vortex model of the Darrieus turbine: an analytical and experimental study, Sandia National Laboratories Report. Technical report, SAND 79-7058, 1980.
- [137] M. V. Ol, L. Bernal, C. Kang, and W. Shyy. Shallow and deep dynamic stall for flapping low Reynolds number airfoils. In *Animal Locomotion*, pages 321–339. Springer, 2010.
- [138] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.
- [139] A. Roshko. On the development of turbulent wakes from vortex streets. 1953.
- [140] M. S. Bloor. The transition to turbulence in the wake of a circular cylinder. *Journal of Fluid Mechanics*, 19(2):290–304, 1964.
- [141] C. Norberg. Flow around a circular cylinder: aspects of fluctuating lift. *Journal of fluids and structures*, 15(3-4):459–469, 2001.
- [142] M. Braza, P. H. H. M. Chassaing, and H. H. Minh. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *Journal of fluid mechanics*, 165:79–130, 1986.
- [143] M. Zhao, L. Cheng, and T. Zhou. Direct numerical simulation of three-dimensional flow past a yawed circular cylinder of infinite length. *Journal of Fluids and Structures*, 25(5):831–847, 2009.

Appendix A

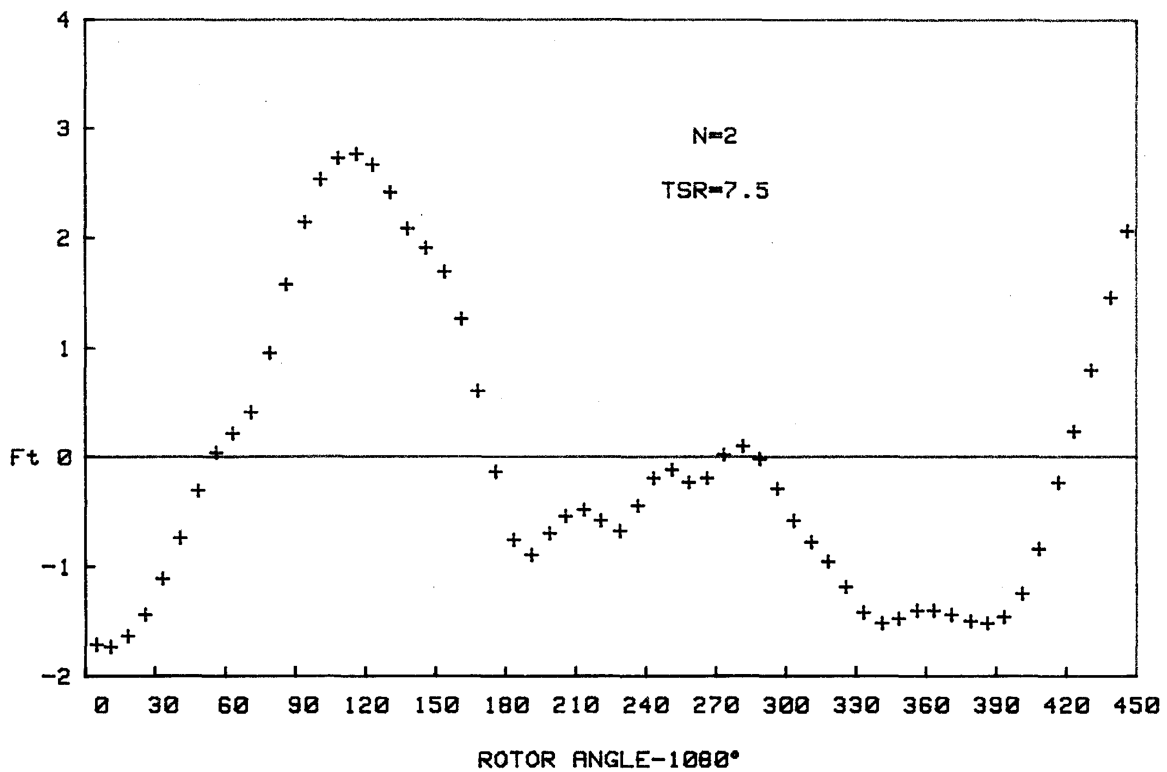
DLB: Element Weights

Table A.1. Element weight W_E defined for different element types and degree polynomials, non-dimensionalized based on the weight of a $K = 1$ triangular element.

Element Type	Degree	W_E
Triangular	$K = 1$	1.00
	$K = 2$	1.55
	$K = 3$	2.10
	$K = 4$	2.81
	$K = 5$	3.61
Quadrilateral	$K = 1$	1.73
	$K = 2$	2.40
	$K = 3$	3.14
	$K = 4$	4.50
	$K = 5$	5.66
Tetrahedral	$K = 1$	3.27
	$K = 2$	5.97
	$K = 3$	10.59
	$K = 4$	16.95
	$K = 5$	26.12
Prism	$K = 1$	6.87
	$K = 2$	12.06
	$K = 3$	21.49
	$K = 4$	38.53
	$K = 5$	65.55
Hexahedral	$K = 1$	11.95
	$K = 2$	19.46
	$K = 3$	35.80
	$K = 4$	73.41
	$K = 5$	132.57

Appendix B

Experimental Tangential Force Coefficient [1]



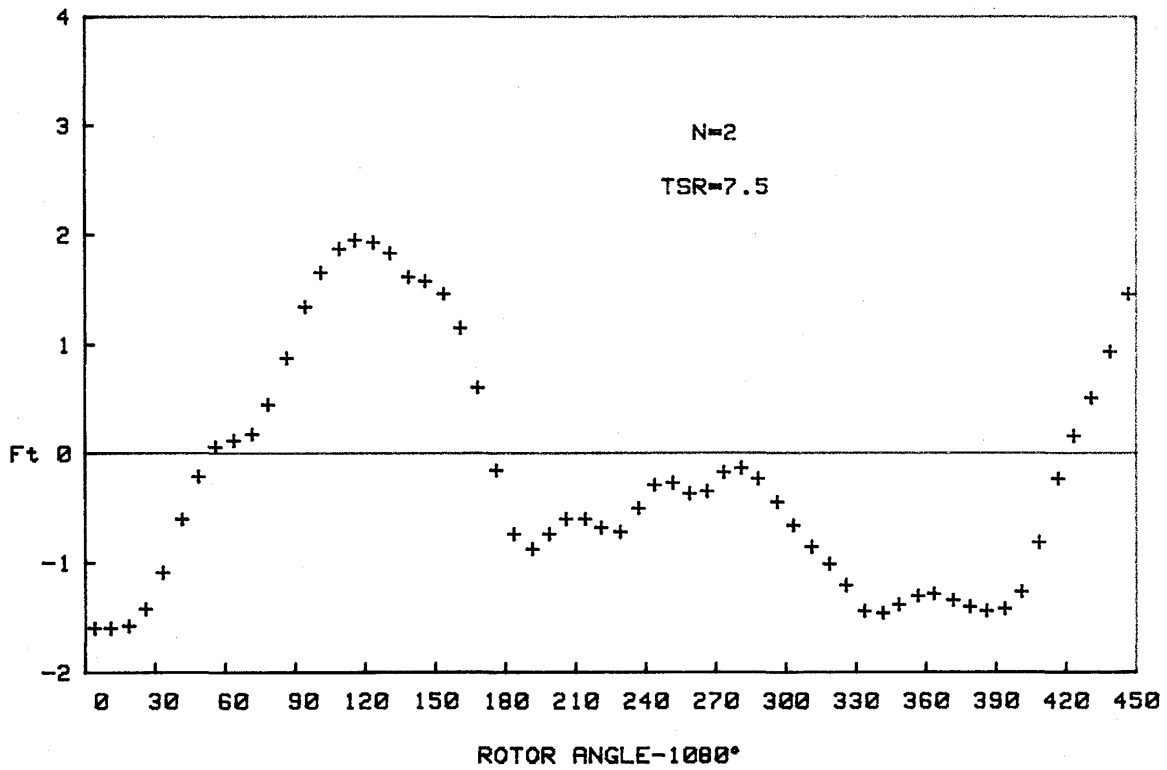
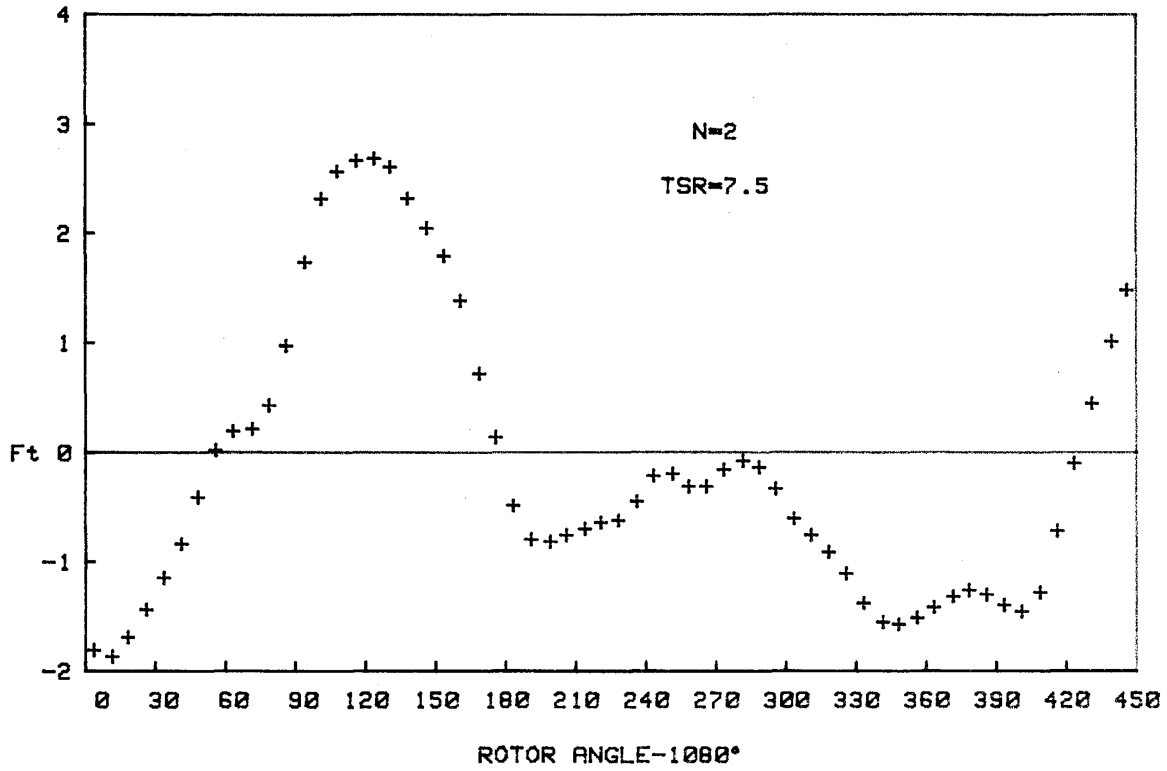


Figure B.1. Tangential force coefficient profiles corresponding to the 4th cycle of the experimental study performed by Strickland et al. [1] for a blade of a two-bladed wind turbine at a tip speed ratio of 0.75 with $\alpha_0 = 0$.