To the University of Wyoming:

The members of the Committee approve the dissertation of Andrew C. Kirby presented on February 9, 2018.

Dr. Dimitri J. Mavriplis, Chairperson

Dr. Victor E. Ginting, External Department Member

Dr. Jonathan W. Naughton

Dr. Jayanarayanan Sitaraman

Dr. Marc W. Spiegelman, Columbia University

APPROVED:

Dr. Carl P. Frick, Head, Department of Mechanical Engineering

Dr. Michael V. Pishko, Dean, College of Engineering and Applied Sciences

Kirby, Andrew C., Enabling High-Order Methods for Extreme-Scale Simulations, Ph.D., Department of Mechanical Engineering, February, 2018.

With the continued growth of computational resources, the development of high-order methods for computational fluid dynamics (CFD) has become an important track for obtaining high performance on new computer architectures and obtaining high-fidelity solutions. This work advances the discontinuous Galerkin (DG) method for extreme-scale simulation through the development of a discretization that is robust, highly computationally efficient, highly parallel scalable, and suitable for simulation of multiscale problems.

To enhance numerical stability of the DG method, the discretization is reformulated using split form flux formulations while possessing the summation-by-parts (SBP) property. The split form DG method with SBP is demonstrated to have superior robustness in comparison to the traditional DG method for both inviscid and viscous flow problems. The discretization is developed to be highly efficient by way of collocated tensor-product basis functions restricted to Cartesian mesh systems with explicit time stepping via Runge-Kutta methods. Dynamic adaptive mesh refinement (AMR) is instrumented via an octree-based AMR framework allowing for multiscale problems to be simulated. The dynamic AMR incorporates both mesh adaption (h-refinement) and solution order enhancement (p-enrichment) to form an hp refinement strategy.

For simulation of fluid dynamics problems containing complex geometries, the development and instrumentation of the DG method into a larger computational framework employing a multi-solver, multi-mesh paradigm with overlapping grids is pursued. The computational framework instruments a *near-body*, *off-body* mesh system through a dynamic overset framework. The near-body mesh uses unstructured grid technologies to accurately capture detailed geometries of the bodies of study, and the off-body mesh is responsible for capturing unsteady turbulent features using dynamic adaptive structured grids.

Applications from aerospace and wind energy are targeted to demonstrate the ability of the high-order discretization embedded in the multi-solver, multi-mesh framework. Blade-resolved simulations of wind energy applications are presented. Simulations of individual wind turbines are studied for accurate prediction. Learned solution strategies such as the number of mesh nodes required to accurately capture integrated forces on a wind turbine are used for baseline to simulate full wind farms incorporating up to 100 wind turbines.

ENABLING HIGH-ORDER METHODS FOR EXTREME-SCALE SIMULATIONS

by

Andrew C. Kirby,
M.S. Applied Mathematics, Columbia University (2013)
B.S. Mathematics, University of Wisconsin-Madison (2011)

A dissertation submitted to the
Department of Mechanical Engineering
and the
University of Wyoming
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
MECHANICAL ENGINEERING

Laramie, Wyoming
February 2018

To Elizabeth J. Gilbert (1951-2016)

Thank you for sharing your light and inspiring my journey to be a lifelong learner.

Contents

List of Figures

List of Tables

List of Computer Programs

Acknowledgments

First and foremost, I am extremely grateful for my advisor Professor Dimitri Mavriplis. His advisement and guidance have helped shape me to be a better researcher and scholar. Thank you for all of your contributions; this would not be possible without you. I really enjoyed attending conferences where you presented our work; attendees would fill the room and even spill out of the doorway. This demonstrated to me that our work was truly important which inspired me to do my best and be a positive representative of our group and research.

Second, I would like to thank my colleagues in the High Altitude CFD Laboratory, members of the Wind Energy Research Center, and members of the Mechanical Engineering Department. Specifically, I want to highlight those who provided the time and patience to help me with my research: Dr. Jonathan Naughton, Dr. Michael Stoellinger, Dr. Michael Brazell, Dr. Zhi Yang, Dr. Behzad Ahrabi, Dr. Asitav Mishra, Arash Hassanzadeh, and Rajib Roy. Additionally, I would like to thank members of the HELIOS team at NASA Ames Research Center for their contributions and insights to this work: Dr. Jay Sitaraman and Dr. Andrew Wissink. I extend my gratitude to my committee members for their time to help guide my research path and provide comments on this work.

I am thankful for my family and friends for supporting my academics. My parents have always provided the support needed for success. I would also like to thank my extended family for their support during my undergraduate years, specifically Heidi and Troy, and Mary and Craig. I would like to thank my closest friends who helped drive my academics and fun times since primary school; thanks Ed and JP. Lastly, I am so thankful for my fiancée, Dr. Rabia Tugce Yazicigil. You let me move across the country for half a decade to pursue my Ph.D. while providing support for our relationship and my education; thank you.

I would like to thank the institutions for the financial and computational resource support during the course of this research. This work was supported in part by the NSF Blue Waters Graduate Fellowship as part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993). Additionally, this work was supported in part by Office of Naval Research Grants N00014-14-1-0045 and N00014-16-1-2737, and by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award DE-SC0012671. Computational resources were provided by the NCAR-Wyoming Supercomputer Center (NWSC), the University of Wyoming Advanced Research Computing Center (ARCC), and the NSF Blue Waters sustained-petascale computing project. This research was partially conducted with an Accelerated Science Discovery (ASD) project which provided early access to the NWSC-2 Cheyenne supercomputer.

Andrew C. Kirby

University of Wyoming

February 2018

Chapter 1

Introduction

Computational fluid dynamics has become an essential tool in science and engineering by enabling scientists and engineers the ability to perform numerical experiments in a virtual laboratory to provide numerical prediction of complex flow phenomenon. The field of computational fluid dynamics has made significant advancements over the past three decades via traditional numerical discretization methods in numerical analysis such as finite-difference and finite-volume methods. However, as high performance computing resources continue to grow exponentially through advanced architecture design, these traditional methods fail to fully utilize the available computational capabilities, thus resulting in lost science and hindering the advancement of new science.

High-order finite-element methods allow for the effective use of computational resources but have previously suffered from high computational cost and robustness issues. Recent advancements in numerical discretization design of these methods have provided viable pathways for higher computational efficiency and improved robustness, which make these methods favorable candidates as the successors to the traditional discretization methods.

This work advances algorithmic design of high-order discretizations to innovate a robust high-order finite-element method that leverages high performance computing with superior performance, and is developed inside an adaptive mesh refinement framework to enable extreme-scale numerical simulations found in aerospace and wind energy applications.

1.1 Historical Perspective of Computing

Computational resources have grown continuously at an exponential rate over the past half century following Moore's Law [1] enabling computational science to become a third pillar of scientific discovery [2]. Supercomputers emerged in the 1960s starting with the CDC 6600 [3] in 1964. The first supercomputers were designed to operate at *fast* processor speeds. For example, the CDC 6600 operated at 40 MHz, executing approximately three million floating point operations per second (FLOPS), or three mega-FLOPS (MFLOPS), which was 10 times faster than traditional computers at that time. Continuous development through increasing processor speeds guided the supercomputer landscape through the early 1970s.

Vector supercomputers [4] came onto the high performance computing (HPC) scene in the mid-1970s and dominated supercomputer design until the early 1990s. Vector machines allowed for arithmetic operations to be applied to large data sets that were stored in one-dimensional arrays, known as vectors. This processing technique allowed for computations to be parallelized through a single instruction, multiple data (SIMD) approach where multiple computations could be performed simultaneously. The Cray-1 [5] supercomputer outperformed every computer in the world in the mid- to late-1970s using this approach.

The only competitor to the Cray-1 in terms of performance in the 1970s was the ILLIAC IV [6] supercomputer. The ILLIAC IV was the first *massively parallel* supercomputer meaning it was composed of 64 floating point units linked together using a single central processing unit. This approach overtook vector-processing supercomputers in the early 1990s. In the 1985, Alan Karp of IBM proposed a challenge to the HPC community to break the then-hypothesized speedup barrier of 100[†], and offered a prize of $100. Gordon Bell of the National Science Foundation believed Karp's challenge was a good idea, but believed no one would succeed. In 1986, the challenge was met with a flourish of solutions utilizing distributed memory and distributed parallelism [7]. In 1987, to keep it interesting, Bell added to the prize with his own money, increasing the winnings to $1,000, for the best speedup for a real application on a real machine leading to the start of the Gordon Bell

[†]Some members of the HPC community hypothesized that Amdahl's Law fundamentally limited speedup to a maximum of 100 [7].

Prize competition [8]. Researchers from Sandia National Laboratory smashed the speedup record demonstrating speedups in the range of 400-600 using a 1024-node nCUBE machine, and further demonstrated a speedup of 1,000 if the problem size was scaled with the number of processors [8]. This breakthrough has driven the supercomputing design and algorithm development for the last 30 years.

HPC systems have reintroduced the concept of vector processing through SIMD or SIMT (single instruction, multiple threads) combined with distributed computing to increase computational efficiency and decrease energy usage through improvement of the number of FLOPs per watt. This has lead to heterogeneous computing systems in the last 10 years which are composed of more than one kind of processor or computing device. For example, such systems are constructed of nodes with a CPU-based host and Graphical Processing Units (GPU) on each node.

The fastest supercomputers today are likely to be heterogeneous systems requiring development of simulation software to target specific computing architectures [9]. Hardware complexity also introduces software complexity. To best leverage the computing capabilities, developers must expose multiple levels of granularity in algorithm parallelism and be keenly aware of data movement. The majority of the computing operations are performed on the coprocessor which usually incorporates specialized arithmetic processing capabilities that are massively parallel but are relatively slow in processing speed. Thus, the algorithm must be able to have lots of parallelism not only through domain decomposition but also at the loop level. Data used for the computation must be minimally moved as data transfers are much more costly than arithmetic computations.

1.2 Computational Fluid Dynamics

The continuous growth of HPC has bolstered the field of computational fluid dynamics (CFD) allowing for numerical simulation of complex flow processes encountered in engineering. The numerical discretizations of the governing equations in CFD have traditionally been founded in finite-difference and finite-volume methods. Finite-difference [10–12] methods are developed in the context of structured meshes for simple geometries, and the computational kernel is based on a Taylor-series expansion composed of a stencil of points. For higher-order discretizations, more stencil points are required. Thus, the computational kernel is not compact nor localized to a single mesh element. This adds complexity to the parallel communication patterns and limits computational performance by forming a data communication bottleneck via limited memory bandwidth of the hardware. Finite-volume [11, 13, 14] methods, alternatively, allow for arbitrary geometries by a control-volume formulation, and can be easily formulated to be discretely conservative for problems where conservation is important. For higher-order finite-volume discretizations, more mesh elements are required for the numerical stencil similar to finite-difference methods. However, most finite-volume discretizations are formulated using a lower-order discretization, traditionally second order. For capturing small-scale flow features, such as unsteady turbulent wakes, extensively refined meshes are required increasing the mesh-generation and computational complexities.

1.2.1 High-Order Finite-Element Methods

An alternative approach to the traditional discretizations is the area of higher-order methods. Development of high-order accurate discretizations for computational aerodynamics has been pursued vigorously over the last decade and substantial advances have been demonstrated for continuous and discontinuous Galerkin (DG) methods [15–23]. However, the use of high-order methods for computational aerodynamics remains a research topic largely due to the poor robustness and large computational expense of these methods [24]. The use of a discontinuous Galerkin discretization offers advantages over traditional finite-difference or finite-volume discretizations in terms of accuracy per degree of freedom, which in turn

enables the use of coarser grids (assuming the underlying functions are well approximated by high-order function spaces), thus minimizing the overheads associated with managing mesh-related processes such as dynamic adaptive mesh refinement. Additionally, the nearest neighbor stencil of the DG discretization simplifies the treatment of fringe data in transition regions between coarse and fine mesh blocks. Further, the compact computational kernel, which is confined locally to a single mesh element, allows for efficient use of the newer computing hardware as the data layout forms a natural cache block [25].

To further enhance computational performance, high-order methods on Cartesian meshes allow the potential for effectively leveraging the power of emerging computer architectures [26,27]. The use of Cartesian grids in computational fluid dynamics provides well known advantages in terms of computational efficiency and accuracy. Cartesian meshes represent the simplest grid structure which can be represented extremely compactly, thus minimizing solver memory footprints, optimizing parallel efficiency, and simplifying both adaptive mesh refinement implementations [28–30] and overset mesh search and interpolation tasks [31,32]. The principal drawback of Cartesian mesh approaches lies in the difficulties of dealing with non-simple geometries. Various approaches for dealing with complex geometries have been developed for use with Cartesian meshes including immersed boundary methods [33–35], cut cell approaches [36–38], and overlapping dual mesh paradigms where a body fitted mesh is used in near-body regions and a Cartesian mesh is used in off-body regions [31, 32, 39]. In particular, this work employs a high-order discontinuous Galerkin finite-element discretization in the off-body region. DG methods and similar FEM solvers have been deployed for a variety of aerospace applications [15–23, 40, 41] including the use of adaptive mesh and solution refinement techniques [42–44].

The goal of this work is to capitalize on the inherent advantages of Cartesian meshes to develop an efficient and highly accurate discontinuous Galerkin solver. The basic approach consists of limiting the discretization to a collocation approach using a tensor-product basis formulation on hexahedral elements, which is well known to provide large gains in efficiency over the general element modal formulation [41,45]. Numerous examples of high-order tensor-product implementations on structured meshes have been described previously [41, 45, 46].

1.2.2 Adaptive Mesh Refinement

The dynamic adaptive mesh refinement approach was initially developed as early as 1982 by Berger and Oliger [47] to solve systems of hyperbolic conservation laws. For many time-dependent problems, the flow is generally smooth in most regions of the computational domain except in small portions. Fine mesh resolution is only required in regions needed to capture the small details of the flow, thus placing the resources only where needed and computationally advantageous. The use of uniformly refined meshes quickly becomes intractable for multiscale problems containing small flow scales but over large computational domains.

Conical problems that require adaptive mesh refinement are typically found in astrophysics where simulations of star formation and evolution of galaxies are performed. Additionally, combustion simulation lends itself well to adaptive mesh refinement involving detonation requiring the reaction zones be resolved. The combination of discontinuous Galerkin methods and adaptive mesh refinement has been studied in many fields such as astrophysical hydrodynamics [48], atmospheric modeling [49], and global tsunami events [50,51]. Additionally, combined h- (mesh refinement) and p- (solution order enrichment) adaptive methods have been shown to be optimal for error reduction of the numerical solution [52].

Aerodynamics problems involving rotorcraft where the need to resolve the wake are very suitable applications for adaptive mesh refinement techniques. The tip-vortices generated by the rotor blades propagate and hit other blades as well as the rotorcraft fuselage creating aerodynamic forces. The use of adaptive mesh refinement for rotorcraft simulation has been successfully implemented by the CREATE-AV HELIOS [39, 53, 54] solver. Additional aerodynamics problems that are very suitable for dynamic adaptive mesh refinement are the simulations of wind farms that resolve the the aerodynamic boundary layer on the wind turbine blades as well as the atmospheric flow conditions. This problem can easily span 10 or more orders of magnitude of spatial scales. For uniform mesh refinement, this equates to 100 billion mesh elements in each spatial direction! That would require nearly 24 billion yottabytes[†] just to store the mesh point coordinates.

[†]Yotta-10^{24}; Zetta-10^{21}; Exa-10^{18}

1.3 Wind Energy Applications

The primary target application of this work is concerned with wind energy. Wind energy is becoming an emergent renewable energy source throughout the United States and the world. Wind energy costs have drastically dropped over the last decade through advanced design and increased scale of turbines, thus making wind energy a desirable renewable alternative to fossil-fuel-based energies. It is estimated that wind energy could produce as much as 20% of the total electrical energy needs by 2030 and 35% by 2050 in the United States, which will have a profound economic and societal impact [55]. The transition from fossil fuels to renewable energies will strengthen energy security and reduce greenhouse-gas emissions. Improved understanding will enable innovation and improved blade-turbine-tower design, better wind turbine placement in wind farm configurations which will increase wind plant efficiency, improve wind turbine lifespan, and decrease wind energy costs. This can produce large economic impacts particularly for wind plants containing a few hundred multi-mega-watt turbines. Predictive simulations of wind plants in complex terrains has ushered in the need for exascale-enabled simulations. Two organizations in the Department of Energy (DoE), the Office of Science and the National Nuclear Security Administration, have formed a collaborative effort to establish the Exascale Computing Project (ECP) [56]. The ECP was established to maximize the benefits of high performance computing and accelerate the development of a capable exascale computing ecosystem. A part of the ECP, the project "Exascale Predictive Wind Plant Flow Physics Modeling" has been formed to advance the understanding of wind plant flow physics such as wake dynamics, complex terrain effects, and turbine-turbine interactions [57]. The primary objective of this wind plant modeling project is to develop an exascale-capable system software application that will accurately simulate a wind plant containing on the order of 100 wind turbines within a 10 km by 10 km area consisting of complex terrain [58]. An estimate of 100 billion degrees of freedom will be required to simulate this problem.

Understanding the aerodynamics of the wind turbine is an essential aspect for energy production optimization, not only for the individual turbine but also for the complete wind farm. Exploration of wind turbine yawing [59–62] for wind farm optimization introduces

complex aerodynamics and possible structural effects. These complex aerodynamics, such as flow separation, cannot be captured accurately using lower-fidelity methods. High-fidelity blade-resolved simulations are required for accurate prediction, thus state-of-the-art modeling techniques of wind plants are transitioning from reduced-fidelity models such as turbine parameterization techniques, e.g. actuator lines [60, 63–65] or actuator discs [65–67], to high-fidelity blade-resolved models [68–82].

High-fidelity simulations require the use of a full-rotor model where the detailed geometry of the turbine blade and tower is used. These models were previously computationally prohibitive until recent advancements in HPC technologies. The present day leadership class supercomputing environment includes systems containing on the order of 1 million to 20 million computing cores [9]. Wind plant simulations using full-rotor models have recently been applied using the CREATE-AV HELIOS [39,53,54] software. HELIOS uses a multiple-mesh, multiple-solver paradigm with an overset framework. A computational study using HELIOS for wind turbine simulation was performed by Gundling *et al.* [83]. In the work of Sitaraman *et al.* [39], HELIOS was used for a blade-resolved wind plant simulation containing 48 wind turbines under ideal and atmospheric conditions using 3,840 CPU cores. The full rotor model mesh in that work contained just under 475,000 nodes per blade and the tower mesh contained approximately 500,000 nodes. The 48 wind turbine plant equated to approximately 96 million near-body mesh points and the off-body adaptive mesh system grew from 50 million to 180 million nodes giving a grand total of nearly 280 million degrees of freedom (DOFs). Those results demonstrated the ability to simulate an entire wind plant using a full-rotor model in an overset framework using multiple meshes and multiple flow solvers. However, coarse meshes for the wind turbine models were used which was unable to accurately capture the integrated forces.

The goal of this work is to make advancements toward the exascale grand challenge problem of simulating wind plants using full-rotor models in complex terrain environments under atmospheric inflow conditions at high resolution. To perform this task, appropriate physics, numerical solvers, and scalability on large high performance computing systems are required. The approach herein involves an analogous simulation environment to the

HELIOS [53] software through a computational overset framework using a multiple-mesh, multiple-solver paradigm. This approach of overlapping grids has been utilized in several works [53, 84–87]. Within this framework, a near-body, off-body mesh philosophy is employed. The near-body mesh system is designed to handle complex geometries by using unstructured meshes and the off-body mesh system is designed to use dynamically adaptive Cartesian meshes for enabling flow feature tracking with high levels of solution accuracy.

1.4 Dissertation Overview

In this work we seek to develop a higher-order numerical method that is robust, highly computationally efficient and parallel scalable, and able to be effectively utilized for extreme-scale problems prescribed by aerospace and wind energy applications. First the development of a computationally efficient discontinuous Galerkin method is conducted, followed by modification of the original discretization to enhance the overall numerical stability. To enable extreme-scale simulations, implementation of the discretization into an adaptive mesh refinement framework is performed allowing for simulation of multiscale problems. The adaptive high-order method is then instrumented into a larger computational framework which is used to simulate complex problems from the aerospace and wind energy fields. Within the wind energy application, multiple single turbines are simulated and verified, followed by simulation of full wind farms containing multiple turbines.

1.5 Dissertation Outline

Chapter 2	Introduction of the discontinuous Galerkin method and exploration of the high-order and robustness characteristics
Chapter 3	Development of the split form discontinuous Galerkin method with the summation-by-parts property to enhance robustness
Chapter 4	Instrumentation of the method into an adaptive mesh refinement framework and validation of numerical discretization
Chapter 5	Introduction of the computational framework used for large-scale simulations founded in aerospace and wind energy applications
Chapter 6	Investigation of single wind turbine problems is performed for informing simulation settings for wind farm simulations
Chapter 7	Demonstration of full wind farm simulations is performed including a study of parallel weak scalability of the full computational framework
Chapter 8	Conclusions of the work are drawn and future directions are outlined

Chapter 2

Discontinuous Galerkin Methods

The discontinuous Galerkin (DG) method combines numerical approaches from finite-element and finite-volume methods. The solution is discretized into a polynomial representation local to a mesh element. At mesh element interfaces, solutions are discontinuous thus requiring treatment similar to finite-volume methods via an upwinding flux approach. This element-wise discontinuity characteristic makes the discontinuous Galerkin method an ideal candidate for adaptive solution techniques such mesh adaptation, known as h-adaption, where the mesh is locally refined, and solution enhancement, known as p-adaption, where the degree of the approximating polynomial is increased locally. These two properties form the method known as the hp finite-element method (hp-FEM). The hp finite-element method has been shown to be optimal for error reduction in solution approximation [52]. Additionally, the discontinuous property requires only a nearest-neighbor communication pattern of the numerical fluxes thus simplifying parallel computing implementation of the method.

The use of a discontinuous Galerkin method offers advantages over traditional finite-difference and finite-volume methods through increased solution order per mesh element. This increased solution order enables the use of coarser meshes, thus minimizing the overhead cost associated with managing dynamic adaptive mesh refinement processes. Further, the increased order is achieved by introducing more degrees-of-freedom per mesh element and increasing the floating-point operations per degree-of-freedom. By raising the floating-point operations per degree-of-freedom, this increases the arithmetic intensity which is the

ratio of floating-point operations to computer memory accesses required to perform those operations. Modern computer architectures can perform floating-point operations significantly faster than the rate at which memory can be read, thus favoring higher arithmetic intensity computations. Finite-difference and finite-volume methods have low arithmetic intensity, which classify them as *memory-bound* algorithms, reducing their computational efficiency. Discontinuous Galerkin methods at higher orders of solution approximation are *compute-bound* in comparison, thus, in theory, are faster per residual evaluation for problems with the same numbers of degrees-of-freedom. This will be demonstrated in this chapter by comparison of computational efficiency of the discontinuous Galerkin method and the finite-difference method.

The discontinuous Galerkin method has many discretization variations by choice of solution expansion or basis functions, which are used to represent the solution in the discretized equations. The expansion basis functions can be either of *modal* or *nodal* type. The choice of the expansion functions can substantially impact the computational efficiency of the method. This work utilizes nodal expansion functions composed of tensor-products of one-dimensional polynomials. Particularly, the polynomials are chosen to be Lagrange interpolating functions constructed to give a Kronecker-delta property leading to a collocation method: the solution points are chosen to be the quadrature points for numerical integration. Analysis of this method will highlight the design choice on this work.

This chapter presents the high-order discontinuous Galerkin discretization for explicit unsteady flow problems governed by the compressible Navier-Stokes equations. Introduction of tensor-product basis formulations and comparison to traditional generalized finite-element methods are highlighted. Further, arithmetic reductions through collocation of solution and integration points and a Cartesian mesh setting reduce the computational work. Results show the p-degree discretization achieves a $p+1$ order-of-accuracy asymptotic convergence rate of spatial error for steady-state problems that are sufficiently smooth. A discussion on robustness of the discontinuous Galerkin method is presented with some possible strategies for improving the numerical stability of the method. Lastly, a performance study demonstrates the high computational efficiency and high parallel scalability of the method.

2.0.1 Notation

Within this work, we represent the a general set of expansion functions by ψ_s, modal expansion functions by ϕ_s, and nodal expansion functions by ℓ_s with s indexing the individual functions. Variables with a subscript index (ϕ_s), bold text ($\boldsymbol{\phi}$), or arrow ($\vec{\phi}$) indicate a vector. The variable $\boldsymbol{x}$ indicates the physical coordinate system and $\boldsymbol{\xi}$ indicates the reference coordinate system. Lastly, the summation notation $\sum_{m,n,l=1}^{N}(\cdot)$ indicates $\sum_{l=1}^{N}\sum_{n=1}^{N}\sum_{m=1}^{N}(\cdot)$.

2.1 Governing Equations

2.1.1 Compressible Navier-Stokes Equations

The governing equations exclusively utilized in this work are the three-dimensional compressible Navier-Stokes equations which can be written in conservative form:

$$\frac{\partial \mathbf{Q}\left(\boldsymbol{x},t\right)}{\partial t}+\vec{\nabla}\cdot\mathbf{F}\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)=0 \tag{2.1}$$

representing the conservation of mass, momentum, and energy for a fluid. The solution vector $\mathbf{Q}$ represents the conservative flow variables and the matrix $\mathbf{F}$ represents the Cartesian flux components. $\mathbf{Q}$ and $\mathbf{F}$ are defined as follows:

$$\mathbf{Q}=\begin{Bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho E\end{Bmatrix},\mathbf{F}=\begin{Bmatrix}\underline{\mathbf{F}^1} & \underline{\mathbf{F}^2} & \underline{\mathbf{F}^3}\\ \rho u & \rho v & \rho w\\ \rho u^2+p-\tau_{11} & \rho uv-\tau_{12} & \rho uw-\tau_{13}\\ \rho uv-\tau_{21} & \rho v^2+p-\tau_{22} & \rho vw-\tau_{23}\\ \rho uw-\tau_{31} & \rho vw-\tau_{32} & \rho w^2+p-\tau_{33}\\ \rho uH+q_1-\tau_{1j}u_j & \rho vH+q_2-\tau_{2j}u_j & \rho wH+q_3-\tau_{3j}u_j\end{Bmatrix} \tag{2.2}$$

where ρ is the density, u,v,w are the velocity components in each spatial coordinate direction, p is the pressure, E is total internal energy, $H=E+\frac{p}{\rho}$ is the total enthalpy, τ is the viscous stress tensor, and q is the heat flux.

The viscous stress tensor, τ, can be approximated via the Boussinesq approach for the Reynolds stresses of a Newtonian fluid:

$$\tau_{ij} = 2\mu S_{ij} \tag{2.3}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{2.4}$$

where μ is the dynamic viscosity and δ is the Kronecker delta operator.
The heat fluxes, q_i, can be written approximated as follows:

$$q_i = -\kappa_T \frac{\partial T}{\partial x_i} \approx -C_p \frac{\mu}{Pr}\frac{\partial T}{\partial x_i} \tag{2.5}$$

where κ_T is the thermal conductivity, T is the fluid temperature, C_p is the heat capacity at constant pressure, and Pr is the Prandtl number. The dynamic viscosity, μ, is a function of the temperature given by the Sutherland's law:

$$\mu = \mu_0 \frac{T_0 + C}{T + C}\left(\frac{T}{T_0}\right)^{\frac{3}{2}} \tag{2.6}$$

where μ_0 is the reference dynamic viscosity, T_0 is the reference temperature, and C is Sutherland's temperature constant for air, the gaseous material under study in this work. To close these set of equations, the assumption of ideal gas is used. The equation of state for an ideal gas is written as follows:

$$\rho E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) \tag{2.7}$$

where $\gamma = 1.4$ is the ratio of specific heats for air.

2.1.2 Large Eddy Simulation and Subgrid-Scale Models

Numerical simulation using a Direct Numerical Simulation (DNS) approach for turbulent flows requires that all scales of turbulence, spatial and temporal, be resolved. This constraint is intractable computationally for all problems except a small subset of problems of low Reynolds number. An alternative to resolving all scales is to resolve only the largest turbulent scales. Large Eddy Simulation (LES) is a mathematical model for turbulence in which the principal idea is to decompose the turbulent length scales stemming from Kolmogorov's theory of self similarity [88]: large turbulent eddies of the fluid flow depend on the geometry where as the small turbulent scales are more universal. This mathematical model introduces the concept of explicitly solving for the large turbulence scales while imposing a mathematical model to implicitly account for the small turbulence scales by using a subgrid-scale (SGS) model [89]. For general flows, LES filtering can be applied spatially and temporally on a field variable $u(\boldsymbol{x}, t)$ to form the filtered variable, denoted by an overbar:

$$\overline{u(\boldsymbol{x}, t)} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(\boldsymbol{x}, t)\, G(\boldsymbol{x} - \mathbf{r}, t - t')\, dt' d\mathbf{r} \tag{2.8}$$

where G is the filter convolution operator which is associated with a length and time cutoff scale. Such scales below these cutoff values are eliminated from the averaged quantity $\overline{u}$.

For compressible flows, a Favre averaging approach [90] is used to obtain an averaged set of equations by applying a low-pass filter to the Navier-Stokes equations such that the smallest scales of the turbulent flow are filtered out. The low-pass filter can be an explicit grid convolution operator defined as a density weighted averaging, denoted by a tilde:

$$\tilde{\Phi} = \frac{\overline{\rho \Phi}}{\overline{\rho}} \tag{2.9}$$

This averaging operator is applied to all flow variables except density and pressure. The averaged flow variables are inserted into the Navier-Stokes equations which introduce additional terms in the momentum and energy equations.

The closure of these additional terms requires the introduction of a turbulence model. One approach to modeling unresolved scales to close the additional terms is the SGS model which assumes any scales smaller than the cutoff filter as determined by the local grid resolution need to be modeled. One particular choice is to use a functional (eddy-viscosity) model which highlights the dissipation of energy analogous to molecular diffusion; this introduces a turbulent eddy viscosity. In most LES formulations, the spacial grid filter is not explicitly performed, but is assumed implicitly through the grid resolution of the numerical simulation. In this work, no explicit grid filter is applied and an additional turbulent viscosity is applied as follows:

$$\begin{aligned} \mu_T &= \mu + \mu_{sgs} && (2.10) \\ \mu_{sgs} &= \rho \left(C_s \Delta\right)^2 \left|\tilde{S}\right| && (2.11) \\ \left|\tilde{S}\right| &= \sqrt{\tilde{S}_{ij}\tilde{S}_{ij}} && (2.12) \\ \tilde{S}_{ij} &= \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) && (2.13) \end{aligned}$$

where C_s is the Smagorinsky model value and Δ is the length scale associated with the grid filter. The value of C_s can be chosen as either constant or dynamic, leading to the constant [89] or dynamic [91] Smagorinsky LES model, respectively.

In this work, C_s is chosen as constant with a value ranging from 0.12 to 0.19. The value Δ is computed as $\Delta = \frac{1}{2}\frac{1}{N}\left(\Delta x \Delta y \Delta z\right)^{\frac{1}{3}}$ where $N = p + 1$ is the polynomial order of the discretization, and Δx, Δy, Δz are the grid widths. In implementation of the LES numerical model, $\frac{\mu}{Pr}$ from Eqn. (2.5) is replaced by $\frac{\mu}{Pr} + \frac{\mu_{sgs}}{Pr_T}$ where Pr_T is the turbulent Prandtl number which is set to a value of 0.90.

2.1.3 Rotating Reference Frame

The governing flow equations, which are usually solved in the inertial reference frame, can alternatively be written in a non-inertial reference frame. To account for a non-inertial reference frame, the fluxes in Eqn. (2.2) are written as follows:

$$\mathbf{F} = \left\{ \begin{array}{lll} \rho(u-u_g) & \rho(v-v_g) & \rho(w-w_g) \\ \rho u(u-u_g)+p-\tau_{11} & \rho u(v-v_g)-\tau_{12} & \rho u(w-w_g)-\tau_{13} \\ \rho v(u-u_g)-\tau_{21} & \rho v(v-v_g)+p-\tau_{22} & \rho v(w-w_g)-\tau_{23} \\ \rho w(u-u_g)-\tau_{31} & \rho w(v-v_g)-\tau_{32} & \rho w(w-w_g)+p-\tau_{33} \\ \rho E(u-u_g)+pu+q_1-\tau_{1j}u_j & \rho E(v-v_g)+pv+q_2-\tau_{2j}u_j & \rho E(w-w_g)+pw+q_3-\tau_{3j}u_j \end{array} \right\} \tag{2.14}$$

where $\mathbf{U}_g = (u_g, v_g, w_g) = \vec{\Omega}$ x $\vec{r}$ is the rotational velocity vector representing the grid velocities with $\vec{\Omega}$ being the angular velocity vector and $\vec{r}$ being the relative position vector. In addition to these flux vector modifications, a source term vector, $\mathbf{S}$, is added to the right-hand-side of the Navier-Stokes equations in Eqn. (2.2):

$$\frac{\partial \mathbf{Q}\left(\boldsymbol{x},t\right)}{\partial t} + \vec{\nabla}\cdot\mathbf{F}\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right) = \mathbf{S}, \qquad \mathbf{S} = \left\{ \begin{array}{c} 0 \\ \Omega_y\rho w - \Omega_z\rho v \\ \Omega_z\rho u - \Omega_x\rho w \\ \Omega_x\rho v - \Omega_y\rho u \\ 0 \end{array} \right\} \tag{2.15}$$

where $\Omega_x, \Omega_y, \Omega_z$ are the angular velocity components prescribed by the rotation rate in the problem. The non-inertial reference frame is used only when an appropriate problem is chosen, such as an isolated rotor simulation (no tower and ground surface), otherwise, the inertial reference frame is used.

2.1.4 Transformation to Generalized Curvilinear Coordinates

The Navier-Stokes equations in Eqn. (2.2) can be expressed in generalized curvilinear coordinates via the chain rule of partial derivatives. The reference coordinate system used herein is written as $\boldsymbol{\xi} = (\xi^1, \xi^2, \xi^3)^T$. The transformed Navier-Stokes equations are written as follows:

$$\frac{\partial \hat{\mathbf{Q}}\left(\boldsymbol{x},t\right)}{\partial t}+\frac{\partial \vec{\mathcal{F}}^{1}\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)}{\partial \xi^{1}}+\frac{\partial \vec{\mathcal{F}}^{2}\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)}{\partial \xi^{2}}+\frac{\partial \vec{\mathcal{F}}^{3}\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)}{\partial \xi^{3}}=0 \tag{2.16}$$

where,

$$\begin{aligned}
\hat{\mathbf{Q}} &= J\mathbf{Q}, \\
\vec{\mathcal{F}}^{1} &= \frac{1}{J}\left[\frac{\partial(\xi^{1})}{\partial x}\mathbf{F}^{1}\right], \\
\vec{\mathcal{F}}^{2} &= \frac{1}{J}\left[\frac{\partial(\xi^{2})}{\partial y}\mathbf{F}^{2}\right], \\
\vec{\mathcal{F}}^{3} &= \frac{1}{J}\left[\frac{\partial(\xi^{3})}{\partial z}\mathbf{F}^{3}\right], \\
J &= \det\left(\frac{\partial(\xi^{1},\xi^{2},\xi^{3})}{\partial(x,y,z)}\right)
\end{aligned} \tag{2.17}$$

and $\mathbf{F}^{1}$, $\mathbf{F}^{2}$, and $\mathbf{F}^{3}$ defined in Eqn. (2.2).

2.1.5 Transformation to Cartesian Coordinates

In this work, a scaled Cartesian reference frame is used exclusively. To transform physical Cartesian coordinates, $\boldsymbol{x}=(x,y,z)^{T}$, to reference coordinates, $\boldsymbol{\xi}=(\xi^{1},\xi^{2},\xi^{3})^{T}$, the transformation is a pure dilation. Thus the integral of a general function G in physical space transformed to the reference space is given as:

$$\begin{aligned}
\int_{\Omega_{k}} G(\boldsymbol{x})\mathrm{d}\boldsymbol{x} &= \int_{z}\int_{y}\int_{x} G\left(x,y,z\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \\
&= \int_{\xi^{3}}\int_{\xi^{2}}\int_{\xi^{1}} G\left(\xi^{1},\xi^{2},\xi^{3}\right)\frac{dx}{d\xi^{1}}\frac{dy}{d\xi^{2}}\frac{dz}{d\xi^{3}}\mathrm{d}\xi^{1}\mathrm{d}\xi^{2}\mathrm{d}\xi^{3} \\
&= \int_{E} G(\boldsymbol{\xi})J(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}
\end{aligned} \tag{2.18}$$

where,

$$J(\boldsymbol{\xi})=\frac{dx}{d\xi^{1}}\frac{dy}{d\xi^{2}}\frac{dz}{d\xi^{3}} \tag{2.19}$$

When the computational mesh is fixed, J is a constant.

2.2 Expansion Basis Functions

In finite-element methods, the expansion basis functions are used to represent the finite solution of the discretized equations. These expansion functions are usually chosen to be polynomials of *modal* or *nodal* type. The solution expansion at a point x and time t using these functions is defined as follows:

$$u(x,t) = \sum_{s=1}^{N} \hat{u}_s(t)\phi_s(x) \tag{2.20}$$

$$u(x,t) = \sum_{s=1}^{N} \tilde{u}_s(t)\ell_s(x) \tag{2.21}$$

The modal basis functions of order N are denoted ϕ_s and the nodal basis functions of order N are denoted ℓ_s, with $\hat{u}_s$ and $\tilde{u}_s$ being the respective expansion coefficients. To evince the difference between modal and nodal basis functions, we define two complete sets of one-dimensional polynomials up to order N.

First, define a set of modal basis functions as:

$$\phi_s(x) = x^{s-1}, \quad s = 1, \ldots, N \tag{2.22}$$

Modal basis functions are hierarchical meaning the order $N-1$ expansion functions are a subset of the order N expansion functions. The particular choice of modal hierarchical basis functions is not unique. One may choose, for example, the Legendre polynomials to form the modal expansion basis functions.

Next, define a set of nodal basis functions as:

$$\ell_s(x) = \prod_{i=1, i\neq s}^{N} \frac{(x-\xi_i)}{(\xi_s-\xi_i)}, \quad s = 1, \ldots, N \tag{2.23}$$

This set of nodal basis functions, $\ell_s(x)$ is a Lagrange interpolating polynomial defined on a set of N nodal points ξ_i. In contrast to the modal expansion basis functions, the Lagrange polynomial is not hierarchical since all polynomials are of degree p.

The Lagrange polynomial has the important property:

$$\ell_s(\xi_i) = \delta_{si} = \begin{cases} 0, & s \neq i \\ 1, & s = i \end{cases} \tag{2.24}$$

This property gives the following consequence:

$$u\left(\xi_i\right) = \sum_{s=1}^{N} u_s \ell_s\left(\xi_i\right) = \sum_{s=1}^{N} u_s \delta_{si} = u_i \tag{2.25}$$

This work selectively chooses the set of N nodal points ξ_i in Eqns. (2.23) and (2.24) as the solution interpolation points. Additionally, the solution interpolation points are chosen to the same points used for numerical integration via quadrature: Gauss-Legendre [92] or Lobatto-Gauss-Legendre [92] points. This is known as a collocation method. The expansion coefficients $\tilde{u}$ are then equal to the solution u at those points. Collocation leads to significant computational savings as there is no interpolation needed from solution points to integration quadrature points. As we will see later in this section, this can be an $\mathcal{O}(N^6)$ operation savings when comparing to a general finite-element method in three dimensional coordinates.

The expansion basis functions have been established for one spatial dimension. To expand to three dimensions for this work, the nodal functions are chosen to be tensor-products of the one-dimensional expansion functions. The solution expansion in three dimensions is written as follows:

$$u(\boldsymbol{x}, t) = \sum_{l=1}^{N} \sum_{n=1}^{N} \sum_{m=1}^{N} u_{mnl}(t) \ell_m(x) \ell_n(y) \ell_l(z) \tag{2.26}$$

Notice in Eqn. (2.26) the indices of the nodal basis functions are independent and, thus, allow for simple one-dimensional operators to be executed in a dimension-by-dimension procedure. General tensor-product operations allows for what is known as sum-factorization which was first recognized by Orszag [93]. Sum-factorization is key to computational efficiency of the tensor-product computations.

To demonstrate this technique, the basis functions can be factored as follows:

$$\begin{aligned} u(\boldsymbol{x},t) &= \sum_{l=1}^{N}\sum_{n=1}^{N}\sum_{m=1}^{N} u_{mnl}(t)\ell_m(x)\ell_n(y)\ell_l(z) \\ &= \sum_{l=1}^{N}\ell_l(z)\left[\sum_{n=1}^{N}\ell_n(y)\left[\sum_{m=1}^{N} u_{mnl}(t)\ell_m(x)\right]\right] \end{aligned} \tag{2.27}$$

To implement this sum-factorization technique, the inner most summation of Eqn. (2.27) is performed first and stored into a temporary variable, namely $U^1_{nl}(x,t)$:

$$U^1_{nl}(x,t) = \sum_{m=1}^{N} u_{mnl}(t)\ell_m(x) \tag{2.28}$$

The index m has been eliminated through the summation operation leaving the indices n and l. Next, the middle summation in Eqn. (2.27) is evaluated with the temporary variable $U^1_{nl}(x,t)$ substituted in for the inner most summation (which was calculated in Eqn. (2.28)) and stored into a second temporary variable, namely $U^2_l(x,y,t)$:

$$U^2_l(x,y,t) = \sum_{n=1}^{N}\ell_n(y)U^1_{nl}(x,t) \tag{2.29}$$

Again, notice the index n has been eliminated through the summation operation leaving only the index l. Finally, the last summation in Eqn. (2.27) is evaluated with the temporary variable $U^2_l(x,y,t)$ substituted in to arrive at the final result:

$$u(\boldsymbol{x},t) = \sum_{l=1}^{N}\ell_l(z)U^2_l(x,y,t) \tag{2.30}$$

The complete sum-factorization process can be seen as:

$$u(\boldsymbol{x},t) = \sum_{l=1}^{N} \ell_l(z) \underbrace{\left[\sum_{n=1}^{N} \ell_n(y) \underbrace{\left[\sum_{m=1}^{N} u_{mnl}(t)\ell_m(x) \right]}_{U^1_{nl}(x,t)} \right]}_{U^2_l(x,y,t)} \tag{2.31}$$

In contrast, the general finite-element method with a full modal expansion basis which is not constructed by tensor-products is written as follows:

$$u(\boldsymbol{x},t) = \sum_{s=1}^{N^3} u_s(t)\phi_s(\boldsymbol{x}) \tag{2.32}$$

The modal expansion basis ϕ_s cannot be decomposed into three independent variables because the basis is not constructed hierarchically.

For context within the DG method, the solution expansion at a point is required for two operations in the discretization: volume flux evaluation in the interior of a mesh element and surface flux evaluation at the boundary of the mesh element. In three-dimensional problems, the solution expansion for the volume flux of a hexahedron mesh element requires N^3 points. Pseudo-codes illustrating the solution expansion using the tensor-product formulation and the general formulation are shown in Listings (2.1) and (2.2). As seen from the tensor-product solution expansion pseudo-code, three summations are required each of size N^4, given the number of one-dimensional solution modes and quadrature points is N. This results in a total asymptotic cost of $\mathcal{O}(N^4)$, whereas the general modal basis solution expansion requires $\mathcal{O}(N^6)$ work. Further, the solution expansion for the surface flux calculations of a hexahedron mesh element requires $6N^2$ points. Thus, the total computational cost of the tensor-product formulation is $\mathcal{O}(6N^3)$ compared to the general modal expansion cost of $\mathcal{O}(6N^5)$.

In addition to solution expansion, the sum-factorization technique can also be applied to integration of a function multiplied with a test function. In this work, the test functions are chosen to be the same as the solution expansion basis functions. In the DG method, two instances of numerical integration occur: volume flux integration in the interior of a mesh

Listing 2.1: 3D Tensor-Product Expansion

```
!3D Tensor-Product Solution Expansion
! nqp1D = # of 1D quadrature points (N)
! tm1D  = # of 1D solution modes (N)
! [in]  u:      solution modes
! [in]  phi1D: 1D basis functions
! [out] U_out: solution at quadrature points

U_1 = 0.0; U_2 = 0.0; U_out = 0.0;
!first sum
do k = 1,nqp1D
 do l = 1,tm1D
  do n = 1,tm1D
   do m = 1,tm1D
    U_1(m,n,k) += phi1D(l,k) * u(m,n,l)

!second sum
do k = 1,nqp1D
 do j = 1,nqp1D
  do n = 1,tm1D
   do m = 1,tm1D
    U_2(m,j,k) += phi1D(n,j) * U_1(m,n,k)

!third sum
do k = 1,nqp1D
 do j = 1,nqp1D
  do i = 1,nqp1D
   do m = 1,tm1D
    U_out(i,j,k) += phi1D(m,i) * U_2(m,j,k)
```

Listing 2.2: 3D General Expansion

```
!3D General Solution Expansion
! nqp1D = # of 1D quadrature points (N)
! tm1D  = # of 1D solution modes (N)
! [in]  u:      solution modes
! [in]  phi3D: 3D basis functions
! [out] U_out: solution at quadrature points

U_out = 0.0;

do k = 1,nqp1D
 do j = 1,nqp1D
  do i = 1,nqp1D
   do l = 1,tm1D
    do n = 1,tm1D
     do m = 1,tm1D
      U_out(i,j,k) += phi3D(m,n,l,i,j,k) * u(m,n,l)
!
!
!
!
!
!
!
!
!
!
!
!
```

element and surface flux integration at the boundary of the mesh element. These operations have the same asymptotic cost as the solution expansion operations stated previously.

As seen from asymptotic analysis above for three dimensions, the cost of solution expansion and integration for volume operations is $\mathcal{O}(N^3)$ per point for tensor-product expansion basis functions, and $\mathcal{O}(N^4)$ for all N^3 points required for numerical integration. General modal expansion basis functions require $\mathcal{O}(N^3)$ operations per point, but require $\mathcal{O}(N^6)$ operations for all N^3 points.

Further computational efficiency is gained by utilizing collocation of solution points and numerical quadrature points. By this design choice, the volume solution expansion is not required as the solution values are the expansion coefficients. Thus for N^3 points, an $\mathcal{O}(N^6)$ operation is saved in comparison to the general finite-element method. From these advantageous design choices, the number of operations is comparable to finite-difference methods. Additionally, with higher arithmetic intensity per degree-of-freedom, the collocated DG method with tensor-product basis functions has superior computational efficiency compared to the finite-difference method, which is demonstrated in the Computational and Parallel Performance Results section. The next section details the DG method with collocation and tensor-production expansion and test basis functions for the three-dimensional compressible Navier-Stokes equations.

2.3 Discontinuous Galerkin Spatial Discretization

The computational domain is partitioned into a block-structured Cartesian collection of non-overlapping hexahedra $\mathcal{T}_h$, of uniform element size h, such that $\Omega = \bigcup_{k \in \mathcal{T}_H} \Omega_k$, where Ω_k refers to the volume of an element k with $k \in \mathcal{T}_H$. Within each element k, a finite-dimensional function space is defined using a finite set of functions with up to degree p. For the discontinuous Galerkin discretization, first an inner product of the governing equations with each function in the finite set is performed in each element. These weighting functions are called test functions and they are chosen to be the same solution expansion functions polynomials in this work. The solution $\mathbf{Q}$ is discretized into polynomial representation $\mathbf{Q}_h$ with unknown polynomial coefficients, also referred as modes. These basis functions are defined on a standard reference element E which is the set of points spanned by $\boldsymbol{\xi}$ where $\xi^{1,2,3} \in [-1, 1]$, i.e. $E = [-1, 1]^3$. The final step of the discretization is to perform integration over the element. To obtain the weak formulation, integration-by-parts is performed once over each element. To obtain the strong formulation* of the discretization, integration-by-parts is performed twice over each element. For the latter, the second integration-by-parts is performed on the weak formulation volume-integration term. For both formulations, the integration is performed using numerical quadrature (cubature in multiple dimensions) in the reference element. The integrals are then transformed back to physical space via the inverse geometric Jacobian mapping (which is the Cartesian mesh scaling).

2.3.1 Weak Formulation

Traditional discontinuous Galerkin formulations perform discretization using the weak form of the conservation laws in Eqns. (2.2). To obtain the weak form of the discretized equations, the governing equations are multiplied by a set of test functions, $\{\psi_s, s = 1, ..., M\}$, and integrated over the element volume k:

$$\int_{\Omega_k} \left(\frac{\partial \mathbf{Q}}{\partial t} + \vec{\nabla} \cdot \mathbf{F} \right) \boldsymbol{\psi}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 0 \tag{2.33}$$

*Nomenclature from *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications* [94]

Integrating Eqn. (2.33) by parts the weak form total residual $\mathbf{R}^{\text{Weak}}$ is defined as:

$$\mathbf{R}^{\text{Weak}} = \int_{\Omega_k} \frac{\partial \mathbf{Q}}{\partial t} \boldsymbol{\psi}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \; - \int_{\Omega_k} \left(\mathbf{F} \cdot \vec{\nabla} \right) \boldsymbol{\psi}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int_{\Gamma_k} \left(\mathbf{F}^* \cdot \vec{\mathrm{n}} \right) \boldsymbol{\psi}(\boldsymbol{x}|_{\Gamma_k}) \mathrm{d}\Gamma_k = 0 \tag{2.34}$$

The residual now contains integrals over faces Γ where special treatment is needed for the fluxes in these terms. The advective fluxes $\mathbf{F}^*$ are calculated using an approximate Riemann solver. The scheme implemented in this work is the Lax-Friedrichs method [95], while the diffusive fluxes are handled using a symmetric interior penalty (SIP) method [96, 97]. This work is focused on subsonic flows without the existence of shocks, thus the exact solution exists in C^∞. Therefore, we do not expect the order of convergence to be limited by solution irregularity.

2.3.2 Strong Formulation

To obtain the strong form[†] of the discretized equations, we integrate-by-parts Eqn. (2.34) a second time. The strong form total residual $\mathbf{R}^{\text{Strong}}$ is defined as:

$$\mathbf{R}^{\text{Strong}} = \int_{\Omega_k} \frac{\partial \mathbf{Q}}{\partial t} \boldsymbol{\psi}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \; + \int_{\Omega_k} \left(\vec{\nabla} \mathbf{F} \cdot \boldsymbol{\psi}(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} + \int_{\Gamma_k} \left((\mathbf{F}^* - \mathbf{F}) \cdot \vec{\mathrm{n}} \right) \boldsymbol{\psi}(\boldsymbol{x}|_{\Gamma_k}) \mathrm{d}\Gamma_k = 0 \tag{2.35}$$

Notice that from the second integration-by-parts, the derivative of the test function in the weak formulation has transferred back to the flux term and an additional term in the face integral has appeared. From the second integration-by-parts, the original flux evaluations interior to the cell boundary, $\mathbf{F}|_{\Gamma_k}$, are subtracted from $\mathbf{F}^*$ as the approximate Riemann solver provides upwinding information from neighboring element flow solutions.

[†]Nomenclature from *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications* [94]

2.3.3 Solution, Flux, and Integration Approximation

To discretize the solution to a finite set of points, we define a set of points to serve as solution points. In a collocation method, we choose the set of solution points to be the same as the numerical quadrature points used for evaluation and integration while imposing that the residual vanish at these points. We utilize either the Gauss-Legendre (GL) points or the Lobatto-Gauss-Legendre (LGL) points as highlighted in the Expansion Basis Functions section. By this choice, we use the GL or LGL as the nodes, ξ_i, in the Lagrange interpolating polynomial in Eqn. (2.23). The solution of each reference mesh element is approximated as:

$$\begin{aligned}
\mathbf{Q}(\boldsymbol{\xi},t) &= \sum_{s=1}^{N^3} \tilde{\mathbf{Q}}_s(t)\psi_s(\boldsymbol{\xi}) \\
&= \sum_{l=1}^{N} \ell_l(\xi^3) \left[\sum_{n=1}^{N} \ell_n(\xi^2) \left[\sum_{m=1}^{N} \mathbf{Q}_{mnl}(t)\ell_m(\xi^1) \right] \right] \\
&= \sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl}(t)\ell_m(\xi^1)\ell_n(\xi^2)\ell_l(\xi^3)
\end{aligned} \tag{2.36}$$

with degrees-of-freedom $\tilde{\mathbf{Q}}_s(t) = \mathbf{Q}_{mnl}(t)$. We denote ψ_s as a general expansion basis function. In this work, we specify the expansion function to be the nodal Lagrange interpolation function.

For transformed flux values as required in Eqns. (2.34) and (2.35), the fluxes are evaluated at the quadrature points:

$$\begin{aligned}
\mathcal{F}(\mathbf{Q}(\boldsymbol{\xi})) &= \sum_{s=1}^{N^3} \tilde{\mathcal{F}}_s\psi_s(\boldsymbol{\xi}) \\
&= \sum_{l=1}^{N} \ell_l(\xi^3) \left[\sum_{n=1}^{N} \ell_n(\xi^2) \left[\sum_{m=1}^{N} \tilde{\mathcal{F}}_{mnl}\ell_m(\xi^1) \right] \right] \\
&= \sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}_{mnl}\ell_m(\xi^1)\ell_n(\xi^2)\ell_l(\xi^3)
\end{aligned} \tag{2.37}$$

where $\tilde{\mathcal{F}}_{mnl}$ is the flux from Eqn. (2.2) evaluated using $\mathbf{Q}_{mnl}$ resulting in $\tilde{\mathcal{F}}_{mnl} = \mathcal{F}(\mathbf{Q}_{mnl})$.

As alluded to by solution approximation, we approximate continuous integration by numerical integration techniques via Gaussian quadrature rules. The general formula for Gaussian quadrature is written as follows:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} \omega_i f(\xi_i) \tag{2.38}$$

where ω_i are quadrature weights and ξ_i are quadrature nodes. The Gauss-Legendre points and weights yield more accuracy than the Lobatto-Gauss-Legendre points and weights [98]; GL quadrature rules can integrate all polynomials up to degree $2N-1$ exactly in comparison to LGL quadrature which can integrate polynomials up to $2N-3$ exactly [99].

2.3.4 Temporal Derivative Integral

The integrands of the integration over the standard element, E, in Eqns. (2.34) and (2.35) are assumed to be finite and regular. Thus, the time derivative is factored out of the integral as follows:

$$\int_E \frac{\partial \mathbf{Q}}{\partial t} \boldsymbol{\psi} J(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} = \frac{\partial}{\partial t} \int_E \mathbf{Q} \boldsymbol{\psi} J(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \tag{2.39}$$

where $J(\boldsymbol{\xi})$ is geometric Jacobian mapping in Eqn. (2.19). Next, we discretize the continuous solution $\mathbf{Q}$ by performing the solution expansion in Eqn. (2.36) and setting the test function to be the same as the basis functions:

$$\frac{\partial}{\partial t} \int_E \mathbf{Q} \boldsymbol{\psi} J(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} = \frac{\partial}{\partial t} \int_E \underbrace{\left(\sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl}(t) \ell_m(\xi^1) \ell_n(\xi^2) \ell_l(\xi^3) \right)}_{\mathbf{Q}} \underbrace{\ell_i(\xi^1) \ell_j(\xi^2) \ell_k(\xi^3)}_{\boldsymbol{\psi}} J(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \tag{2.40}$$

Next, we introduce numerical integration via one-dimensional quadrature in each spatial direction highlighted by Eqn. (2.38) to replace the continuous integration over the volume of the element.

Thus, when continuous integration is replaced by numerical integration, Eqn. (2.40) becomes:

$$\begin{aligned} & \frac{\partial}{\partial t} \int_E \left(\sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl}(t) \ell_m(\xi^1) \ell_n(\xi^2) \ell_l(\xi^3) \right) \ell_i(\xi^1) \ell_j(\xi^2) \ell_k(\xi^3) J(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \\ \approx & \frac{\partial}{\partial t} \sum_{\lambda,\mu,\nu=1}^{N} \left(\sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl}(t) \ell_m(\xi_\lambda^1) \ell_n(\xi_\mu^2) \ell_l(\xi_\nu^3) \right) \ell_i(\xi_\lambda^1) \ell_j(\xi_\mu^2) \ell_k(\xi_\nu^3) J(\xi_\lambda^1, \xi_\mu^2, \xi_\nu^3) \omega_\lambda \omega_\mu \omega_\nu \end{aligned} \tag{2.41}$$

where $\xi_\lambda^1, \xi_\mu^2, \xi_\nu^3$ represent the quadrature points, $\omega_\lambda, \omega_\mu, \omega_\nu$ represent the quadrature weights shown in Eqn. (2.38).

Recall in this work, collocation of the solution points and quadrature points is utilized, therefore, the property of Lagrange interpolating polynomials in Eqn. (2.24) holds. Thus, we can replace all basis function evaluations by Kronecker delta products, as the quadrature points, $(\xi_\lambda, \xi_\mu, \xi_\nu)$, are the same as the basis collocation points. Note that the Kronecker delta property states that the indices of the product must be the same, else the value is zero. Recall Eqn. (2.24) for the following:

$$\ell_i(\xi_\lambda)\omega_\lambda = \delta_{i\lambda}\omega_\lambda = \omega_i \tag{2.42}$$

The index i must be the same as the index λ in the Kronecker delta product, else the product is zero. Therefore, every i index must be changed to a $j\lambda$ index. Following this example, we see all indices must match. Therefore, we change indices $\lambda \to i$, $\mu \to j$, $\nu \to k$, and $m \to \lambda$, $n \to \mu$, $l \to \nu$. From these chained relations, then $m \to i$, $n \to j$, $l \to k$.

Eqn. (2.41) becomes:

$$
\begin{aligned}
& \frac{\partial}{\partial t} \sum_{\lambda,\mu,\nu=1}^{N} \left(\sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl} \underbrace{\ell_m(\xi_\lambda^1)}_{=\delta_{m\lambda}} \underbrace{\ell_n(\xi_\mu^2)}_{=\delta_{n\mu}} \underbrace{\ell_l(\xi_\nu^3)}_{=\delta_{l\nu}} \right) \underbrace{\ell_i(\xi_\lambda^1)}_{=\delta_{i\lambda}} \underbrace{\ell_j(\xi_\mu^2)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi_\nu^3)}_{=\delta_{k\nu}} J(\xi_\lambda^1, \xi_\mu^2, \xi_\nu^3) \omega_\lambda \omega_\mu \omega_\nu \\
= \; & \frac{\partial}{\partial t} \sum_{\lambda,\mu,\nu=1}^{N} \mathbf{Q}_{\lambda\mu\nu} \underbrace{\ell_i(\xi_\lambda^1)}_{=\delta_{i\lambda}} \underbrace{\ell_j(\xi_\mu^2)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi_\nu^3)}_{=\delta_{k\nu}} J(\xi_\lambda^1, \xi_\mu^2, \xi_\nu^3) \omega_\lambda \omega_\mu \omega_\nu \qquad \boxed{\text{by: } \underbrace{\delta_{m\lambda}}_{m\to\lambda}, \; \underbrace{\delta_{n\mu}}_{n\to\mu}, \; \underbrace{\delta_{l\nu}}_{l\to\nu}} \\
= \; & J(\xi_i^1, \xi_j^2, \xi_k^3) \omega_i \omega_j \omega_k \frac{\partial \mathbf{Q}_{ijk}}{\partial t} \qquad \boxed{\text{by: } \underbrace{\delta_{\lambda i}}_{\lambda\to i}, \; \underbrace{\delta_{\mu j}}_{\mu\to j}, \; \underbrace{\delta_{\nu k}}_{\nu\to k}}
\end{aligned}
\tag{2.43}
$$

This work uses a fixed Cartesian mesh of uniform element size, thus, the mesh mapping Jacobian J is a constant. Finally, we define the mass matrix, $\mathbb{M}$, and its inverse, $\mathbb{M}^{-1}$, as follows:

$$
\mathbb{M} = M_{ijk} = J\omega_i\omega_j\omega_k \quad \mathbb{M}^{-1} = \left(J\omega_i\omega_j\omega_k\right)^{-1} \tag{2.44}
$$

The time derivative integral of Eqns. (2.34) and (2.35) simplified becomes:

$$
\boxed{\underline{\textbf{Temporal Derivative Integral:}} \qquad \int_{\Omega_k} \frac{\partial \mathbf{Q}}{\partial t} \boldsymbol{\psi}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{M} \frac{\partial \mathbf{Q}_{ijk}}{\partial t}} \tag{2.45}
$$

2.3.5 Volume Integrals

Analysis for both the weak and strong forms are performed in this section. Note that for the weak formulation, the derivative in the volume integral operates on the test function conversely to the strong form, where the derivative operates on the flux function.

Weak Form Volume Integral

The volume integral in Eqn. (2.34) is split into three contravariant flux components:

$$\int_{\Omega_k} \left(\mathbf{F}(\mathbf{Q}) \cdot \vec{\nabla}\right) \boldsymbol{\psi}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \sum_{d=1}^{3} \int_E \mathcal{F}^d(\mathbf{Q}(\boldsymbol{\xi})) \frac{\partial \boldsymbol{\psi}(\boldsymbol{\xi})}{\partial \xi^d} \mathrm{d}\boldsymbol{\xi} \tag{2.46}$$

The contravariant fluxes are approximated by the interpolation at the nodal solution points given by Eqn. (2.37) written as follows:

$$\mathcal{F}^d(\mathbf{Q}(\boldsymbol{\xi})) = \sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^d_{mnl} \ell_m(\xi^1) \ell_n(\xi^2) \ell_l(\xi^3) \tag{2.47}$$

Further, this work uses nodal basis functions for the test functions:

$$\boldsymbol{\psi}(\boldsymbol{\xi}) = \ell_i(\xi^1) \ell_j(\xi^2) \ell_k(\xi^3) \tag{2.48}$$

Examining the first sum in the Eqn. (2.46), we replace the the flux function by its approximation and replace the test function by its Galerkin basis functions.

$$\int_E \mathcal{F}^1(\mathbf{Q}(\boldsymbol{\xi})) \frac{\partial \boldsymbol{\psi}(\boldsymbol{\xi})}{\partial \xi^1} \mathrm{d}\boldsymbol{\xi} = \int_E \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^1_{mnl} \ell_m(\xi^1) \ell_n(\xi^2) \ell_l(\xi^3) \right) \frac{\partial \left[\ell_i(\xi^1) \ell_j(\xi^2) \ell_k(\xi^3)\right]}{\partial \xi^1} \mathrm{d}\boldsymbol{\xi} \tag{2.49}$$

The partial differentiation is with respect to ξ^1, thus the Lagrange basis functions which do not depend on ξ^1 can be factored out of the differential operator. Further, the differentiation

of the test function dependent on ξ^1 can be transformed into a total derivative since it is one-dimensional. Eqn. (2.49) can be written as follows:

$$\int_E \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^d_{mnl} \ell_m(\xi^1)\ell_n(\xi^2)\ell_l(\xi^3) \right) \frac{\partial \left[\ell_i(\xi^1)\ell_j(\xi^2)\ell_k(\xi^3)\right]}{\partial \xi^1} \mathrm{d}\boldsymbol{\xi} =$$
$$\int_E \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^d_{mnl} \ell_m(\xi^1)\ell_n(\xi^2)\ell_l(\xi^3) \right) \frac{\mathrm{d}\ell_i(\xi^1)}{\mathrm{d}\xi^1} \ell_j(\xi^2)\ell_k(\xi^3) \mathrm{d}\boldsymbol{\xi} \tag{2.50}$$

The continuous integration is now approximated by numerical quadrature. Recall the use of collocation of solution and quadrature points gives the Kronecker delta property in Eqn. (2.24). Following these operations, Eqn. (2.50) is written as follows:

$$\begin{aligned}
&\int_E \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^d_{mnl} \ell_m(\xi^1)\ell_n(\xi^2)\ell_l(\xi^3) \right) \frac{\mathrm{d}\ell_i(\xi^1)}{\mathrm{d}\xi^1} \ell_j(\xi^2)\ell_k(\xi^3) \mathrm{d}\boldsymbol{\xi} \\
\approx\ & \sum_{\lambda,\mu,\nu=1}^{N} \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^1_{mnl} \underbrace{\ell_m(\xi^1_\lambda)}_{=\delta_{m\lambda}} \underbrace{\ell_n(\xi^2_\mu)}_{=\delta_{n\mu}} \underbrace{\ell_l(\xi^3_\nu)}_{=\delta_{l\nu}} \right) \frac{\mathrm{d}\ell_i(\xi^1_\lambda)}{\mathrm{d}\xi^1} \underbrace{\ell_j(\xi^2_\mu)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi^3_\nu)}_{=\delta_{k\nu}} \omega_\lambda\omega_\mu\omega_\nu \\
=\ & \sum_{\lambda,\mu,\nu=1}^{N} \tilde{\mathcal{F}}^1_{\lambda\mu\nu} \frac{\mathrm{d}\ell_i(\xi^1_\lambda)}{\mathrm{d}\xi^1} \underbrace{\ell_j(\xi^2_\mu)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi^3_\nu)}_{=\delta_{k\nu}} \omega_\lambda\omega_\mu\omega_\nu \qquad \boxed{\text{by: } \underbrace{\delta_{m\lambda}}_{m\to\lambda}, \underbrace{\delta_{n\mu}}_{n\to\mu}, \underbrace{\delta_{l\nu}}_{l\to\nu}} \\
=\ & \omega_j\omega_k \sum_{\lambda=1}^{N} \tilde{\mathcal{F}}^1_{\lambda jk} \frac{\mathrm{d}\ell_i(\xi^1_\lambda)}{\mathrm{d}\xi^1} \omega_\lambda \qquad \boxed{\text{by: } \underbrace{\delta_{\mu j}}_{\mu\to j}, \underbrace{\delta_{\nu k}}_{\nu\to k}}
\end{aligned} \tag{2.51}$$

As seen from the last equation in Eqn. (2.51), one can define a differentiation matrix $\overline{D}$ defined as follows:

$$\overline{D}_{ij} = \frac{\mathrm{d}\ell_i(\xi_j)}{\mathrm{d}\xi}, \quad i,j = 0, ..., N \tag{2.52}$$

With this definition of the derivative matrix and the second/third summation terms in Eqn. (2.49) written analogously, the entire volume integral for the weak formulation is written as follows:

Weak Formulation Volume Integral:

$$\begin{aligned}\int_{\Omega_k}\left(\mathbf{F}(\mathbf{Q})\cdot\vec{\nabla}\right)\boldsymbol{\psi}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \omega_j\omega_k\sum_{\lambda=1}^{N}\overline{D}_{i\lambda}\tilde{\mathcal{F}}^1_{\lambda jk}\omega_\lambda \\ + \omega_i\omega_k\sum_{\mu=1}^{N}\overline{D}_{j\mu}\tilde{\mathcal{F}}^2_{i\mu k}\omega_\mu \\ + \omega_i\omega_j\sum_{\nu=1}^{N}\overline{D}_{k\nu}\tilde{\mathcal{F}}^3_{ij\nu}\omega_\nu\end{aligned} \tag{2.53}$$

Strong Form Volume Integral

As previously, the strong form volume integral in Eqn. (2.35) is split into three contravariant flux components:

$$\int_{\Omega_k}\left(\vec{\nabla}\mathbf{F}(\mathbf{Q})\cdot\boldsymbol{\psi}(\boldsymbol{x})\right)\mathrm{d}\boldsymbol{x} = \sum_{d=1}^{3}\int_E\frac{\partial\mathcal{F}^d(\mathbf{Q}(\boldsymbol{\xi}))}{\partial\xi^d}\boldsymbol{\psi}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi} \tag{2.54}$$

We use the facts from Eqns. (2.47) and (2.48), and move the differential operator inside the summation operator. Analogously from the weak formulation arithmetic, the differentiation operation is with respect to ξ^d. Thus the terms independent of ξ^d are factored out of the differentiation operator. The first summation term in the strong form volume integral in Eqn. (2.54) is written as follows:

$$\begin{aligned}\int_E\frac{\partial\mathcal{F}^1(\mathbf{Q}(\boldsymbol{\xi}))}{\partial\xi^1}\boldsymbol{\psi}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi} &= \int_E\left(\sum_{m,n,l=1}^{N}\tilde{\mathcal{F}}^1_{mnl}\frac{\partial\left[\ell_m(\xi^1)\ell_n(\xi^2)\ell_l(\xi^3)\right]}{\partial\xi^1}\right)\ell_i(\xi^1)\ell_j(\xi^2)\ell_k(\xi^3)\mathrm{d}\boldsymbol{\xi} \\ &= \int_E\left(\sum_{m,n,l=1}^{N}\tilde{\mathcal{F}}^1_{mnl}\frac{\mathrm{d}\ell_m(\xi^1)}{\mathrm{d}\xi^1}\ell_n(\xi^2)\ell_l(\xi^3)\right)\ell_i(\xi^1)\ell_j(\xi^2)\ell_k(\xi^3)\mathrm{d}\boldsymbol{\xi}\end{aligned} \tag{2.55}$$

The continuous integration is now approximated by numerical quadrature. Recall the use of collocation of solution and quadrature points giving the Kronecker delta property in Eqn. (2.24). Following these operations, Eqn. (2.55) is written as follows:

$$\begin{aligned}
&\int_E \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^1_{mnl} \frac{\mathrm{d}\ell_m(\xi^1)}{\mathrm{d}\xi^1} \ell_n(\xi^2)\ell_l(\xi^3) \right) \ell_i(\xi^1)\ell_j(\xi^2)\ell_k(\xi^3)\mathrm{d}\boldsymbol{\xi} \qquad (2.56)\\
\approx\ & \sum_{\lambda,\mu,\nu=1}^{N} \left(\sum_{m,n,l=1}^{N} \tilde{\mathcal{F}}^1_{mnl} \frac{\mathrm{d}\ell_m(\xi^1_\lambda)}{\mathrm{d}\xi^1} \underbrace{\ell_n(\xi^2_\mu)}_{=\delta_{n\mu}} \underbrace{\ell_l(\xi^3_\nu)}_{=\delta_{l\nu}} \right) \underbrace{\ell_i(\xi^1_\lambda)}_{=\delta_{i\lambda}} \underbrace{\ell_j(\xi^2_\mu)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi^3_\nu)}_{=\delta_{k\nu}} \omega_\lambda\omega_\mu\omega_\nu\\
=\ & \sum_{\lambda,\mu,\nu=1}^{N} \left(\sum_{m=1}^{N} \tilde{\mathcal{F}}^1_{m\mu\nu} \frac{\mathrm{d}\ell_m(\xi^1_\lambda)}{\mathrm{d}\xi^1} \right) \underbrace{\ell_i(\xi^1_\lambda)}_{=\delta_{i\lambda}} \underbrace{\ell_j(\xi^2_\mu)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi^3_\nu)}_{=\delta_{k\nu}} \omega_\lambda\omega_\mu\omega_\nu \qquad \boxed{\text{by: } \underbrace{\delta_{n\mu}}_{n\to\mu},\ \underbrace{\delta_{l\nu}}_{l\to\nu}}\\
=\ & \omega_i\omega_j\omega_k \sum_{m=1}^{N} \tilde{\mathcal{F}}^1_{mjk} \frac{\mathrm{d}\ell_m(\xi^1_i)}{\mathrm{d}\xi^1} \qquad \boxed{\text{by: } \underbrace{\delta_{i\lambda}}_{\lambda\to i},\ \underbrace{\delta_{j\mu}}_{\mu\to j},\ \underbrace{\delta_{k\nu}}_{\nu\to k}}
\end{aligned}$$

As seen from the last equation in Eqn. (2.56), one can define a second differentiation matrix $\overline{\overline{D}}$ defined as follows:

$$\overline{\overline{D}}_{ij} = \frac{\mathrm{d}\ell_j(\xi_i)}{\mathrm{d}\xi}, \quad i,j = 0,...,N \qquad (2.57)$$

Thus the volume integral for the strong formulation can now be written as follows:

Strong Formulation Volume Integral:

$$\begin{aligned}
\int_{\Omega_k} \left(\vec{\nabla}\mathbf{F}(\mathbf{Q}) \cdot \boldsymbol{\psi}(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} = \omega_i\omega_j\omega_k \sum_{m=1}^{N} \overline{\overline{D}}_{im} \tilde{\mathcal{F}}^1_{mjk} \qquad (2.58)\\
+ \omega_i\omega_j\omega_k \sum_{n=1}^{N} \overline{\overline{D}}_{jn} \tilde{\mathcal{F}}^2_{ink}\\
+ \omega_i\omega_j\omega_k \sum_{l=1}^{N} \overline{\overline{D}}_{kl} \tilde{\mathcal{F}}^3_{ijl}
\end{aligned}$$

Relation between Weak and Strong Formulations

Note the relation between the weak form and the strong form differentiation matrices from Eqns. (2.52) and (2.57), respectively:

$$\overline{\overline{D}} = \overline{D}^T \tag{2.59}$$

The summation in the weak formulation volume integral occurs over the quadrature point evaluations in comparison to the strong formulation volume integral which has the sum over the derivative basis functions.

2.3.6 Surface Integral

The surface integral is drastically simplified in the Cartesian framework. This occurs because the outward-facing unit normal for the ξ^1-direction, for example, is $\vec{n} = (\pm 1, 0, 0)^T$ with analogous expression for the other reference coordinate directions. We are able to eliminate the orthogonal components of the numerical flux $\mathbf{F}^*$. The surface integral at $\xi^1 = +1$ is given as:

$$\int_\Gamma (\mathbf{F}^* \cdot \vec{\mathrm{n}})\, \boldsymbol{\psi} \mathrm{d}\Gamma = \int_{-1}^{1}\int_{-1}^{1} \left(\tilde{\mathcal{F}}^*_{(+1)nl}\right) \boldsymbol{\psi}(+1, \xi^2, \xi^3)\mathrm{d}\xi^2\mathrm{d}\xi^3 \tag{2.60}$$

Evaluating the Lagrange polynomials and numerically integrating results in the following:

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} \left(\tilde{\mathcal{F}}^*_{(+1)nl}\right) \boldsymbol{\psi}(+1, \xi^2, \xi^3)\mathrm{d}\xi^2\mathrm{d}\xi^3 \\
\approx\ & \sum_{\mu,\nu=1}^{N} \left(\sum_{n,l=1}^{N} \tilde{\mathcal{F}}^*_{(+1)nl} \underbrace{\ell_n(\xi^2_\mu)}_{=\delta_{n\mu}} \underbrace{\ell_l(\xi^3_\nu)}_{=\delta_{l\nu}} \right) \ell_i(+1) \underbrace{\ell_j(\xi^2_\mu)}_{=\delta_{j\mu}} \underbrace{\ell_k(\xi^3_\nu)}_{=\delta_{k\nu}} \omega_\mu \omega_\nu \\
=\ & \tilde{\mathcal{F}}^*_{(+1)jk} \ell_i(+1) \omega_j \omega_k \qquad \boxed{\text{by: } \underbrace{\delta_{n\mu}}_{n\to\mu},\ \underbrace{\delta_{l\nu}}_{l\to\nu},\ \underbrace{\delta_{j\mu}}_{\mu\to j},\ \underbrace{\delta_{k\nu}}_{\nu\to k}}
\end{aligned} \tag{2.61}$$

The ξ^2- and ξ^3-directional fluxes are analogous to Eqn. (2.61) while paying particular attention to the outward-facing unit normal. In order to evaluate the surface integrals, projection the solution to the faces is required. To project to the surface quadrature points, we perform a product of the solution coefficients and the basis functions evaluated at the quadrature points. For example, the projection of the solution to the $\xi^1 = +1$ surface is the product of the solution coefficients and the basis function evaluated at $\xi^1 = +1$ as follows:

$$\mathcal{Q}(\xi^1 = +1, \xi^2, \xi^3) = \sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl} \ell_m(\xi^1 = +1) \ell_n(\xi^2) \ell_l(\xi^3) \tag{2.62}$$

Further, if the projection points ξ^2 and ξ^3 collocate with the solution points, then the projected solution is a total of N^2, one-dimensional products as follows:

$$\begin{aligned} \mathcal{Q}(\xi^1 = +1, \xi^2_\mu, \xi^3_\nu) &= \sum_{\mu,\nu=1}^{N} \sum_{m,n,l=1}^{N} \mathbf{Q}_{mnl} \ell_m(\xi^1 = +1) \underbrace{\ell_n(\xi^2_\mu)}_{=\delta_{n\mu}} \underbrace{\ell_l(\xi^3_\nu)}_{=\delta_{l\nu}} \\ &= \sum_{\mu,\nu=1}^{N} \sum_{m=1}^{N} \mathbf{Q}_{m\mu\nu} \ell_m(\xi^1 = +1) \qquad \boxed{\text{by: } \underbrace{\delta_{n\mu}}_{n\to\mu}, \underbrace{\delta_{l\nu}}_{l\to\nu}} \end{aligned} \tag{2.63}$$

resulting in an asymptotic cost of $\mathcal{O}(N^3)$ per face. Fig. (2.1) shows the locations of Gauss-Legendre quadrature points in addition to boundary flux point locations. There is an imagined one-dimensional line of solutions points in line for the desired surface quadrature point. The complete surface integral contribution is written as follows:

Surface Integral:

$$\begin{aligned} \int_\Gamma (\mathbf{F}^* \cdot \vec{\mathrm{n}}) \, \boldsymbol{\psi} \mathrm{d}\Gamma = &\left(\tilde{\mathcal{F}}^*_{(+1)jk} \ell_i(+1) - \tilde{\mathcal{F}}^*_{(-1)jk} \ell_i(-1) \right) \omega_j \omega_k \\ &+ \left(\tilde{\mathcal{F}}^*_{i(+1)k} \ell_j(+1) - \tilde{\mathcal{F}}^*_{i(-1)k} \ell_j(-1) \right) \omega_i \omega_k \\ &+ \left(\tilde{\mathcal{F}}^*_{ij(+1)} \ell_k(+1) - \tilde{\mathcal{F}}^*_{ij(-1)} \ell_k(-1) \right) \omega_i \omega_j \end{aligned} \tag{2.64}$$

To compute the strong form of the surface integral, an additional subtraction of the flux given in Eqn. (2.2) is required at the surface quadrature flux points before the surface integration is performed as seed in Eqn. (2.35).

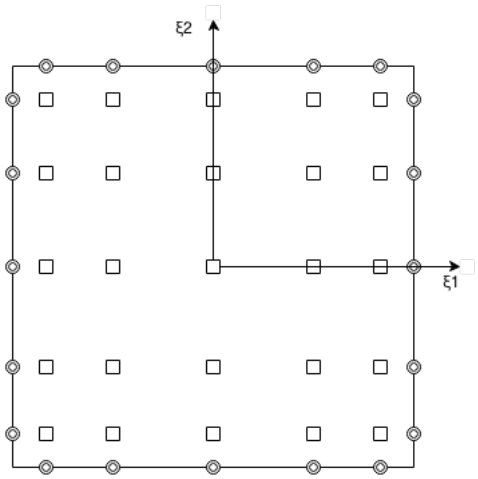


Figure 2.1: Gauss-Legendre solution quadrature points ⊡ for $N = 5$ and Gauss-Legendre boundary quadrature points ⊙.

2.3.7 Semi-Discrete Formulation

From the previous section on numerical discretization, we combine the volume integral and the surface integral into a spatial residual, $\mathbf{R}_{ijk}(\mathbf{Q})$. To evolve the equations temporally, we use an explicit time-stepping scheme. Via the method of lines, the semi-discrete form of equations are written as:

$$\mathbb{M}\frac{\partial \mathbf{Q}_{ijk}(t)}{\partial t} + \mathbf{R}_{ijk}(\mathbf{Q}) = 0 \tag{2.65}$$

where $\mathbb{M}$ is the mass matrix defined Eqn. (2.44). The spatial dimension of the solution $\mathbf{Q}$ is discretized while the temporal dimension remains continuous. Eqn. (2.65) is a system of coupled ordinary differential equations which is solved numerically using explicit Runge-Kutta time-stepping methods [100].

To formulate the Runge-Kutta time-stepping method, we give context through an initial value problem:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0 \tag{2.66}$$

where y is an unknown function dependent on time t with initial value y_0. To approximate the value of y at time t_{n+1} using the present known value $y_n = y(t_n)$ with time step $h = t_{n+1} - t_n$, the family of explicit Runge-Kutta (RK) methods of s stages with coefficients a_{ij}, b_j, and c_i is generalized as follows:

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j k_j \tag{2.67}$$

where,

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + c_2 h, y_n + h(a_{21} k_1)), \\ k_3 &= f(t_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2)), \\ &\vdots \\ k_s &= f(t_n + c_s h, y_n + h(a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{s,s-1} k_{s-1})) \end{aligned} \tag{2.68}$$

The values of a_{ij}, b_j, and c_i can be arranged into a *Butcher tableau* [101]:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} \tag{2.69}$$

0		
1	1	
	1/2	1/2

(a) SSP-TVD RK2

0			
1	1		
1/2	1/4	1/4	
	1/6	1/6	2/3

(b) SSP-TVD RK3

0				
1/3	1/3			
2/3	−1/3	1		
1	1	−1	1	
	1/8	3/8	3/8	1/8

(c) RK 3/8-rule

Figure 2.2: Runge-Kutta method Butcher Tableaus.

The Runge-Kutta method is consistent if $\sum_{j=1}^{i-1} a_{ij} = c_i$ for $i = 2, \ldots, s$. This work makes use of strong stability preserving (SSP) [102] and Total Variation Diminishing (TVD) [103] time discretizations. Explicit Runge-Kutta time-schemes implemented in this work are:

- 2-stage, 2$^{\text{nd}}$-order SSP-TVD RK2 method [104]
- 3-stage, 3$^{\text{rd}}$-order SSP-TVD RK3 method [104]
- 4-stage, 4$^{\text{th}}$-order RK 3/8-rule method

with their respective Butcher tableaus listed in Fig. (2.2).

Using the method of lines formulation in Eqn. (2.65), during each stage of the explicit scheme, the mass matrix must be inverted. For the collocation DG formulation in this work, we demonstrated the mass matrix is block-diagonal and constant allowing for easy inversion. We precompute the inverse mass matrix and multiply through each of the terms in the spatial residual given by Eqn. (2.34) for the weak formulation or by Eqn. (2.35) for the strong formulation. By this process, we can embed the inverted mass matrix into specialized basis functions. To demonstrate this process, we analyze the weak formulation volume integral.

We multiply the volume integral in Eqn. (2.53) by the inverse mass matrix shown in Eqn. (2.44):

$$
\begin{aligned}
& \mathbb{M}^{-1} \cdot \int_{\Omega_k} \left(\mathbf{F}(\mathbf{Q}) \cdot \vec{\nabla} \right) \boldsymbol{\psi}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \qquad (2.70) \\
= \; & \frac{1}{J\omega_i\omega_j\omega_k} \left(\omega_j\omega_k \sum_{\lambda=1}^{N} \overline{D}_{i\lambda} \tilde{\mathcal{F}}^1_{\lambda jk} \omega_\lambda + \omega_i\omega_k \sum_{\mu=1}^{N} \overline{D}_{j\mu} \tilde{\mathcal{F}}^2_{i\mu k} \omega_\mu + \omega_i\omega_j \sum_{\nu=1}^{N} \overline{D}_{k\nu} \tilde{\mathcal{F}}^3_{ij\nu} \omega_\nu \right) \\
= \; & \frac{1}{J} \left(\frac{\cancel{\omega_j\omega_k}}{\omega_i\cancel{\omega_j\omega_k}} \sum_{\lambda=1}^{N} \overline{D}_{i\lambda} \tilde{\mathcal{F}}^1_{\lambda jk} \omega_\lambda + \frac{\cancel{\omega_i\omega_k}}{\cancel{\omega_i}\omega_j\cancel{\omega_k}} \sum_{\mu=1}^{N} \overline{D}_{j\mu} \tilde{\mathcal{F}}^2_{i\mu k} \omega_\mu + \frac{\cancel{\omega_i\omega_j}}{\cancel{\omega_i\omega_j}\omega_k} \sum_{\nu=1}^{N} \overline{D}_{k\nu} \tilde{\mathcal{F}}^3_{ij\nu} \omega_\nu \right) \\
= \; & \frac{1}{J} \left(\sum_{\lambda=1}^{N} \overline{D}_{i\lambda} \tilde{\mathcal{F}}^1_{\lambda jk} \frac{\omega_\lambda}{\omega_i} + \sum_{\mu=1}^{N} \overline{D}_{j\mu} \tilde{\mathcal{F}}^2_{i\mu k} \frac{\omega_\mu}{\omega_j} + \sum_{\nu=1}^{N} \overline{D}_{k\nu} \tilde{\mathcal{F}}^3_{ij\nu} \frac{\omega_\nu}{\omega_k} \right) \\
= \; & \sum_{\lambda=1}^{N} \overline{D}^w_{i\lambda} \tilde{\mathcal{F}}^1_{\lambda jk} + \sum_{\mu=1}^{N} \overline{D}^w_{j\mu} \tilde{\mathcal{F}}^2_{i\mu k} + \sum_{\nu=1}^{N} \overline{D}^w_{k\nu} \tilde{\mathcal{F}}^3_{ij\nu}
\end{aligned}
$$

where the specialized precomputed basis derivative functions $\overline{D}^w_{i\lambda}$, $\overline{D}^w_{j\mu}$, and $\overline{D}^w_{k\nu}$ are written as:

$$
\begin{aligned}
\overline{D}^w_{i\lambda} &= \frac{1}{J} \overline{D}_{i\lambda} \frac{\omega_\lambda}{\omega_i} = \frac{1}{J} \frac{\mathrm{d}\ell_i(\xi_\lambda)}{\mathrm{d}\xi} \frac{\omega_\lambda}{\omega_i} \\
\overline{D}^w_{j\mu} &= \frac{1}{J} \overline{D}_{j\mu} \frac{\omega_\mu}{\omega_j} = \frac{1}{J} \frac{\mathrm{d}\ell_j(\xi_\mu)}{\mathrm{d}\xi} \frac{\omega_\mu}{\omega_j} \\
\overline{D}^w_{k\nu} &= \frac{1}{J} \overline{D}_{k\nu} \frac{\omega_\nu}{\omega_k} = \frac{1}{J} \frac{\mathrm{d}\ell_k(\xi_\nu)}{\mathrm{d}\xi} \frac{\omega_\nu}{\omega_k}
\end{aligned} \qquad (2.71)
$$

Thus, the weak formulation volume integration with the embedded inverse mass matrix reduces to three tensor-product summations (which can be cast into matrix-matrix products) with specialized derivative functions. A special case occurs with the strong formulation volume integral: the inverse mass matrix quadrature weights perfectly cancel the quadrature weights in the volume integration. Thus, no specialized basis functions are needed as the original basis functions suffice. Similar inverse multiplication and analysis can be performed for the surface integrals in Eqn. (2.64).

Explicit time-marching methods have a maximum stable time-step restricted by the Courant-Friedrichs-Lewy (CFL) condition which states for explicit methods, the maximum stable allowable time-step is restricted by the distance traveled by a wave at speed U_{max} must be less than or equal to one mesh element width. For inviscid flows, the maximum wave speed is given by $|u| + c$, where u is the fluid velocity, and c is the speed of sound. Further restrictions on the maximum allowable time-step may be induced by the diffusion rate in the case of viscous flows. To account for this physical rate that is consistent mathematically, the ratio of mesh element area to kinematic viscosity is compared to the ratio of mesh element length to the maximum wave speed. We compute the time-step under the assumption that the CFL number must be less than or equal one. We compute the time-step Δt for the high-order discretization of degree p where ν is the kinematic viscosity, $|U|$ is the maximum advective speed, c is the acoustic wave speed, as follows:

$$\begin{aligned} \Delta t &= \text{CFL} \cdot \min\left(\frac{h}{(|U| + c)}, \frac{h^2}{\nu}\right) \\ h &= \frac{\min\left(dx, dy, dz\right)}{(p+1)^{1.8}} \end{aligned} \tag{2.72}$$

2.4 Discontinuous Galerkin Discretization Stability

Standard implementations of the discontinuous Galerkin method are prone to numerical instability caused by inexact evaluation and integration of the nonlinear flux functions in the governing equations. Since the flux functions are nonlinear, truncation errors of the orthogonal polynomial infinite series representation are introduced which incur aliasing to the higher-order terms [105]. This error is sometimes referred as a *variational crime* [106].

Also note that primitive variables required to compute the nonlinear flux functions of the governing equations are rational functions of the conservative variables thus making the nonlinear flux functions rational functions themselves. Numerical integration through quadrature is only exact for polynomial integrands, thus aliasing errors are inevitable for integration of rational polynomial expressions. Standard implementations utilize a strategy of minimal number of arithmetic operations which has disastrous consequences for under-resolved simulations. In under-resolved turbulence simulations, aliasing of low-frequency-energy containing solution modes into high-frequency-energy containing solution modes can cause instability and failure. To help alleviate this problem, it has been suggested for the compressible Navier-Stokes equations to use $2N$ quadrature points for numerical integration [107, 108].

2.4.1 Over-Integration

Over-integration increases the number of quadrature nodes used for the volume flux integration and surface flux integration shown in Eqns. (2.46) and (2.61), respectively. When using the Gauss-Legendre quadrature points, the minimum number of quadrature points, $Q_{min}^{\mathbf{GL}}$, required to exactly integrate any polynomial of order N, $u(\xi) \in \mathscr{P}_N$, to machine precision is:

$$Q_{min}^{\mathbf{GL}} = \frac{N+1}{2} \tag{2.73}$$

The Lobatto-Gauss-Legendre quadrature points to integrate the same polynomial, $u(\xi)$, exactly to machine precision requires:

$$Q_{min}^{\mathbf{LGL}} = \frac{N+3}{2} \tag{2.74}$$

In Galerkin methods, the evaluation of the L^2 inner-product of two polynomials (ϕ_i, ϕ_j) is required to compute the mass matrix where each polynomial is ansatz to the polynomial space of order N. In the instance of LGL quadrature rules, if the integrand is a product of polynomials, then the number of quadrature points required for exact integration is as follows:

Polynomial Order N	$Q_{min}^{\mathbf{LGL}}$
$[\phi(\xi)]^2 \in \mathscr{P}_{2N}$	$Q \geq N + \frac{3}{2}$

Thus for the polynomial integrands, over-integration may be used to exactly integrate the function. However, recall that the nonlinear flux evaluation using solution expansion is a rational function of polynomials therefore preventing exact integration. It has been shown that using up to $4N$ quadrature points in each spatial direction, discretizations utilizing either the Lax-Friedrichs or Roe flux function fail for under-resolved turbulence simulations [109].

2.4.2 Modal Decomposition Filtering

An alternative strategy for polynomial dealiasing is modal decomposition filtering. Hierarchical modal basis functions, ϕ, allow for classification of solution modes based on polynomial degree, e.g. the first solution mode is the average of the solution (0^{th}-degree), the last solution mode is of the highest polynomial degree (N^{th}-degree). By decomposing a solution into its modal solution expansion coefficients, the solution can be filtered by setting higher-order modes to zero or by scaling the higher-order coefficients by a factor, $\alpha \in [0, 1]$. In the case where the higher-order solution modes are set to zero, this is referred to a sharp modal cut-off filter. For the sharp cut-off filter, one may choose a solution cut-off order, P_c, in which any

solution modes above the P_c are set to 0. Note that by choosing $P_c = 0$, all higher-order solution content is removed therefore resulting in cell-averaged solution.

The nodal basis functions used within this work are not hierarchical, i.e. every basis function contains high-order solution content. So removing higher-order solution modes is not possible in a trivial manner. The procedure for modal decomposition filtering first involves transforming the nodal solution coefficients to a hierarchical modal representation, applying a filtering mechanism on the modal coefficients, then transforming back into the nodal representation. We demonstrate this modal decomposition filtering technique for one-dimensional problems; three-dimensional modal decomposition filtering naturally extends this procedure which follows the outline by Gassner et. al. [108].

To perform the transformation from nodal basis representation to the modal basis representation, we first define modal basis functions to be the Legendre polynomials represented as ϕ defined recursively from Bonnet's formula [110]:

$$\begin{aligned} \phi_0(\xi) &= 1 \\ \phi_1(\xi) &= \xi \\ \phi_{n+1}(\xi) &= \frac{(2n+1)\xi\phi_n(\xi) - n\phi_{n-1}(\xi)}{n+1} \end{aligned} \tag{2.75}$$

Next, we define the modal mass matrix, $[\mathbf{M}]$, and mixed mass matrix, $[\mathbf{C}]$, as:

$$[\mathbf{M}] = M_{ij} = \int_E \phi_i(\xi)\phi_j(\xi)\mathrm{d}\xi, \quad 1 \leq i,j \leq N \tag{2.76}$$

$$[\mathbf{C}] = C_{ij} = \int_E \phi_i(\xi)\ell_j(\xi)\mathrm{d}\xi, \quad 1 \leq i,j \leq N \tag{2.77}$$

where ℓ_j are the nodal basis functions. We denote the nodal solution coefficients as $\tilde{u}_j$ and the modal solution coefficients as b_j with relation to the flow variables:

$$u(\xi) = \sum_{j=1}^{N} \ell_j(\xi)\tilde{u}_j = \sum_{j=1}^{N} \phi_j(\xi)b_j \tag{2.78}$$

Multiplying Eqn. (2.78) by ϕ_i and integrating over the standard element:

$$\begin{aligned}
\int_E \left(\sum_{j=1}^{N} \phi_i(\xi) \ell_j(\xi) \tilde{u}_j \right) \mathrm{d}\xi &= \int_E \left(\sum_{j=1}^{N} \phi_i(\xi) \phi_j(\xi) b_j \right) \mathrm{d}\xi \qquad (2.79) \\
\sum_{j=1}^{N} \left(\int_E \phi_i(\xi) \ell_j(\xi) \mathrm{d}\xi \right) \tilde{u}_j &= \sum_{j=1}^{N} \left(\int_E \phi_i(\xi) \phi_j(\xi) \mathrm{d}\xi \right) b_j \qquad \boxed{\text{by: Fubini's Theorem}} \\
\sum_{j=1}^{N} \left(C_{ij} \right) \tilde{u}_j &= \sum_{j=1}^{N} \left(M_{ij} \right) b_j \qquad \boxed{\text{by: Eqns. (2.76) and (2.77)}} \\
[\mathbf{C}] \, \vec{\tilde{u}} &= [\mathbf{M}] \, \vec{b}
\end{aligned}$$

The final line in Eqn. (2.79) is a composed of matrix-vector multiplications. Thus to obtain the modal solution coefficients, $\vec{b}$, from the nodal solution coefficients, $\vec{\tilde{u}}$, we invert the modal mass matrix $[\mathbf{M}]$ on both sides of last equation in Eqn. (2.79):

$$\vec{b} = [\mathbf{M}]^{-1} \, [\mathbf{C}] \, \vec{\tilde{u}} \qquad (2.80)$$

Now that the hierarchical modal basis coefficients have been obtained, a square filter matrix, $[\mathbf{F}]$, can be applied as a matrix-vector product:

$$\widehat{\vec{b}} = [\mathbf{F}] \, \vec{b} \qquad (2.81)$$

where $\widehat{\vec{b}}$ are the filtered modal solution coefficients. The last step in the modal decomposition filtering procedure is to transform the filtered modal coefficients back to nodal basis coefficients to give the filtered nodal solution:

$$
\begin{aligned}
\widehat{\vec{\tilde{u}}} &= [\mathbf{C}]^{-1}[\mathbf{M}]\,\widehat{\vec{b}} && (2.82)\\
&= [\mathbf{C}]^{-1}[\mathbf{M}][\mathbf{F}]\,\vec{b} && \boxed{\text{by: Eqn. (2.81)}}\\
&= [\mathbf{C}]^{-1}[\mathbf{M}][\mathbf{F}][\mathbf{M}]^{-1}[\mathbf{C}]\,\vec{\tilde{u}} && \boxed{\text{by: Eqn. (2.80)}}
\end{aligned}
$$

Lastly, we can introduce the matrices $[\mathbf{B}]$ and $\widehat{[\mathbf{F}]}$:

$$
\begin{aligned}
[\mathbf{B}] &= [\mathbf{M}]^{-1}[\mathbf{C}] && (2.83)\\
\widehat{[\mathbf{F}]} &= [\mathbf{B}]^{-1}[\mathbf{F}][\mathbf{B}] && (2.84)
\end{aligned}
$$

resulting in the final filtered nodal solution coefficients written as follows:

$$
\begin{aligned}
\widehat{\vec{\tilde{u}}} &= [\mathbf{C}]^{-1}[\mathbf{M}][\mathbf{F}][\mathbf{M}]^{-1}[\mathbf{C}]\,\vec{\tilde{u}} && \boxed{\text{by: Eqn. (2.82)}}\ (2.85)\\
&= [\mathbf{B}]^{-1}[\mathbf{F}][\mathbf{B}]\,\vec{\tilde{u}} && \boxed{\text{by: Eqn. (2.83)}}\\
&= \widehat{[\mathbf{F}]}\vec{\tilde{u}} && \boxed{\text{by: Eqn. (2.84)}}
\end{aligned}
$$

The filter matrix $[\mathbf{F}]$ in Eqn. (2.81) is diagonal with its entries being 0, 1, or α. If all entries are 1 giving the identity matrix, the filtering operation returns the original solution. To obtain a cut-off filter of order P_c, all diagonal entries of the filter matrix are 1 up to and including the P_c diagonal entry with the rest of the entries 0.

Alternative to the sharp cut-off filter, a popular filter is the Hesthaven exponential modal filter [94]. The entries of the exponential filter matrix are computed as follows:

$$
\mathbf{F}_{jj} = \begin{cases} 1.0, & \text{if } 0 \le j \le P_c \\ \exp\left[-\alpha_\epsilon \left(\frac{j-P_C}{N-P_c}\right)^p\right], & \text{if } P_c < j \le N \end{cases} \qquad (2.86)
$$

where $\alpha_\epsilon = -\log(\epsilon)$ with ϵ machine zero. Note that the final filter matrix can be assembled as preprocessing step as it does not have a dependence on the solution.

The filtering operation can be applied at the discretion of the user, e.g. always apply the modal decomposition filter. This work deploys a resolution indicator [111] which estimates the solution smoothness by examining the influence of the highest-order modal coefficient.The resolution indicator is denoted s_e and calculated as follows:

$$u(\xi) = \sum_{i=1}^{N} b_i \phi_i(\xi), \quad \widehat{u}(\xi) = \sum_{i=1}^{N-1} b_i \phi_i(\xi), \quad s_e = \log_{10}\left(\frac{(u-\widehat{u}, u-\widehat{u})_e}{(u,u)_e}\right) \tag{2.87}$$

where $(a,b)_e$ is the L^2 inner-product. The value s_e is problem dependent, thus, the user is required to investigate what threshold value of s_e is used for filtering activation.

2.4.3 Alternative Robustness Strategies

In addition to the dealiasing techniques outlined in the previous sections, alternative solution limiting strategies are available. Some examples are flux limiting [112], positivity-preserving solution limiting [113], and artificial viscosity schemes [114]. Rather than trying to improve the robustness of the standard DG method, an alternative DG formulation via nonlinear flux split forms will be investigated in the next chapter demonstrating superior numerical stability without the need to filtering strategies.

2.5 Standard DG Formulation Results

This section presents validation and computational performance of the standard tensor-product discontinuous Galerkin method for Cartesian meshes. Three model problems are used to demonstrate these properties: a mesh resolution study is performed using the Ringleb [115] flow problem, the Taylor-Green vortex [116, 117], a three-dimensional, diagonally lid-driven cavity flow [118–121]. Lastly, the computational performance and parallel computing performance of the numerical method is examined on multiple supercomputing systems.

2.5.1 Ringleb Flow Mesh Resolution Study

An exact solution is used to verify the accuracy of the finite-element formulation and implementation. Ringleb flow is an exact solution to the two-dimensional steady-state inviscid flow equations and is solved analytically using the hodograph method [122]. Although it is a two-dimensional solution, the three-dimensional equations can be verified by setting the momentum in the z-direction to zero. Characteristic boundary conditions are used in the x- and y-directions and an inviscid wall boundary condition is used in the z-direction. The domain is a $[1, 1]$ square discretized with a Cartesian, hexahedral mesh. The mesh resolution study utilizes multiple meshes ranging in mesh elements [1 x 1 x 1] to [40 x 40 x 1] which are used analyze the error between the analytic solution and the converged numerical solution. The flow is initialized using the analytic solution and then the residual is driven to machine precision. The density of Ringleb flow is shown in Fig. (2.3).

The errors in a finite-element formulation are expected to decrease asymptotically following the power law Ch^{p+1}, where C is a constant, h is the mesh size, and p is the polynomial degree. By comparing to an analytic solution, the difference between the numerical solution the error can be measured. The error is measured using an L_∞-norm which is written as follows:

$$|\mathbf{Q} - \mathbf{Q}^*|_\infty = \max_i |Q_i - Q_i^*| \tag{2.88}$$

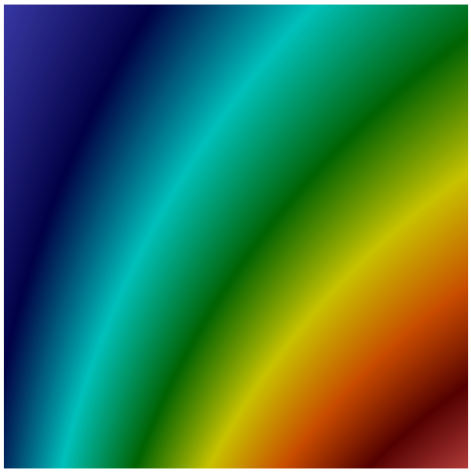


Figure 2.3: Analytic Ringleb flow (density) for the two-dimensional inviscid Euler equations.

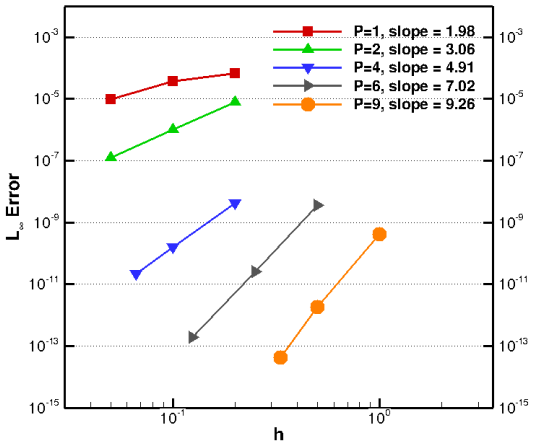


Figure 2.4: Ringleb flow mesh resolution study L_∞-error versus mesh size h.

where Q is the numerical solution and Q^* is the analytic solution. Mesh resolution studies are performed multiple polynomial degrees: $p = 1, 2, 4, 6, 9$. The results of the study are shown in Fig. (2.4) with mesh size h plotted against L_∞-error. The desired design accuracy is achieved for all polynomial degrees noting that at $p = 9$, the L_∞-error is near machine precision thus giving a slight degradation in the slope of the error.

2.5.2 Diagonally Lid-Driven Cavity Flow

The second test case to demonstrate the capability of the three-dimensional finite-element method implementation is a diagonally lid-driven cavity flow. Recent efforts for an extension to the standard two-dimensional lid-driven cavity benchmark [123] have been proposed by Povitsky [118, 124] and Feldman *et al.* [119]. The extension is a steady driven flow in a domain $[0, 1]^3$ with the top flow surface moving horizontally with a 45° angle with respect to the domain walls. The x- and z-directional velocities are of equal magnitude. For this study, a Reynolds number of 1000 is used. Fig. (2.5) depicts the diagonally lid-driven flow.

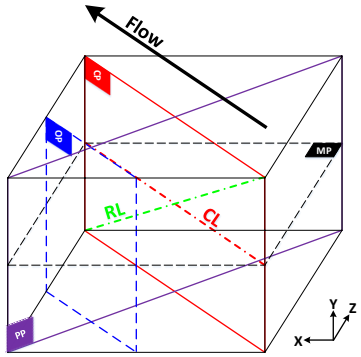


Figure 2.5: Lid-driven cavity flow with forcing at 45° to the x-axis, $Re = 1000$.

Numerical results are presented in Table (2.1) using Feldman et. al. [119] as a comparative reference. The solver used in Feldman et. al. is a second-order conservative finite volume method with a full pressure-velocity coupling discretizing the incompressible Navier-Stokes equations [121, 123]. Their results were performed on 152^3 and 200^3 grids; the latter results are for comparison. Results within are performed at $p = 3$, fourth-order, using a mesh consisting of 32^3 elements. The total degrees-of-freedom accounts to just under 2.1 million whereas the reference used 8 million degrees-of-freedom.

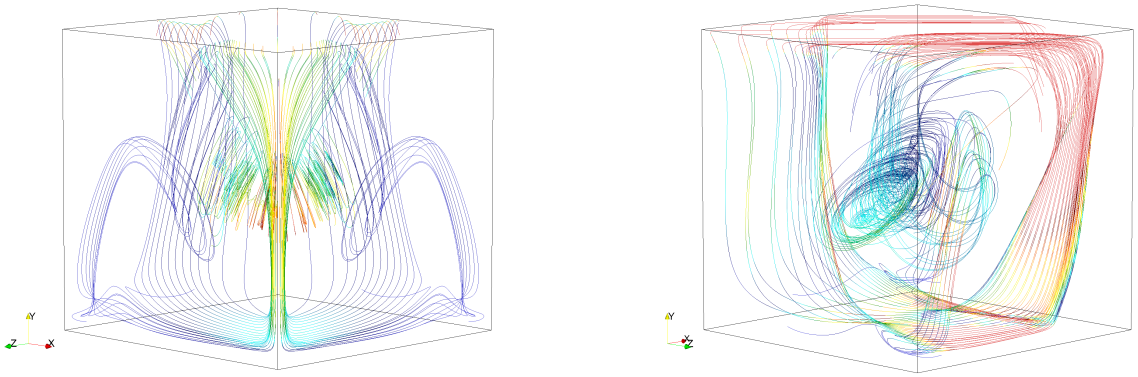
The flow is considered converged when the L_2-norm of the spatial residual is less than 10^{-12}. The results in Table (2.1) demonstrate that the V_x and V_z velocities towards the boundaries of the computational domain are similar in magnitude to the comparison reference. In the center of the domain near $(0.5, 0.5, 0.5)$, the V_x and V_z velocities present the

Diagonally Lid-Driven Cavity Flow
$Re = 1000$

y	$V_x, V_z \cdot 10^3$		$V_y \cdot 10^3$	
	Feldman [119]	**Present**	**Feldman [119]**	**Present**
1.0000	707.1	707.1	0.000	0.000
0.9766	417.8	418.2	5.357	9.735
0.9688	341.3	339.5	8.774	9.804
0.9609	277.2	276.7	12.47	9.622
0.9531	226.6	227.9	16.03	13.90
0.8516	76.82	76.06	30.13	29.99
0.7344	62.56	61.74	22.28	21.95
0.6172	41.77	40.68	5.439	4.832
0.5000	-1.649	-4.218	-34.41	-36.76
0.4531	-31.93	-35.25	-65.23	-68.61
0.2813	-131.0	-130.7	-160.5	-160.1
0.1719	-134.7	-133.0	-138.0	-136.2
0.1016	-143.0	-141.8	-86.68	-85.15
0.0703	-158.8	-157.6	-52.73	-51.60
0.0625	-161.9	-160.2	-43.93	-42.98
0.0547	-162.2	-160.4	-35.40	-34.60
0.0000	0.000	0.000	0.000	0.000

Table 2.1: Velocities along the vertical center line $(0.5, y, 0.5)$. Feldman *et al.* performed on 200^3 grid and present work performed on 32^3 grid at $p = 3$.

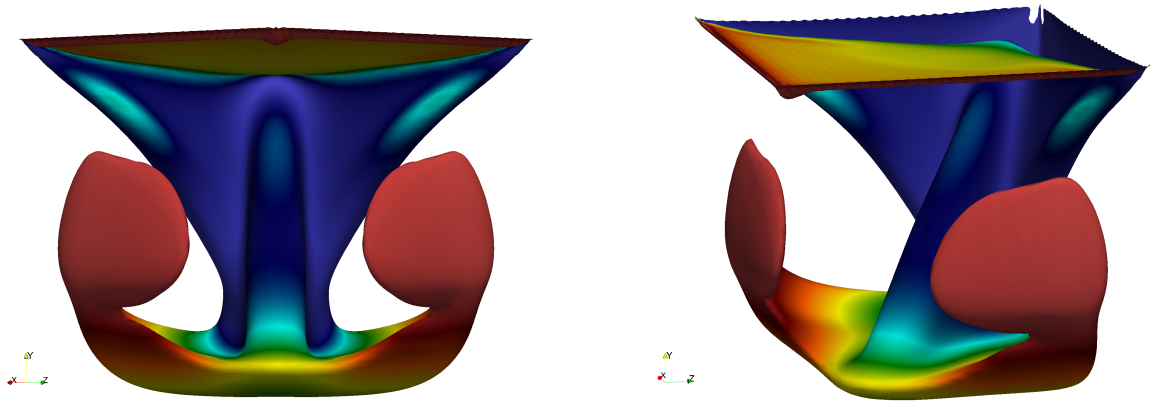
most variation in the domain. The V_y velocities agree more on the lower half of the domain in comparison to the upper half. Fig. (2.6) visualizes the streamlines of the converged flow observed in the center plane, denoted (cp), direction and parallel plane, denoted (pp), direction. Fig. (2.7) visualizes a contour of velocity magnitude and Fig. (2.8) demonstrates the streamlines in the center plane.



(a) flow towards observer

(b) flow left to right

Figure 2.6: Cubic lid-driven cavity flow streamlines colored by velocity magnitude.



(a) flow away from observer

(b) flow left to right

Figure 2.7: Cubic lid-driven cavity flow velocity magnitude contour colored by y-velocity.

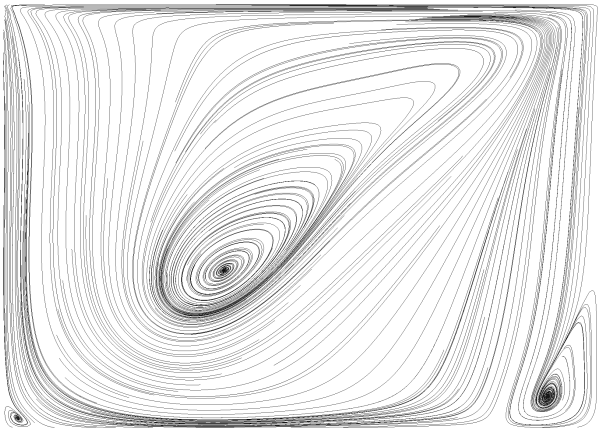


Figure 2.8: Center plane streamlines in the direction of the flow.

2.5.3 Taylor-Green Vortex

The Taylor-Green vortex flow is simulated using the compressible Navier-Stokes equations at $M_0 = 0.1$. The flow is solved on an isotropic domain which spans $[-\pi L, \pi L]$ in each coordinate direction where L is the characteristic length. The initial conditions are given as:

$$\begin{aligned} u &= V_0 \sin(x/L)\cos(y/L)\cos(z/L) \\ v &= -V_0 \cos(x/L)\sin(y/L)\cos(z/L) \\ w &= 0 \\ p &= \rho_0 V_0^2 \left[\frac{1}{\gamma M_0^2} + \frac{1}{16}\left(\cos(2x)+\sin(2y)\right)\left(\cos(2z)+2\right)\right] \end{aligned} \tag{2.89}$$

where u, v, and w are the components of the velocity in the x-, y- and z-directions, p is the pressure and ρ is the density. The flow is initialized to be isothermal $\left(\frac{p}{\rho} = \frac{p_0}{\rho_0} = RT_0\right)$. The initial flow is laminar and subsequently transitions to turbulence, with the creation of small scales, followed by a decay phase similar to decaying homogeneous turbulence. Fully periodic boundary conditions are utilized. Fig. (2.9) demonstrates a volume rendering of the density. Fig. (2.10) shows the temporal evolution of the kinetic energy dissipation rate $-\frac{dE_k}{dt}$ for the Taylor-Green vortex computed at $M_0 = 0.1$, $Pr = 0.71$, $Re = \frac{\rho_0 V_0 L}{\mu} = 1600$, and 256 degrees-of-freedom in each coordinate direction with comparison to an incompressible spectral solver with 512 degrees-of-freedom in each coordinate direction [125]. The temporal evolution of the kinetic energy integrated over the domain Ω is calculated as follows:

$$E_k = \frac{1}{\rho_0 \Omega}\int_\Omega \rho \frac{\vec{v}\cdot\vec{v}}{2} d\Omega \tag{2.90}$$

As the solution order increases, the dissipation rate, $-\frac{dE}{dt}$, which is computed via a forward finite-difference, tends towards the incompressible spectral code. The DG scheme becomes numerically less dissipative with higher-order basis functions resulting in the solver's ability to match more accurately the spectral solver results. Reference [41] demonstrates similar dissipation curves with an unstructured mesh version of the current approach.

For further solver validation, comparison to a three-dimensional mixed-element mesh DG solver with shock capturing [21] is conducted using the Taylor-Green vortex problem on identical hexahedral meshes. Results are demonstrated in Fig. (2.11) for $p = 4$, fifth-order, 64^3 mesh. Kinetic energy dissipation, enstrophy, and pressure dilation contribution to kinetic energy dissipation results are plotted in Fig. (2.11)(a), (b), and (c), respectively. Enstrophy is computed as follows:

$$\epsilon = \frac{1}{\rho_0 \Omega} \int_\Omega \rho \frac{\vec{\omega} \cdot \vec{\omega}}{2} d\Omega \tag{2.91}$$

where ω is vorticity. The pressure dilation contribution, ϵ_3, is computed as follows:

$$\epsilon_3 = -\frac{1}{\rho_0 \Omega} \int_\Omega p \nabla \cdot \vec{v} \; d\Omega \tag{2.92}$$

As seen from Fig. (2.11), the DG solver developed herein gives nearly identical solution curves for kinetic energy dissipation and enstrophy as the mixed-element unstructured mesh, shock-capturing, discontinuous Galerkin solver when using identical hexahedral structured meshes. The pressure dilation, ϵ_3, curve slightly different compared to the unstructured solver. Overall, the trends in the ϵ_3 results are very similar and can be attributed to both solvers discretizing the compressible Navier-Stokes equations and using the symmetric interior penalty method for the viscous flux calculations.

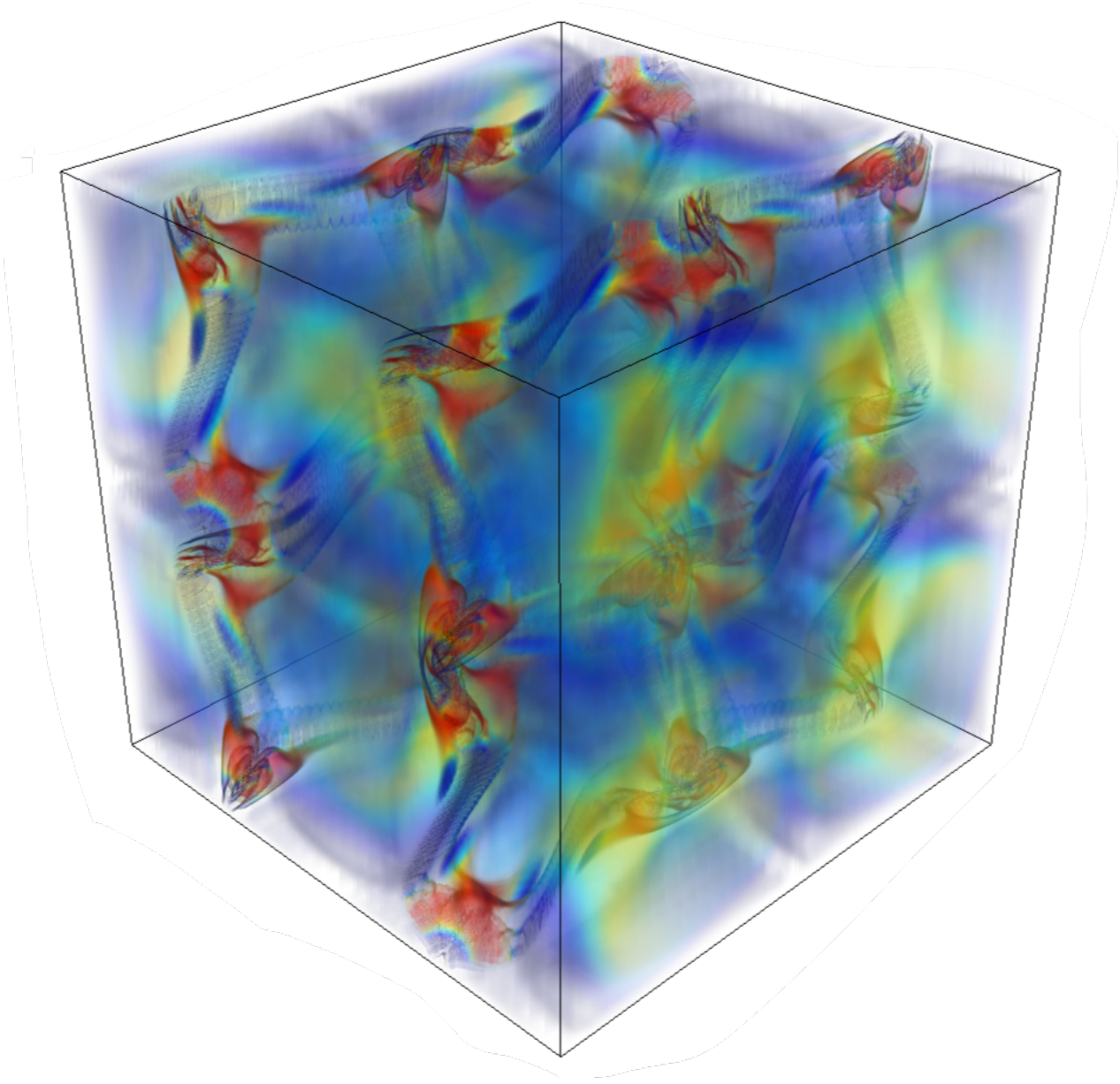


Figure 2.9: Taylor-Green vortex volume rendering, fourth-order accuracy ($p = 3$), opacity based on vorticity and colored by density.

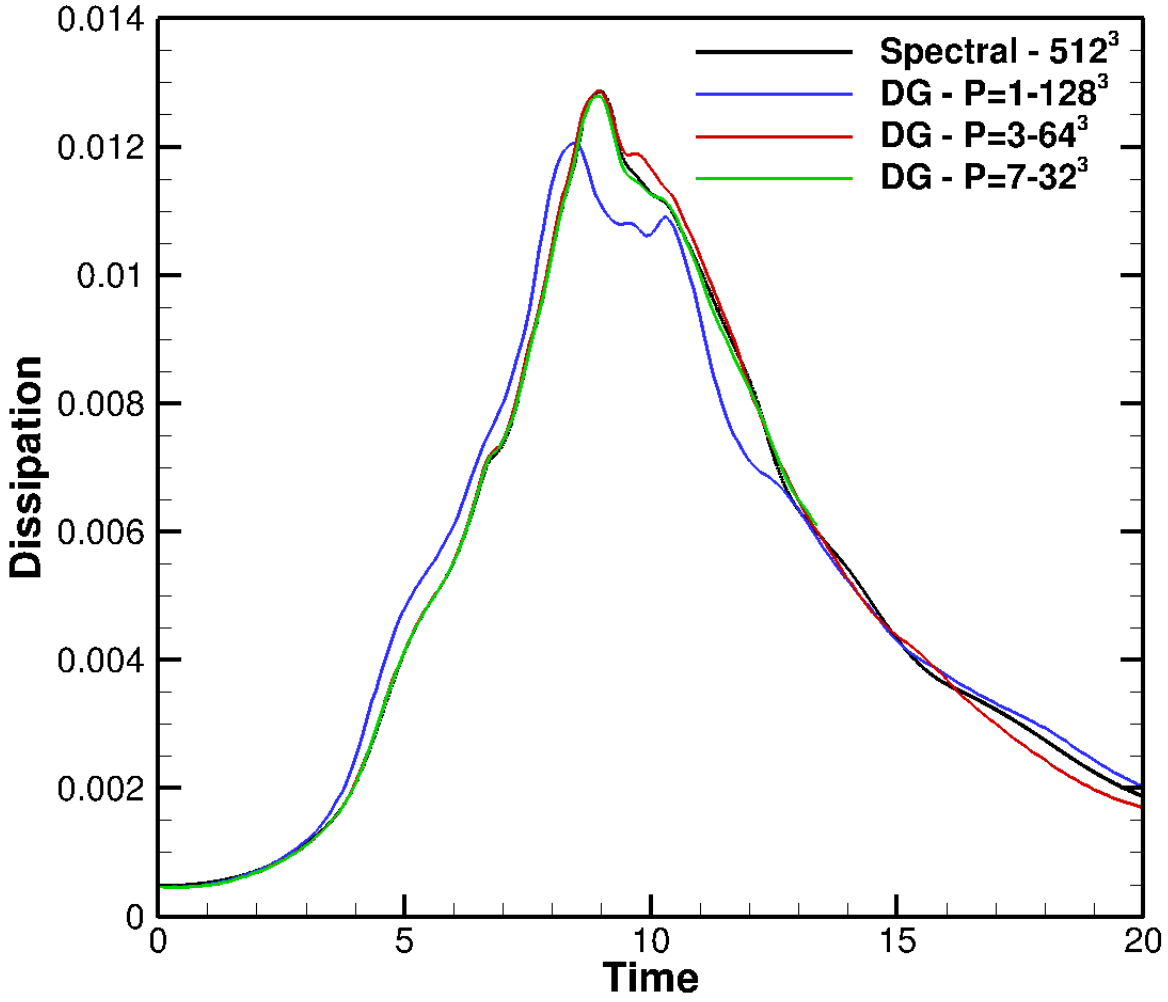
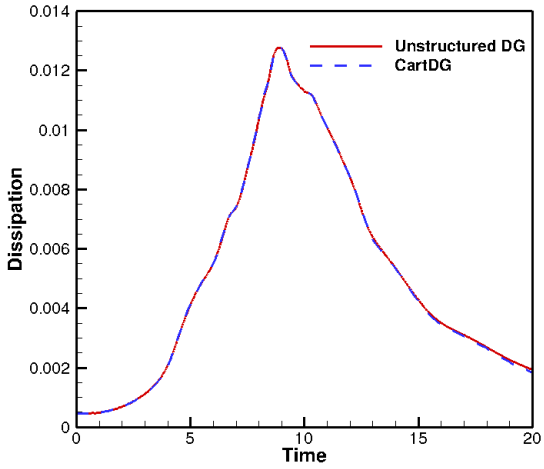
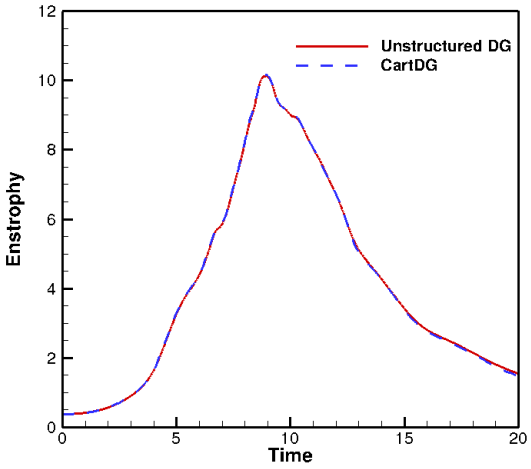


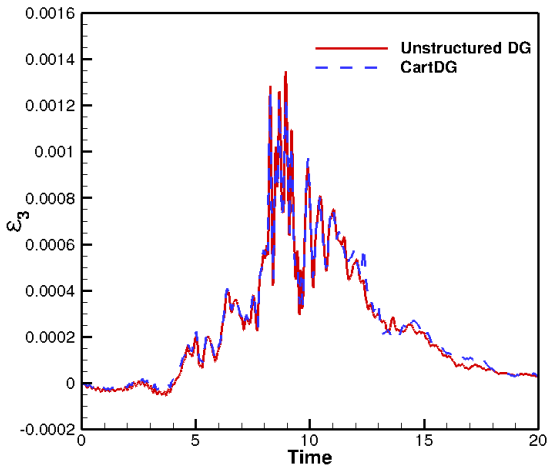
Figure 2.10: Taylor-Green vortex dissipation over time at $M = 0.1$, $Re = 1600$ computed using 256^3 degrees-of-freedom.



(a) kinetic energy dissipation



(b) enstrophy



(c) ϵ_3

Figure 2.11: Taylor-Green Vortex results comparison between an unstructured mesh DG solver and present work (annotated CartDG): $p = 4, 64^3$ mesh at $M = 0.1$, $Re = 1600$.

2.6 Computational and Parallel Performance Results

This section examines the computational performance and parallel scalability of the discontinuous Galerkin solver developed for this work.

2.6.1 Computational Performance

The numerical discretization of tensor-product collocation basis and test functions in a Cartesian coordinate system introduces several cost-saving simplifications for performance. Designing a nodal collocation method by choosing the solution interpolation points the same as the integration points saves a solution interpolation operation, namely, the volume integration term. Additionally, the tensor-product basis functions convert costly interpolations and integrations to one-dimensional products. Combining these properties makes the DG implementation competitive with finite-difference discretizations on a cost per degree-of-freedom basis. Fig. (2.12) demonstrates the computational cost of residual evaluation per degree-of-freedom for a non-optimized collocated nodal DG implementation (non-sum-factorization operations) compared to a finite-difference method for the three-dimensional inviscid Euler equations. Higher-order discretizations for the DG method demonstrate the higher computational cost per degree-of-freedom. This is attributed to higher arithmetic intensity of the DG method which is the number of floating-point operations (FLOP) per byte moved in computer memory. Modern architectures are designed to perform lots of FLOPs. However, the rate at which data transfers occur in computer memory is much slower than the rate to perform FLOPs.

To achieve higher computational performance, tensor-product operations via sum-factorization shown in Eqn. (2.27) are utilized with matrix-matrix multiplication implementations when possible. The matrix-matrix multiplications are performed using efficient math libraries such as Intel's Math Kernel Library (MKL). Additionally, since the normal vectors in a Cartesian coordinate system only contain one component, e.g. $\vec{n} = (1, 0, 0)$, specialized routines for flux calculations are constructed for each coordinate direction to eliminate floating point operations related to the orthogonal coordinate directions.

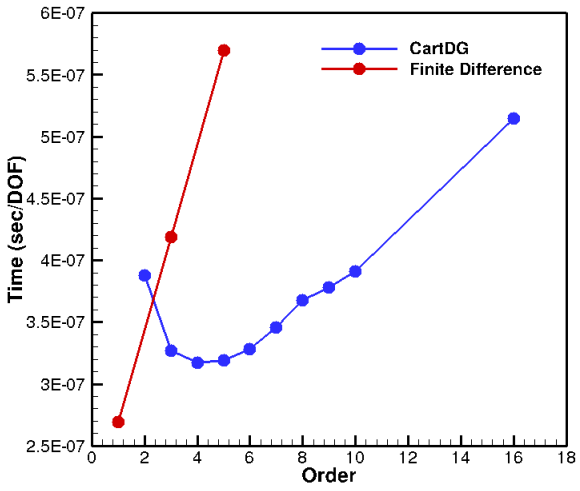


Figure 2.12: Residual cost per degrees-of-freedom comparison of non-optimized collocated nodal DG method versus a finite-difference method for the three-dimensional inviscid Euler equations.

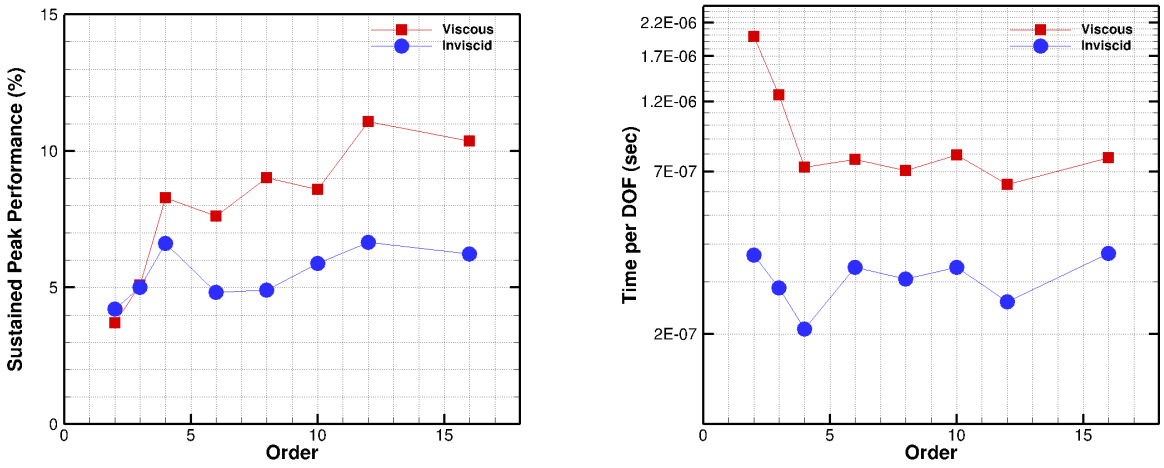
Advanced computer vectorization and data alignment techniques are employed throughout the implementation. At higher polynomial degrees, it is advantageous to rearrange the solution order in memory. Traditionally, for column-major memory storage, the solution vector is stored by ordering flow fields first, followed by solution modes and elements, where the flow fields are the conservative flow variables. However, in order to perform single-instruction-multiple-data (SIMD) vectorization of flux calculations, reordering the storage to modes, fields and elements is needed. When this action is performed, vectorization allows for either two, four, or eight flux evaluations to be performed in parallel with the Intel AVX, AVX-2, or AVX-512 vector instructions, respectively.

To demonstrate the higher computational efficiency via these advanced mathematical and computational techniques, performance statistics for a wide range of polynomial degrees are provided. The sustained peak performance of the inviscid and viscous residual evaluations are shown in Fig. (2.13)(a) on an Intel Core i7-5960X Haswell-E processor with eight CPU cores, clock speed of 3GHz and 4GB of memory per core. For reference, the TAU benchmark for this processor is 5.25 seconds with a theoretical peak performance of 384 Giga-floating-

point-operations-per-second (GFLOPS). As seen from the figure, the tensor-product based FEM formulation achieves roughly 12% sustained compute peak performance for evaluation of the residual of the three-dimensional Navier-Stokes equations at high polynomial degrees. Fig. (2.13)(b) demonstrates the time per degree-of-freedom for inviscid and viscous residual evaluations. Overall, the time required for a viscous residual evaluation is higher than for an equivalent inviscid residual evaluation. This is because there are more operations required for the viscous residual evaluation. For higher polynomial degrees, the time per degree-of-freedom becomes approximately constant for both inviscid and viscous residuals. The performance suffers at low polynomial degrees because the number of FLOPs is very low compared to the number of bytes transferred from dynamic random-access-memory (DRAM) required for the calculation. This severely limits the performance because the computation becomes communication-bound meaning the performance is limited by the communication bandwidth of the CPU.

Computational comparison to a general finite-element method (non-tensor-product basis DG method) is shown in Fig. (2.14)(a) which demonstrates the sustained peak performance on the NWSC-1 Yellowstone supercomputer using 128 Intel Xeon E5-2670 cores with a clock speed of 2.6Ghz and 2GB per core memory. The general FEM implementation is able achieve nearly 65% sustained peak performance whereas the tensor-product formulation achieves 10% of peak performance. However, the general FEM formulation requires significantly more floating point operations, most of which are multiplication and addition of zeros for a nodal collocated basis. This is demonstrated in Fig. (2.14)(b) comparing the time per degree-of-freedom for a viscous residual evaluation. The cost of the general FEM formulation is nearly 10 times larger than that required by the tensor-product formulation at high polynomial degrees. The only polynomial degree for which the general FEM formulation is more efficient is $p = 0$, first-order solution accuracy. This polynomial degree is never used in simulations because of its severe inaccuracies and large numerical dissipation.

Hindenlang et. al. [45] report a computational time per degree-of-freedom for a single residual evaluation is $2.0e-06$ seconds for their tensor-product DG solver at sixth-order compared to their in-house compact finite-difference solver at $4.0e-06$ seconds. Reference [41]



(a) Sustained peak performance.

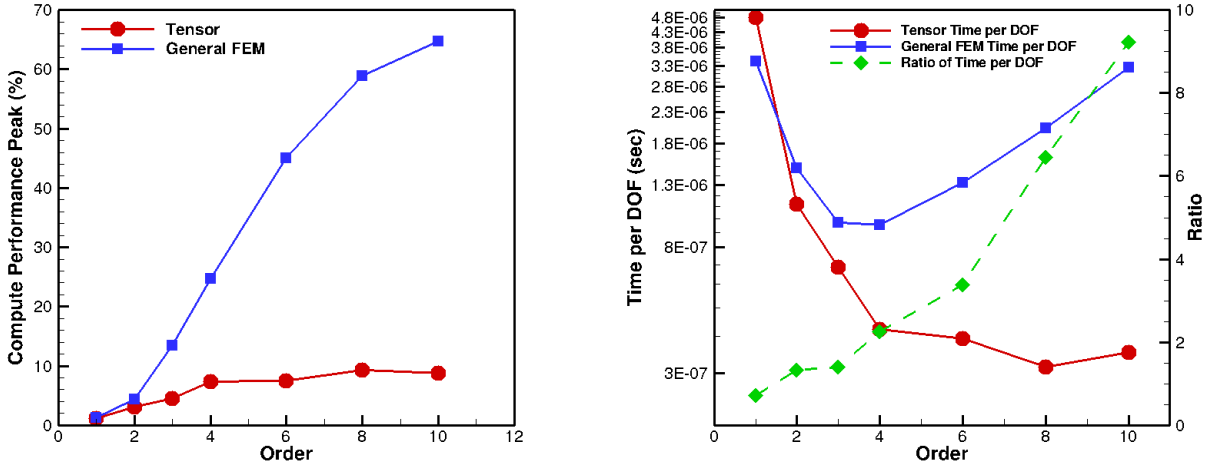
(b) Time per degree-of-freedom.

Figure 2.13: Computational performance for inviscid and viscous calculations for $p = 1 - 15$ polynomial degrees using collocated nodal tensor-product DG formulation.

report their similar unstructured mesh DG solver averaging $7.5e-07$ seconds per degree-of-freedom for a single residual evaluation which is comparable to the cost for a Navier-Stokes residual evaluation using the OVERFLOW finite-difference solver. Reference [41] also report they were able to obtain a computationally cost independent of solution order up to 16^{th}-order through optimized computational kernels and carefully alignment of data and unit-strided memory operations. Current implementation of the DG solver herein achieves a similar computational cost to Reference [41].

2.6.2 Parallel Performance

In additional to the high computational efficiency of the DG method, its main advantages are based on its parallel scalability. Distributed memory parallelism is realized through the MPI application program interface. The DG algorithm is inherently parallel, since all elements communicate only with their direct neighbors. Only element surface data is needed by neighboring elements thus minimizing the data transfer. This is an advantage over finite-difference methods, which couple distant neighboring elements and thus grow the fringe area.



(a) Sustained peak performance.

(b) Time per degree-of-freedom.

Figure 2.14: Computational performance for $p = 1 - 9$ polynomial degrees using collocated nodal tensor-product DG formulations versus the non-tensor-product DG formulation.

In addition to the finite-element method's straight-forward parallelism, the Cartesian mesh framework allows for nearly optimal load balancing by requiring all domain decomposition blocks to be identical. This method also takes advantage of the Cartesian virtual topology framework in MPI simplifying the implementation. The DG algorithm can be split into the two calculation categories: volume integrations and the surface integrations. The volume integral calculation depends only on element-local degrees-of-freedom whereas the surface integral calculation requires neighboring element data. This fact can help to hide communication latency by overlapping computation and data transfer. This is achieved by communicating surface data while simultaneously performing volume integral operations.

Parallel scalability results are obtained on Yellowstone [126]. The strong scalability results are obtained by performing a simulation of the Taylor-Green vortex problem using a 128 x 128 x 128 mesh. This mesh equates to approximately 260 million DOF, 1 billion DOF, and 2 billion DOF for $p = 4$, 7, and 9, respectively. Strong scaling results up to 32,768 cores for polynomial degrees of 4, 7, and 9 are shown in Fig. (2.15). The scaling improves as the solution order increases. At 32,768 cores, the problem contained 8,000, 32,768, and 64,000 DOF on each core for $p = 4, 7$, and 9, respectively.

Mira Scalability: One MPI Rank per Core			
Cores	Time (sec)	Strong Scalability	Speedup
8,192	161.32199	100.0%	8192.00
16,384	80.76674	99.87%	16362.54
32,768	40.39420	99.84%	32716.23
65,536	20.43908	98.66%	64657.82
131,072	10.40850	96.87%	126968.14
262,144	5.27547	95.56%	250507.43
524,288	2.58876	97.37%	510493.98

Mira Scalability: Two MPI Ranks per Core				
Cores	MPI Ranks	Time (sec)	Strong Scalability	Time Ratio
8,192	16,384	97.297702	100.0%	1.658
16,384	32,768	48.75938	99.77%	1.656
32,768	65,536	24.77189	98.19%	1.631
65,536	131,072	12.39752	98.10%	1.649
131,072	262,144	6.311845	96.34%	1.649
262,144	524,288	3.21362	94.61%	1.642
524,288	1,048,576	1.57219	96.70%	1.647

Table 2.2: Strong scalability to over one million MPI ranks using ALCF Mira.

A strong scalability test is conducted on the Department of Energy Mira supercomputer using a range of cores from 8,192 to 524,288 with one MPI rank per core and two MPI ranks per core. The strong scalability results are obtained by performing a simulation of the Taylor-Green vortex problem using a 512 x 512 x 512 mesh at $p = 4$, fifth order accuracy. The problem contains approximately 16.8 billion DOF. Table (2.2) demonstrates execution times for one MPI rank per core and two MPI ranks per core. Fig. (2.18) demonstrates the time to solution, and the strong scaling percentage up to 524,288 cores using one and two MPI ranks per core. The strong scaling results assume the time to solution for 8,192 cores is ideal. Note that executing two MPI ranks per core is approximately 64% faster in execution time in comparison to one MPI rank per core. This trend holds to over one million MPI ranks.

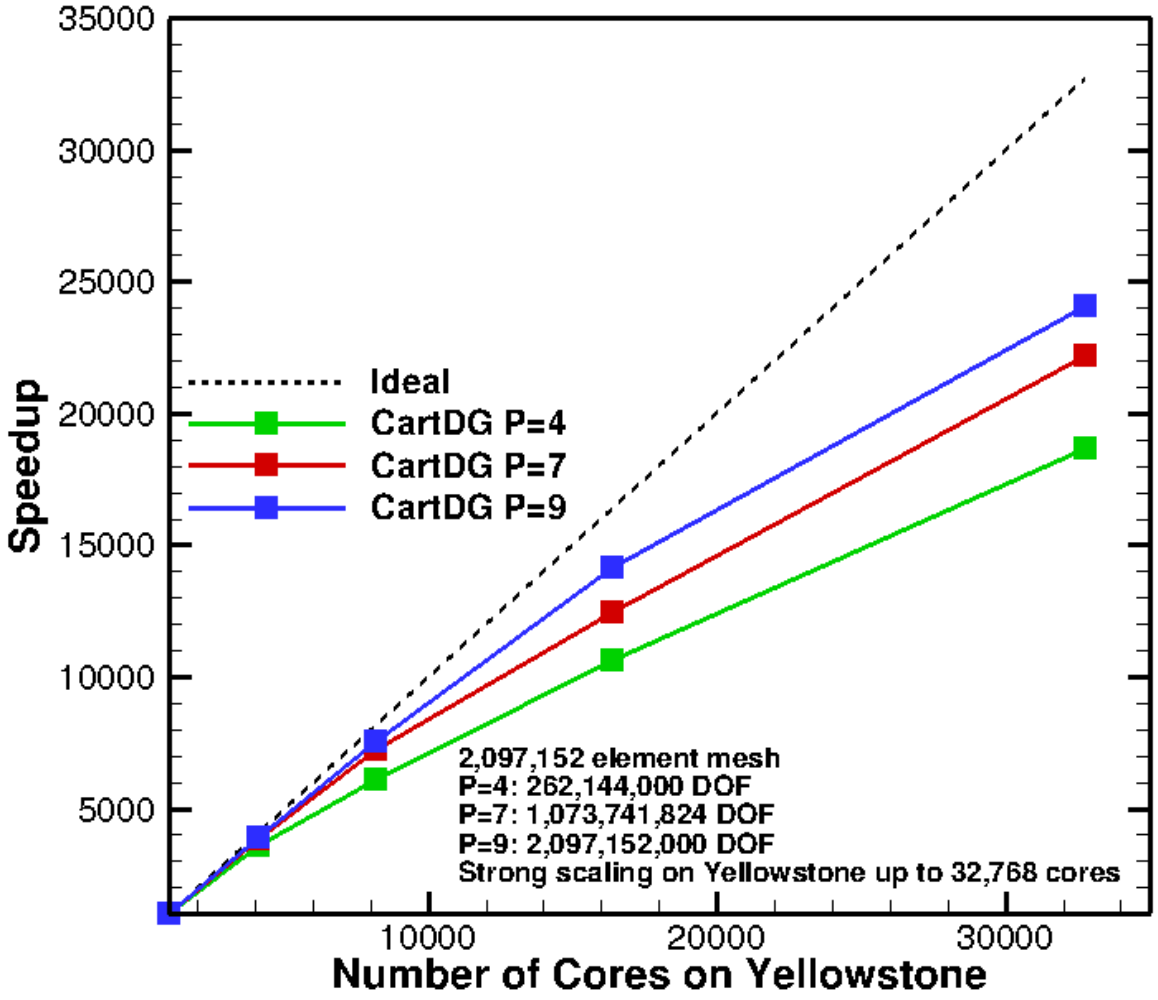


Figure 2.15: Strong scaling for polynomial degrees 4, 7, and 9 on the NCAR-Wyoming Supercomputer.

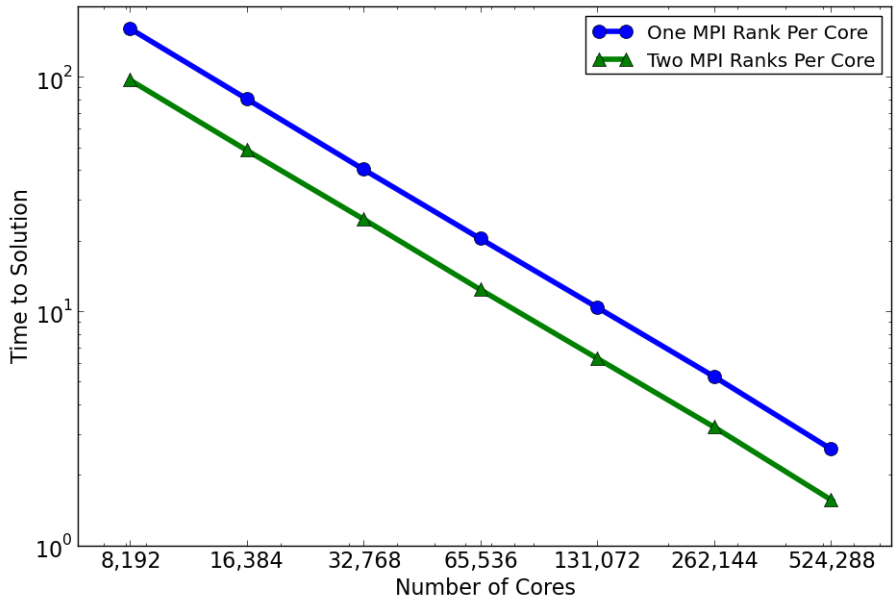


Figure 2.16: Time to Solution

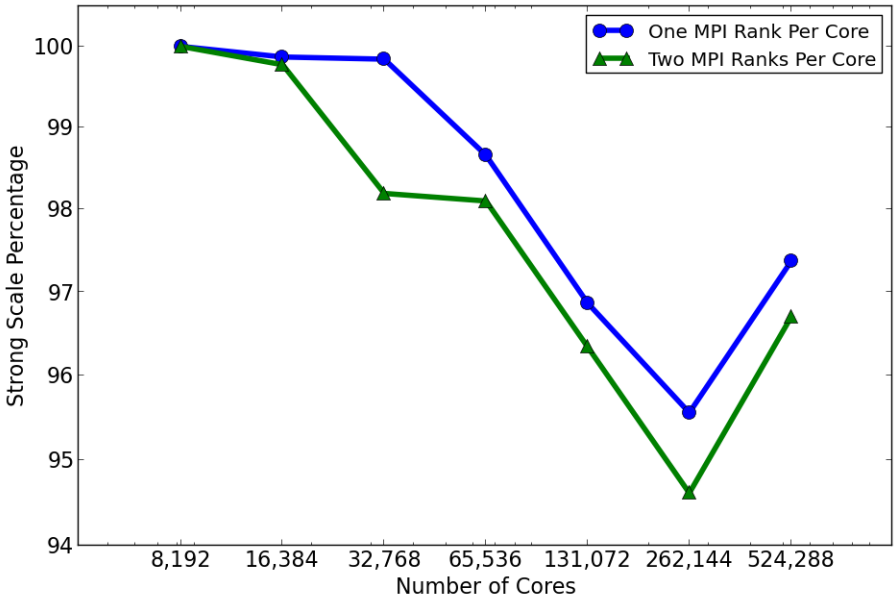


Figure 2.17: Strong Scaling Percentage

Figure 2.18: Strong scalability on DoE's Mira supercomputer up to 524,288 cores on a problem containing nearly 16.8 billion degrees-of-freedom.

Chapter 3

Split Form Discontinuous Galerkin Methods with Summation-By-Parts Property

Standard discontinuous Galerkin methods can suffer from numerical instability issues in under-resolved turbulent simulations. Various dealiasing strategies such as over-integration and modal decomposition filtering were presented. An alternative strategy for stabilizing higher-order DG discretizations is nonlinear flux splitting. Split form DG methods allow for development of discretizations with native numerical stability properties. Even more so, split form schemes can be developed to be kinetic energy preserving as defined by Jameson [127]. The property known as summation-by-parts (SBP) [128] is essential to building a split form DG scheme. The SBP operator is a discrete analog of the continuous integration-by-parts procedure. This allows the construction of provably entropy-stable schemes [129].

This chapter outlines the summation-by-parts property, flux split formulations and implementation strategies for DG methods. Additional details regarding particular split forms with the kinetic energy preservation property are examined. Finally, numerical investigations are performed demonstrating the kinetic energy preserving schemes and increased numerical stability compared to the standard DG method.

3.1 Summation-By-Parts Property

Recall the continuous form of the integration-by-parts property of two continuously differentiable functions $u(x) : [x_L, x_H] \to \mathbb{R}$ and $v(x) : [x_L, x_H] \to \mathbb{R}$:

$$\int_{x_L}^{x_H} u(x) \frac{\partial v(x)}{\partial x} \mathrm{d}x = u(x)v(x)\Big|_{x_L}^{x_H} - \int_{x_L}^{x_H} \frac{\partial u(x)}{\partial x} v(x) \mathrm{d}x \tag{3.1}$$

The summation-by-parts operator discretely mimics the integration-by-parts property. First, define a nearly skew-symmetric matrix $[\mathbf{Q}]$ as follows:

$$[\mathbf{Q}] := [\mathbf{M}][\mathbf{D}] \quad \text{with} \quad [\mathbf{Q}] + [\mathbf{Q}]^T = [\mathbf{B}] \tag{3.2}$$

where the matrix $[\mathbf{B}]$ is a diagonal matrix defined as $\mathrm{diag}(-1, 0, ..., 0, 1)$, the matrix $[\mathbf{M}]$ is a discrete mass matrix, and the matrix $[\mathbf{D}]$ is a discrete differentiation matrix. We rearrange Eqn. (3.2) to mimic integration-by-parts discretely as follows:

$$[\mathbf{D}] = [\mathbf{M}]^{-1}[\mathbf{Q}] = [\mathbf{M}]^{-1}[\mathbf{B}] - [\mathbf{M}]^{-1}[\mathbf{Q}]^T \tag{3.3}$$

Notice that the $[\mathbf{B}]$ matrix is the boundary condition operator, and the rows of $[\mathbf{Q}]$ are undivided differences. The particular choice of the collocated Lagrange polynomials basis functions constructed with the Lobatto-Gauss-Legendre (LGL) quadrature points as solution nodes provides the DG method with the SBP property. From Chapter 2, we define the matrix $[\mathbf{M}]$ as the one-dimensional diagonal mass-lumped mass matrix as follows:

$$[\mathbf{M}] = \mathrm{diag}(\omega_0, ..., \omega_N) \tag{3.4}$$

where ω_i is the i-th one-dimensional LGL quadrature weight. Further, define the derivative matrix $[\mathbf{D}]$ as follows:

$$[\mathbf{D}] = D_{ij} = \frac{\mathrm{d}\ell_j(\xi_i)}{\mathrm{d}\xi}, \quad i, j = 0, ..., N \tag{3.5}$$

where $\ell_j(\xi_i)$ is the one-dimensional Lagrange polynomial constructed from the LGL quadrature points evaluated at the i-th LGL node. Note that D_{ij} corresponds to the differential matrix acting on the basis functions used in the strong form volume integration in Chapter 2. Through this construction, the SBP property in Eqn. (3.2) is satisfied as follows:

$$([\mathbf{M}][\mathbf{D}]) + ([\mathbf{M}][\mathbf{D}])^T = [\mathbf{B}]$$

3.2 Split Form Methods

Using the DG formulation developed with the SBP property, construction of nonlinear flux split formulations known from the finite difference community [130–132] is presented. First, reintroduction of the governing equations in general form is as follows:

$$\frac{\partial \mathbf{Q}\left(\boldsymbol{x},t\right)}{\partial t}+\vec{\mathcal{L}}_X\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)+\vec{\mathcal{L}}_Y\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)+\vec{\mathcal{L}}_Z\left(\mathbf{Q}\left(\boldsymbol{x},t\right)\right)=0 \tag{3.6}$$

where $\vec{\mathcal{L}}_X, \vec{\mathcal{L}}_Y, \vec{\mathcal{L}}_Z$ are the nonlinear flux functions in the x-, y-, and z-directions, respectively. Using the coordinate transformation defined in Section (2.1.4) of Chapter 2, the general governing equations can be written as follows:

$$J\frac{\partial \tilde{\mathbf{Q}}}{\partial t}+\tilde{\vec{\mathcal{L}}}_X\left(\mathbf{Q}\right)+\tilde{\vec{\mathcal{L}}}_Y\left(\mathbf{Q}\right)+\tilde{\vec{\mathcal{L}}}_Z\left(\mathbf{Q}\right)=0 \tag{3.7}$$

To access split form schemes via the DG method, a special formulation known as the DG spectral element method (DGSEM) [133] is required. The derivation and design choices of the DG method in Chapter 2, namely the choice of p-degree Lagrange interpolation polynomials with abscissa begin the GL or LGL quadrature points, is actually the DG spectral element method. By choosing the nodal points (GL or LGL points) to define both the Lagrange interpolating polynomial and the quadrature rule, we gain a discrete orthogonality of the basis functions and quadrature evaluation, meaning the Kronecker-delta property defined by Eqn. (2.24) in Chapter 2 is gained through collocation.

As defined in Eqns. (3.4) and (3.5), the strong form DGSEM method is constructed following the procedures in Chapter 2 using Lagrange basis functions and LGL points for both the solution points and the quadrature points. Inverting the one-dimensional mass matrix M, which is composed of the one-dimensional quadrature weights, the inviscid volume flux terms at LGL node (i,j,k) read as follows:

$$
\begin{aligned}
\left(\tilde{\vec{\mathcal{L}}}_X(\mathbf{Q})\right)_{i,j,k} &\approx \tfrac{1}{\omega_i}\left(\delta_{iN}\left[\tilde{\mathcal{F}}^* - \tilde{\mathcal{F}}\right]_{Njk} - \delta_{i1}\left[\tilde{\mathcal{F}}^* - \tilde{\mathcal{F}}\right]_{1jk}\right) &&+ \sum_{m=1}^{N} \mathbf{D}_{im}(\tilde{\mathcal{F}})_{mjk} \\
\left(\tilde{\vec{\mathcal{L}}}_Y(\mathbf{Q})\right)_{i,j,k} &\approx \tfrac{1}{\omega_j}\left(\delta_{jN}\left[\tilde{\mathcal{G}}^* - \tilde{\mathcal{G}}\right]_{iNk} - \delta_{j1}\left[\tilde{\mathcal{G}}^* - \tilde{\mathcal{G}}\right]_{i1k}\right) &&+ \sum_{m=1}^{N} \mathbf{D}_{jm}(\tilde{\mathcal{G}})_{imk} \\
\left(\tilde{\vec{\mathcal{L}}}_Z(\mathbf{Q})\right)_{i,j,k} &\approx \tfrac{1}{\omega_k}\left(\delta_{kN}\left[\tilde{\mathcal{H}}^* - \tilde{\mathcal{H}}\right]_{ijN} - \delta_{k1}\left[\tilde{\mathcal{H}}^* - \tilde{\mathcal{H}}\right]_{ij1}\right) &&+ \sum_{m=1}^{N} \mathbf{D}_{km}(\tilde{\mathcal{H}})_{ijm}
\end{aligned}
\tag{3.8}
$$

where $\tilde{\mathcal{F}}^*$, $\tilde{\mathcal{G}}^*$, $\tilde{\mathcal{H}}^*$ are the x-, y-, z-direction numerical flux functions evaluated at element interfaces, respectively. Also, $(\tilde{\mathcal{F}})_{mjk} = \tilde{\mathcal{F}}\left(\mathbf{Q}_{mjk}\right)$ is the non-linear inviscid flux evaluation, with $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{H}}$ defined similarly. The split form DG method is based on the property of diagonal norm SBP operators, discovered by Fisher and Carpenter [134], where the differentiation matrix $\mathbf{D}$ can be interpreted as a sub-cell volume differencing operator. Carpenter and Fisher constructed entropy-stable split forms of the volume terms using the following:

$$
\left(\tilde{\vec{\mathcal{L}}}_X(\mathbf{Q})\right)_{i,j,k} \approx \frac{1}{\omega_i}\left(\delta_{iN}\left[\tilde{\mathcal{F}}^* - \tilde{\mathcal{F}}\right]_{Njk} - \delta_{i1}\left[\tilde{\mathcal{F}}^* - \tilde{\mathcal{F}}\right]_{1jk}\right) + \sum_{m=1}^{N} 2\mathbf{D}_{im} F^{\#}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}) \tag{3.9}
$$

where a two-point entropy-stable volume flux $F^{\#}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk})$ is used. Gassner et al. [135] demonstrated every symmetric and consistent two-point numerical flux $F_X^{\#}$ serves as a novel DGSEM split form. Thus, the sub-cell differencing operators replace the the original differentiation matrix operators in Eqn. (3.8) as follows:

$$
\begin{aligned}
\sum_{m=1}^{N} \mathbf{D}_{im}(\tilde{\mathcal{F}})_{mjk} &\approx \sum_{m=1}^{N} 2\mathbf{D}_{im} F^{\#}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}) \\
\sum_{m=1}^{N} \mathbf{D}_{jm}(\tilde{\mathcal{G}})_{imk} &\approx \sum_{m=1}^{N} 2\mathbf{D}_{jm} G^{\#}(\mathbf{Q}_{ijk}, \mathbf{Q}_{imk}) \\
\sum_{m=1}^{N} \mathbf{D}_{km}(\tilde{\mathcal{H}})_{ijm} &\approx \sum_{m=1}^{N} 2\mathbf{D}_{km} H^{\#}(\mathbf{Q}_{ijk}, \mathbf{Q}_{ijm})
\end{aligned}
\tag{3.10}
$$

Split formulations of the nonlinear fluxes use approximations of the derivatives as products to form new methods. The notation $(G)_x$ is used to denote the derivative of a quantity G with respect to the variable x. The derivative of a product to two quantities, $a(x)$ and $b(x)$, can be written as as split form as follows:

$$(a \cdot b)_x = \alpha (ab)_x + (1-\alpha)(a_x b + a b_x) \tag{3.11}$$

where $\alpha \in \mathbb{R}$. One particular choice is $\alpha = \frac{1}{2}$, which corresponds to quadratic splitting, gives the following:

$$(a \cdot b)_x = \frac{1}{2}\left[(ab)_x + (a_x b + a b_x)\right] \tag{3.12}$$

It should be noted that using quadratic splitting, it is possible to prove stability of linear variable coefficient problems [136]. For split forms involving cubic products, Kennedy and Gruber [131] proposed the following form:

$$\begin{aligned}(a \cdot b \cdot c)_x = \quad & \alpha(abc)_x + \\ & \beta\left[a(bc)_x + bc(a)_x\right] + \\ & \kappa\left[b(ac)_x + ac(b)_x\right] + \\ & \delta\left[c(ab)_x + ab(c)_x\right] + \\ & \epsilon\left[bc(a)_x + ac(b)_x + ab(c)_x\right]\end{aligned} \tag{3.13}$$

where $\epsilon = 1 - \alpha - \beta - \kappa - \delta$ and $\alpha, \beta, \kappa, \delta \in \mathbb{R}$. One particular choice of $\alpha = \beta = \kappa = \delta = \frac{1}{4}$ gives $\epsilon = 0$:

$$(a \cdot b \cdot c)_x = \frac{1}{4}(abc)_x + \frac{1}{4}\left[bc(a)_x + ac(b)_x + ab(c)_x\right] + \frac{1}{4}\left[a(bc)_x + b(ac)_x + c(ab)_x\right] \tag{3.14}$$

Eqns. (3.12) and (3.14) provide mechanisms to develop equivalent flux forms by splitting various products of variables in the fluxes of the governing equations.

3.2.1 Split Form Schemes

Split formulations are expressed via the advective fluxes of the governing equations, i.e. inviscid volume fluxes. Therefore, when utilizing the split form methodology, modification of only the inviscid volume terms is required. We first introduce the x-direction nonlinear flux of the inviscid equations for the standard DG discretization, denoted by superscript **SDG**, with the y- and z-directions constructed analogously as follows:

$$\vec{\mathcal{L}}_X^{\mathbf{SDG}}(\mathbf{Q}) = \begin{bmatrix} (\rho u)_x \\ (\rho u^2 + p)_x \\ (\rho uv)_x \\ (\rho uw)_x \\ (u(\rho e + p))_x \end{bmatrix} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(\rho e + p) \end{bmatrix}_x \tag{3.15}$$

The standard DG method gives the divergence form of the flux vector. However, the differentiation of each of the flux components can be rewritten using Eqns. (3.12) and (3.14) to construct new split formulations.

This work makes use of two split formulations, namely operators proposed by Kennedy & Gruber [131], and Pirozzoli [137]. Both schemes satisfy the definition of a kinetic energy (KE) preserving scheme, defined by Jameson [127] as follows: Ignoring boundary conditions, a discretization is KE preserving if the discrete integral of the kinetic energy is not changed by the advective terms of the governing equations, but only by the pressure work. Details of this definition will be established later in this chapter.

The Kennedy & Gruber split form scheme was the first to make use of both quadratic and cubic split forms [131]. The Kennedy & Gruber split form, denoted as **KG**, is defined as follows:

$$\tilde{\vec{\mathcal{L}}}_X^{\mathbf{KG}}(\mathbf{Q}) = \begin{bmatrix} \frac{1}{2}\left((\rho u)_x + \rho(u)_x + u(\rho)_x\right) \\ \frac{1}{4}\left((\rho u^2)_x + \rho(u^2)_x + 2u(\rho u)_x + u^2(\rho)_x + 2\rho u(u)_x\right) + p_x \\ \frac{1}{4}\left((\rho uv)_x + \rho(uv)_x + u(\rho v)_x + v(\rho u)_x + uv(\rho)_x + \rho v(u)_x + \rho u(v)_x\right) \\ \frac{1}{4}\left((\rho uw)_x + \rho(uw)_x + u(\rho w)_x + w(\rho u)_x + uw(\rho)_x + \rho w(u)_x + \rho u(w)_x\right) \\ \frac{1}{2}((\rho u)_x + p(u)_x + u(p)_x) \; + \; \frac{1}{4}((\rho eu)_x + \rho(eu)_x + e(\rho u)_x + u(\rho e)_x + eu(\rho)_x + \rho u(e)_x + \rho e(u)_x) \end{bmatrix} \tag{3.16}$$

The Pirozzoli [137] split form is similar to the Kennedy & Gruber split form but rewrites the energy equation by substituting ρuh for $(\rho e + p)u$, where the specific enthalpy is given by $h = e + \frac{p}{\rho}$. The Pirozzoli split form, denoted as **PZ**, is written as follows:

$$\tilde{\vec{\mathcal{L}}}_X^{\mathbf{PZ}}(\mathbf{Q}) = \begin{bmatrix} \frac{1}{2}\left((\rho u)_x + \rho(u)_x + u(\rho)_x\right) \\ \frac{1}{4}\left((\rho u^2)_x + \rho(u^2)_x + 2u(\rho u)_x + u^2(\rho)_x + 2\rho u(u)_x\right) + p_x \\ \frac{1}{4}\left((\rho uv)_x + \rho(uv)_x + u(\rho v)_x + v(\rho u)_x + uv(\rho)_x + \rho v(u)_x + \rho u(v)_x\right) \\ \frac{1}{4}\left((\rho uw)_x + \rho(uw)_x + u(\rho w)_x + w(\rho u)_x + uw(\rho)_x + \rho w(u)_x + \rho u(w)_x\right) \\ \frac{1}{4}\left((\rho uh)_x + \rho(uh)_x + h(\rho u)_x + u(\rho h)_x + uh(\rho)_x + \rho u(h)_x + \rho u(u)_x\right) \end{bmatrix} \tag{3.17}$$

Using the notation (dot annotation) introduced in Eqns. (3.12) and (3.14), the split formulations are as follows:

$$\tilde{\vec{\mathcal{L}}}_X^{\mathbf{KG}}(\mathbf{Q}) = \begin{bmatrix} \rho \cdot u \\ \rho \cdot u \cdot u + p \\ \rho \cdot u \cdot v \\ \rho \cdot u \cdot w \\ \rho \cdot u \cdot e + p \cdot u \end{bmatrix}_x, \qquad \tilde{\vec{\mathcal{L}}}_X^{\mathbf{PZ}}(\mathbf{Q}) \begin{bmatrix} \rho \cdot u \\ \rho \cdot u \cdot u + p \\ \rho \cdot u \cdot v \\ \rho \cdot u \cdot w \\ \rho \cdot u \cdot h \end{bmatrix}_x \tag{3.18}$$

Following the notation and derivation of Gassner et al. [135], the numerical split form sub-cell operators in Eqn. (3.10) can be expressed as two-point numerical fluxes using arithmetic means as follows:

$$F^{\#,1}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}) = \{\!\{\rho\}\!\}\{\!\{u\}\!\} := \frac{1}{2}\left(\rho_{ijk} + \rho_{mjk}\right) \cdot \frac{1}{2}\left(u_{ijk} + u_{mjk}\right) \tag{3.19}$$

where the notation $\{\!\{a\}\!\} = \frac{1}{2}\left(a_{ijk} + a_{mjk}\right)$ for some quantity a, with the indicies dictated by the flux direction. For example, the y-direction flux places index m in the second position, and the z-direction flux has the index m in the last position as follows:

$$\begin{aligned} G^{\#,1}(\mathbf{Q}_{ijk}, \mathbf{Q}_{imk}) &= \{\!\{\rho\}\!\}\{\!\{v\}\!\} := \frac{1}{2}\left(\rho_{ijk} + \rho_{imk}\right) \cdot \frac{1}{2}\left(v_{ijk} + v_{imk}\right) \\ H^{\#,1}(\mathbf{Q}_{ijk}, \mathbf{Q}_{ijm}) &= \{\!\{\rho\}\!\}\{\!\{w\}\!\} := \frac{1}{2}\left(\rho_{ijk} + \rho_{ijm}\right) \cdot \frac{1}{2}\left(w_{ijk} + w_{ijm}\right) \end{aligned}$$

The split form defined in Eqn. (3.16) can be formed using the two-point numerical volume flux to form the Kennedy & Gruber (**KG**) split form as follows:

$$F^{\#,\mathbf{KG}}\left(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}\right) = \begin{bmatrix} \{\!\{\rho\}\!\}\{\!\{u\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{u\}\!\} + \{\!\{p\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{v\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{w\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{e\}\!\} + \{\!\{p\}\!\}\{\!\{u\}\!\} \end{bmatrix} \tag{3.20}$$

From Eqn. (3.17), the Pirozzoli (**PZ**) split form is expressed as follows:

$$F^{\#,\mathbf{PZ}}\left(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}\right) = \begin{bmatrix} \{\!\{\rho\}\!\}\{\!\{u\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{u\}\!\} + \{\!\{p\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{v\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{w\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{h\}\!\} \end{bmatrix} \tag{3.21}$$

with the nonlinear fluxes in the y- and z-directions constructed similarly.

Numerical Flux Consistency

Lastly, the sub-cell differencing operators are connected to the cell interface numerical flux functions. Gassner et al. [138] showed that when the numerical flux split form scheme is also implemented into the numerical surface flux function, consistency of the volume and surface fluxes allow for the multi-element discretization to be kinetic energy preserving when the split form is kinetic energy preserving. We examine the general numerical flux as follows:

$$F^{*}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) := F^{\text{Symmetric}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) - F^{\text{Stab}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) \tag{3.22}$$

Using the Lax-Friedrichs numerical flux function as an example:

$$\begin{aligned} F^{\text{Symmetric}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) &= \frac{1}{2}\left(F\left(\mathbf{Q}_{-}\right) + F\left(\mathbf{Q}_{+}\right)\right) & (3.23)\\ F^{\text{Stab}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) &= \frac{1}{2}|\lambda|\left(\mathbf{Q}_{+} - \mathbf{Q}_{-}\right) & (3.24) \end{aligned}$$

where $|\lambda|$ is the max wave speed at the element interface. To be consistent with the split form volume flux, we replace the symmetric term with the split form as follows:

$$F^{*}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) = F^{\#}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) - F^{\text{Stab}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) \tag{3.25}$$

Note that for the scheme to be kinetic energy preserving, the stabilization term in Eqn. (3.25) is withheld. The stabilization term is added to the numerical flux function for added dissipation at the cell interfaces.

For the **KG** and **PZ** split forms, the stabilization term from the Lax-Friedrichs numerical flux function is chosen. Thus, the surface numerical flux for the **KG** split form scheme, for example, reads as follows:

$$F^{*,\mathbf{KG}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) = F^{\#,\mathbf{KG}}\left(\mathbf{Q}_{-},\mathbf{Q}_{+}\right) - \frac{1}{2}|\lambda|\left(\mathbf{Q}_{+} - \mathbf{Q}_{-}\right) \tag{3.26}$$

Split Form Complete Construction

For full expression of the split form DGSEM method, the Kennedy & Gruber method is demonstrated, as example, through the following. The governing equations in a mesh element at a LGL solution point (i, j, k) are written as follows:

$$\left(J\frac{\partial\tilde{\mathbf{Q}}}{\partial t}\right)_{i,j,k} + \left(\tilde{\vec{\mathcal{L}}}_X(\mathbf{Q})\right)_{i,j,k} + \left(\tilde{\vec{\mathcal{L}}}_Y(\mathbf{Q})\right)_{i,j,k} + \left(\tilde{\vec{\mathcal{L}}}_Z(\mathbf{Q})\right)_{i,j,k} = 0$$

The nonlinear flux functions are constructed as follows:

$$\begin{aligned}
\left(\tilde{\vec{\mathcal{L}}}_X(\mathbf{Q})\right)_{i,j,k} &\approx \frac{1}{\omega_i}\left(\delta_{iN}\left[\tilde{\mathcal{F}}^{*,\mathbf{KG}} - \tilde{\mathcal{F}}\right]_{Njk} - \delta_{i1}\left[\tilde{\mathcal{F}}^{*,\mathbf{KG}} - \tilde{\mathcal{F}}\right]_{1jk}\right) + \sum_{m=1}^{N} 2\mathbf{D}_{im}F^{\#,\mathbf{KG}}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}) \\
\left(\tilde{\vec{\mathcal{L}}}_Y(\mathbf{Q})\right)_{i,j,k} &\approx \frac{1}{\omega_j}\left(\delta_{jN}\left[\tilde{\mathcal{G}}^{*,\mathbf{KG}} - \tilde{\mathcal{G}}\right]_{iNk} - \delta_{j1}\left[\tilde{\mathcal{G}}^{*,\mathbf{KG}} - \tilde{\mathcal{G}}\right]_{i1k}\right) + \sum_{m=1}^{N} 2\mathbf{D}_{jm}G^{\#,\mathbf{KG}}(\mathbf{Q}_{ijk}, \mathbf{Q}_{imk}) \\
\left(\tilde{\vec{\mathcal{L}}}_Z(\mathbf{Q})\right)_{i,j,k} &\approx \frac{1}{\omega_k}\left(\delta_{kN}\left[\tilde{\mathcal{H}}^{*,\mathbf{KG}} - \tilde{\mathcal{H}}\right]_{ijN} - \delta_{k1}\left[\tilde{\mathcal{H}}^{*,\mathbf{KG}} - \tilde{\mathcal{H}}\right]_{ij1}\right) + \sum_{m=1}^{N} 2\mathbf{D}_{km}H^{\#,\mathbf{KG}}(\mathbf{Q}_{ijk}, \mathbf{Q}_{ijm})
\end{aligned}$$

where the matrix $\mathbf{D}$ defined by Eqn. (3.5), and the numerical flux functions, $\mathcal{F}^{*,\mathbf{KG}}$, $\mathcal{G}^{*,\mathbf{KG}}$, $\mathcal{G}^{*,\mathbf{KG}}$ are constructed consistently with the split form method, as shown in Eqn. (3.26).
The sub-cell differencing flux functions $F^{\#,\mathbf{KG}}$, $G^{\#,\mathbf{KG}}$, $H^{\#,\mathbf{KG}}$ are written as:

$$F^{\#,\mathbf{KG}}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}) = \begin{bmatrix} \{\!\{\rho\}\!\}\{\!\{u\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{u\}\!\} + \{\!\{p\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{v\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{w\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{u\}\!\}\{\!\{e\}\!\} + \{\!\{p\}\!\}\{\!\{u\}\!\} \end{bmatrix}$$

$$G^{\#,\mathbf{KG}}(\mathbf{Q}_{ijk}, \mathbf{Q}_{imk}) = \begin{bmatrix} \{\!\{\rho\}\!\}\{\!\{v\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{v\}\!\}\{\!\{u\}\!\} + \{\!\{p\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{v\}\!\}\{\!\{v\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{v\}\!\}\{\!\{w\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{v\}\!\}\{\!\{e\}\!\} + \{\!\{p\}\!\}\{\!\{v\}\!\} \end{bmatrix}$$

$$H^{\#,\mathbf{KG}}(\mathbf{Q}_{ijk}, \mathbf{Q}_{ijm}) = \begin{bmatrix} \{\!\{\rho\}\!\}\{\!\{w\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{w\}\!\}\{\!\{u\}\!\} + \{\!\{p\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{w\}\!\}\{\!\{v\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{w\}\!\}\{\!\{w\}\!\} \\ \{\!\{\rho\}\!\}\{\!\{w\}\!\}\{\!\{e\}\!\} + \{\!\{p\}\!\}\{\!\{w\}\!\} \end{bmatrix}$$

where the arithmetic averaging, e.g. first component of each split flux, is formed as follows:

$$\begin{aligned}
F^{\#,1}(\mathbf{Q}_{ijk}, \mathbf{Q}_{mjk}) &= \{\!\{\rho\}\!\}\{\!\{u\}\!\} = \frac{1}{2}(\rho_{ijk} + \rho_{mjk}) \cdot \frac{1}{2}(u_{ijk} + u_{mjk}) \\
G^{\#,1}(\mathbf{Q}_{ijk}, \mathbf{Q}_{imk}) &= \{\!\{\rho\}\!\}\{\!\{v\}\!\} = \frac{1}{2}(\rho_{ijk} + \rho_{imk}) \cdot \frac{1}{2}(v_{ijk} + v_{imk}) \\
H^{\#,1}(\mathbf{Q}_{ijk}, \mathbf{Q}_{ijm}) &= \{\!\{\rho\}\!\}\{\!\{w\}\!\} = \frac{1}{2}(\rho_{ijk} + \rho_{ijm}) \cdot \frac{1}{2}(w_{ijk} + w_{ijm})
\end{aligned}$$

The remaining arithmetic flux components are expressed analogously.

3.2.2 Kinetic Energy Preserving Discretizations

The definition of a kinetic energy preserving discretization as defined by Jameson [127] is as follows: Ignoring boundary conditions, the disretization is kinetic energy preserving if the discrete integral of kinetic energy is not changed by the advective terms, but only by the pressure work. The kinetic energy is written as follows:

$$\kappa := \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) \tag{3.27}$$

The kinetic energy balance derived from the inviscid Euler equations is written as follows:

$$\frac{\partial \kappa}{\partial t} + \underbrace{\left(\tfrac{1}{2}\rho u(u^2+v^2+w^2)\right)_x + \left(\tfrac{1}{2}\rho v(u^2+v^2+w^2)\right)_y + \left(\tfrac{1}{2}\rho w(u^2+v^2+w^2)\right)_z}_{\text{advective divergence}} + \underbrace{up_x + vp_y + wp_z}_{\text{pressure work}} = 0 \tag{3.28}$$

Notice that the advective terms can be expressed in conservative form whereas the pressure work (pressure gradients) results in a non-conservative form. For finite-volume schemes, Jameson [127] found that a discretization is kinetic energy preserving if the surface flux components can be expressed as follows:

$$\begin{aligned}
&F^{*,2} = F^{*,1}\{\!\{u\}\!\} + \tilde{p}, && F^{*,3} = F^{*,1}\{\!\{v\}\!\}, && F^{*,4} = F^{*,1}\{\!\{w\}\!\}, \\
&G^{*,2} = G^{*,1}\{\!\{u\}\!\}, && G^{*,3} = G^{*,1}\{\!\{v\}\!\} + \tilde{p}, && G^{*,4} = G^{*,1}\{\!\{w\}\!\}, \\
&H^{*,2} = H^{*,1}\{\!\{u\}\!\}, && H^{*,3} = H^{*,1}\{\!\{v\}\!\}, && H^{*,4} = H^{*,1}\{\!\{w\}\!\} + \tilde{p}
\end{aligned} \tag{3.29}$$

where $F^{*,1}$ represents the first component of the x-direction flux vector, with G^* and H^* representing the y- and z-direction flux vectors. The pressure component $\tilde{p}$ as noted by Jameson can be any consistent approximation; Gassner et al. [135] demonstrated that the best results for discrete kinetic energy preservation are achieved when the arithmetic mean is used for $\tilde{p}$.

Observe Eqns. (3.20) and (3.21) hold for Eqn. (3.29). Gassner et al. [138] showed that if the numerical volume and surface flux functions $F^{\#}, G^{\#}, H^{\#}$ satisfy the kinetic energy preserving condition given in Eqn. (3.29), then the multi-element discretization is kinetic energy preserving. When a discretization is kinetic energy preserving, the discrete total kinetic energy is not dissipated by the advective terms. Demonstration of this property will be shown in the next section.

3.3 Numerical Experiments

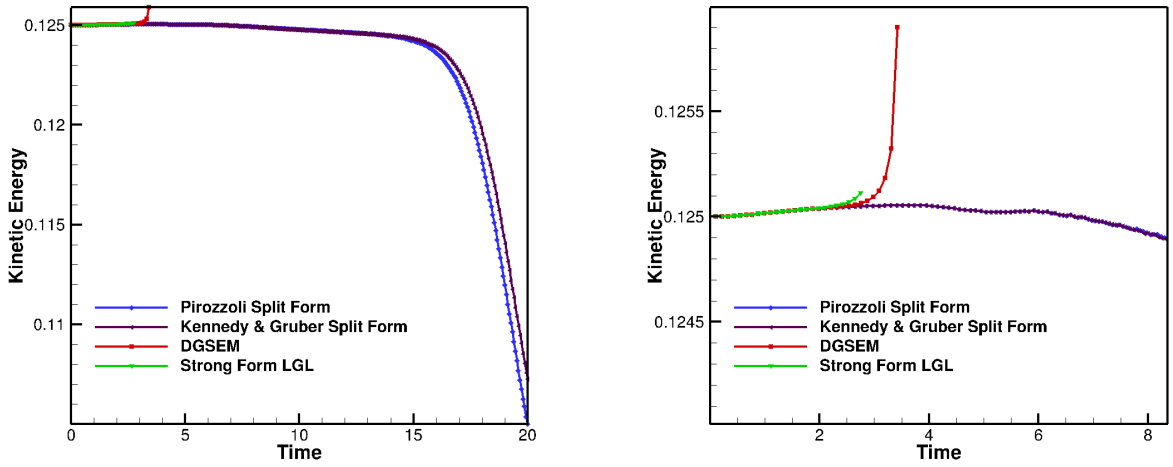
Investigations of the Pirozzoli and Kennedy & Gruber split formulations are performed in this section demonstrating their kinetic energy preservation and their superior stability compared to the standard DG method expressed in Chapter 2. The test problem we examine herein is the inviscid and viscous Taylor-Green Vortex problem. All simulations were obtained using the SSP-TVD three-stage, third-order Runge-Kutta explicit time stepping scheme with a CFL of 0.5. The ratio of specific heats is chosen to be $\gamma = 1.4$.

3.3.1 Inviscid Taylor-Green Vortex

To test the kinetic energy preservation of the DG method with the Pirozzoli and Kennedy & Gruber split forms, we first investigate the inviscid Taylor-Green Vortex problem. The initial conditions over a periodic cubic domain of $[-2\pi, 2\pi]^3$ are defined as follows:

$$\begin{aligned} \rho &= \rho_0 \\ u &= V_0 \sin(x)\cos(y)\cos(z) \\ v &= -V_0 \cos(x)\sin(y)\cos(x) \\ w &= 0 \\ p &= p_0 + \frac{\rho_0 V_0^2}{16}\left(\cos(2x)\cos(2z) + 2\cos(2y) + 2\cos(2x) + \cos(2y)\cos(2z)\right) \end{aligned} \tag{3.30}$$

where initial reference velocity $V_0 = 0.1$, initial density $\rho_0 = 1$, and initial pressure $p_0 = \frac{100}{\gamma}$. This problem is quite challenging in the absence of physical viscosity, there is no dissipative mechanism to dampen the turbulent scales thus the problem is always under-resolved for long enough simulation time. The Taylor-Green Vortex solution conserves total kinetic energy and total entropy for all time. This is due to the fully periodic boundary conditions which allows for the pressure work to cancel when integrated over the entire domain which prevents any change of the total kinetic energy.



(a) Kinetic energy evolution for 64^3 DOF.

(b) Solver failure for non-split-form methods.

Figure 3.1: Kinetic energy evolution for the inviscid Taylor-Green Vortex for DG discretizations without flux stabilization with 16, $p = 3$ elements in each spatial direction.

Kinetic Energy Preservation

For kinetic energy preservation, the element interface stabilization must be omitted from the surface flux function. The stabilization term in the surface flux function, as shown in Eqn. (3.25) introduces numerical dissipation which would dissipate kinetic energy, thus it is omitted for these tests. We note that the symmetric term, also shown in Eqn. (3.25), (which is consistent with the split form) is used in the surface flux calculation which allows for inter-cell interaction. A polynomial degree of $p = 3$ with 16^3 mesh elements giving a total of 64^3 degrees-of-freedom is used for this case (to resolve the scales in this problem, approximately 512^3 degrees-of-freedom are required).

Fig. (3.1) demonstrates the kinetic energy evolution four DG discretizations: Kennedy & Gruber split form, Pirozzoli split form, strong form DG method using LGL collocated solution and quadrature points, and the standard DGSEM method using collocated GL points. As seen in the figure, the split form methods are able to conserve kinetic energy for a majority of the simulation. However, as seen in the figure, the kinetic energy decreases due to the inexact integration via numerical integration. These numerical inaccuracies eventually change the total kinetic energy.

Further, we see that the non-split form DG methods without the flux stabilization term become unstable and fail early in the simulation. The Pirozzoli and Kennedy & Gruber split-form DG discretizations demonstrate their superior stability and are able to complete the full simulation even without the flux stabilization term.

Stability

As seen from the previous subsection, the split form methods can solve severely under-resolved simulations without the use of a flux stabilization dissipation term. Next, we enable the flux stabilization term for all DG methods. Fig. (3.2) demonstrates enstrophy evolution of the inviscid Taylor-Green Vortex. Enstrophy is mathematically defined as follows:

$$\mathcal{E} = \frac{1}{\rho_0 \Omega} \int_{\Omega} \rho \frac{\omega \cdot \omega}{2} d\Omega \tag{3.31}$$

where ρ and ρ_0 are the density and initial density, respectively, ω is the vorticity, and Ω is the cell volume. As seen in the figure, again the non-split form methods fail numerically in comparison to both split form discretizations. Further, note that the split form methods *without* the flux stabilization term are more numerically stable than the standard DGSEM method *with* the flux stabilization term for this case. Additionally, we see the standard DGSEM with the GL collocation points is more stable in comparison to the strong form method using the LGL collocation points.

Fig. (3.3) demonstrates the split form methods with and without flux stabilization. Without the flux stabilization, no formal dissipation is present in the discretizations, therefore allowing the total vorticity density of the flow to dramatically rise in comparison to the methods with the stabilization term present. We see that the schemes are very similar with the exception of the non-stabilized Kennedy & Gruber method maintaining a slightly higher enstrophy near the end of the simulation compared to the non-stabilized Pirozzoli method.

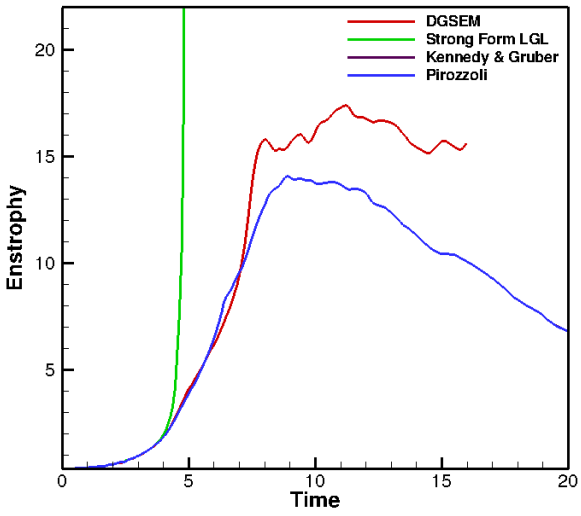


Figure 3.2: Enstrophy evolution for the inviscid Taylor-Green Vortex for different DG discretizations with flux stabilization. Each discretization used 16, $p = 3$ elements in each spatial direction.

3.3.2 Viscous Taylor-Green Vortex

The second test case used is the viscous Taylor-Green Vortex with same initial conditions as the inviscid case, but with the initial pressure changed to $p_0 = \frac{1}{\gamma}$ and the Reynolds number $Re = 1,600$. Fig. (3.4) shows the kinetic energy dissipation rate for the four discretizations. A reference solution is provided by a pseudo-spectral method [125] contextualizing the accuracy of the various DG methods. As seen from the figure, the strong form LGL DG method becomes numerically unstable early in the simulation and fails. The standard DGSEM and split form methods successfully complete the simulation. The split form results are nearly identical to each other but differ from the DGSEM discretization. Recall, DGSEM uses the Gauss-Legendre points for both solution and numerical integration in comparison to the Lobatto-Gauss-Legendre points. The GL points are more accurate for numerical integration compared to the LGL points highlighting the reason why the DGSEM results are closer to the reference solution. However, as seen previously, the split form methods provide superior stability in comparison to other DG discretizations.

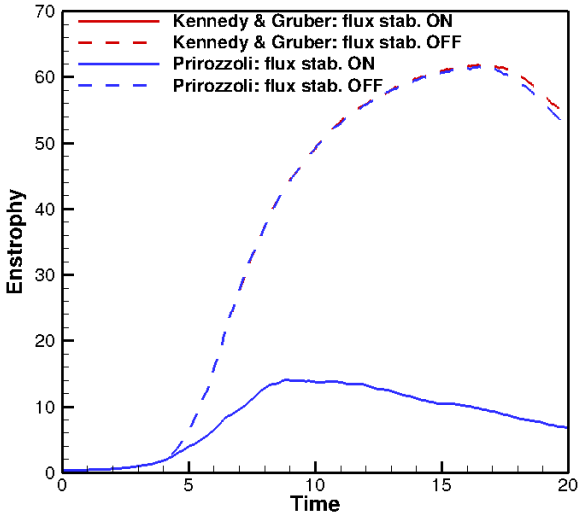


Figure 3.3: Enstrophy evolution comparison for the inviscid Taylor-Green Vortex for the split form DG discrizations with and without flux stabilization. Each discretization used 16, $p = 3$ elements in each spatial direction.

3.4 Summary

Split form discontinuous Galerkin methods with the summation-by-property show great promise for stability, and construction of numerical discretizations with favorable properties such as kinetic energy or entropy stability. The computational cost of the split-form methods is slightly more in comparison to the standard collocated discontinuous Galerkin method. However, the significant increase in stability justifies the additional, minimal, cost of the split form SBP DG method. Additionally, no explicit filtering is required for numerical stability for the SBP DG method, contrary to the standard DG method, which further increases the computational cost.

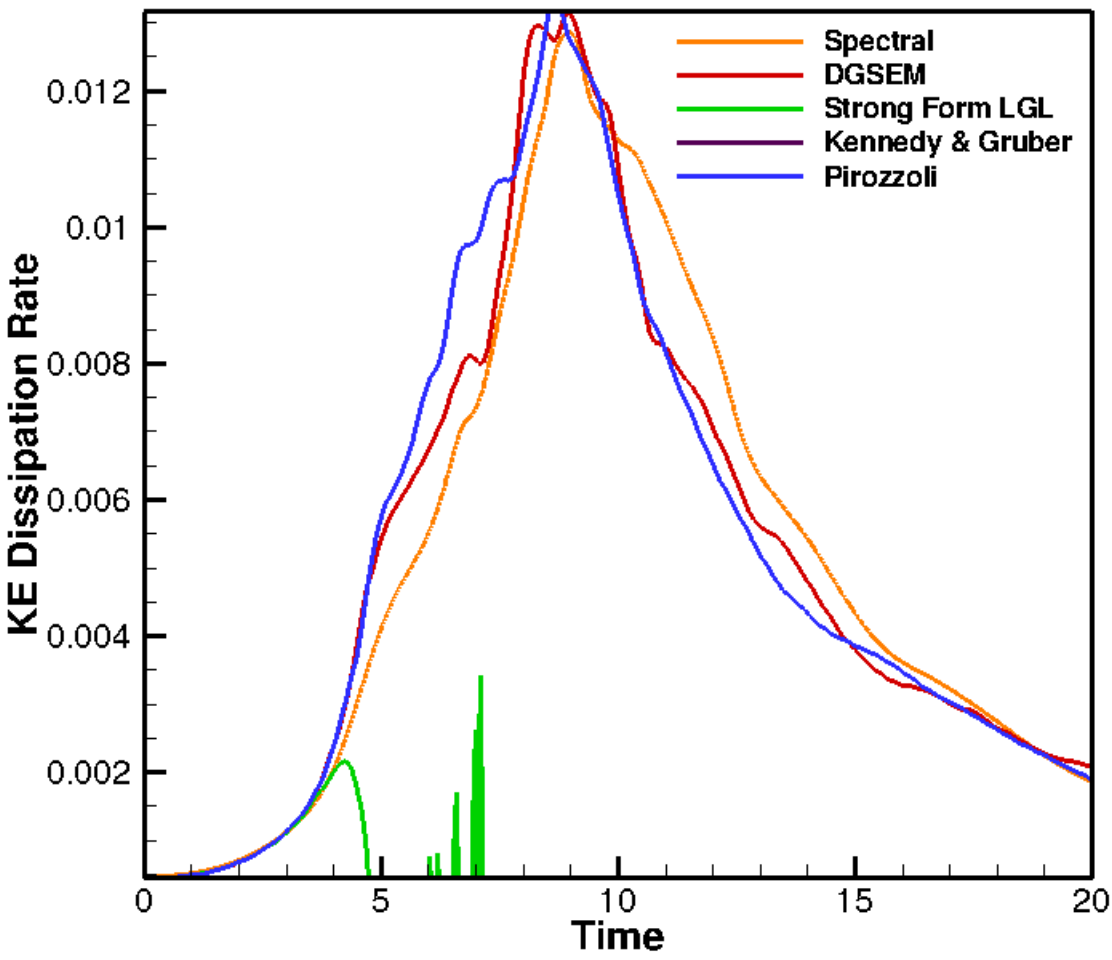


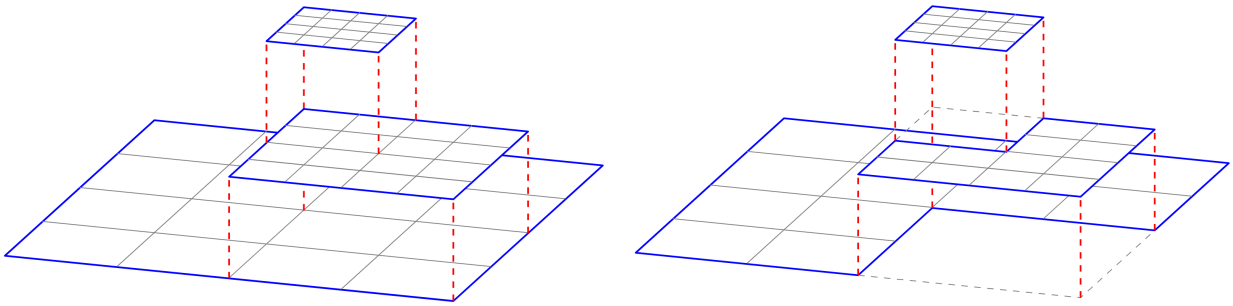
Figure 3.4: Kinetic energy dissipation rate evolution for the viscous Taylor-Green Vortex for the split form, DGSEM, and strong form DG methods. Each discretization used 16, $p = 3$ elements in each spatial direction.

Chapter 4

Adaptive Mesh and Solution Refinement Methods

To enable extreme scale simulations for applications in wind energy, without the use of excessive computational resources, adaptive mesh refinement (AMR) methods allow for increased resolution to be focused in areas of interest, e.g. regions of high vorticity found in wake regions. The goal of this work, practically, is to accurately simulate multiscale problems that contain unsteady flow features, such as wakes from wind turbines and rotary-wing vehicles, using high-order numerical methods. Utilization of uniformly refined grids for multiscale flow problems, e.g. 10 km^3 wind farm simulation region requiring localized small-scale spatial resolutions on-the-order-of microns, is computationally intractable for modern-day supercomputers. In contrast, adaptive mesh refinement allows for the initial calculation to start with a very coarse computational mesh, using large mesh elements, to identify regions of interest, and to place increased spatial resolution in those respective regions.

Adaptive mesh refinement methods can be classified into two general mesh refinement techniques: r-refinement and h-refinement. The former, r-refinement is a technique which modifies the mesh without changing the number of nodes in the mesh. The increased mesh resolution is achieved through moving grid points into areas of interest, which results in clustering of grid points. The latter adaptive mesh refinement technique, h-refinement is a modification of the mesh resolution through the addition or subtraction of grid points.



(a) patch-based AMR (b) octree-based AMR

Figure 4.1: Patch-based and octree-based adaptive mesh refinement grid level structures. Patch-based AMR methods have cells overlaid in contrast to octree-based AMR methods. Images courtesy of Carsten Burstedde.

The simplest strategy of h-refinement is to subdivide a mesh element into multiple sub-elements. For example in three-dimensional space, one hexahedron can be subdivided into eight hexahedra. This approach of element subdivision can lead to hanging nodes, which are grid points that are positioned on an interface between mesh elements, but do not properly belong to all elements in contact as seen in Fig. (4.3).

In finite-element methods, a third type of refinement is possible known as p-refinement or p-enrichment. This refinement technique is not concerned with mesh refinement but rather with increasing the solution accuracy inside a mesh element. This is achieved via increasing the polynomial order locally. This technique has been shown to give exponential convergence for sufficiently smooth solutions, thus giving more accurate solutions in comparison to h-refinement using fixed degree polynomials [52]. The use of both h-refinement and p-enrichment leads to hp refinement finite-element methods.

To best achieve adaptive mesh refinement dynamically in time, structured adaptive refinement methods are utilized. Structured adaptive mesh refinement methods can adapt dynamically in a time-dependent manner to resolve unsteady effects with relative ease. In a parallel compute setting, for every mesh adaption step, the grid needs to be re-partitioned for load balancing, and parallel communications must be redirected. In a structured mesh setting, work-load repartitioning efficiency is much greater in comparison to adaptive unstructured mesh environments. Structured adaptive mesh refinement techniques are most common through block-structured AMR and tree-based AMR.

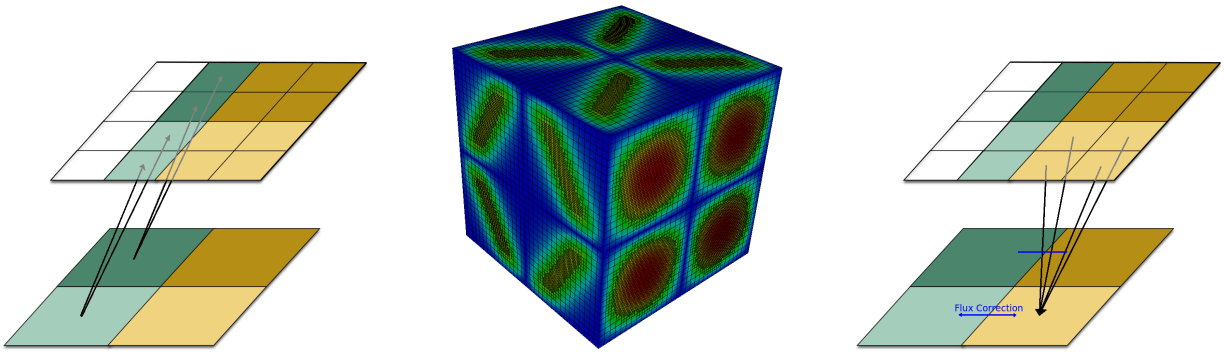
Block-structured AMR first appeared in 1984 following Berger and Colella [139]. Many block-structured AMR software frameworks have been developed, e.g. BoxLib [140], Chombo [30], PARAMESH [141], SAMRAI [142], AMReX [143]. In block-structured AMR, the computational mesh is decomposed into a collection of logically rectangular overlapping patches that form mesh levels, each of fixed spatial resolution, as shown in Fig. (4)(a). The spatial resolution ratio between two consecutive mesh levels, known as the refinement ratio, is generally two, although other refinement ratios are possible. The computational mesh structure starts at the coarsest level, which tessellates the entire computational domain using the coarsest spatial resolution prescribed by the user. Finer mesh levels are constructed sequentially, from coarsest to finest, by tagging cells for refinement, clustering them into patches, and refining the solution. These newly refined patches overlap coarser level patches, which require their solutions to be filled using the more accurate, fine level, solution. This requires the development of a coarsening operator, which transfers the fine level solution to the coarse level solution.

In contrast to block-structured AMR methods where blocks are permitted to overlap, tree-based AMR methods make use of recursive encoding algorithms for non-overlapping mesh refinement, as shown in Fig. (4)(b). Thus, there is no need to fill covered coarse level solutions, since tree-based AMR methods do not contain overlapping mesh elements. This work utilizes an *hp*-refinement strategy with hanging nodes in a forest-of-octrees AMR framework known as `p4est` [144].

4.0.1 Patch-Based and Octree-Based Communication Protocols

In the author's experience, the use of the octree-based AMR system allows for greater ease of implementation for finite-element methods compared to the patch-based AMR system due to traditional communication implementations for each of the systems. Within this work, algorithm development has been performed in both AMR systems: SAMRAI [142] for patch-based AMR, and `p4est` [144, 145] for tree-based AMR.

SAMRAI and `p4est` both implement *h*-refinement, however, neither framework provides direct support for *p*-enrichment, which requires variable storage in individual element data



(a) coarse-to-fine level communication via refinement operator

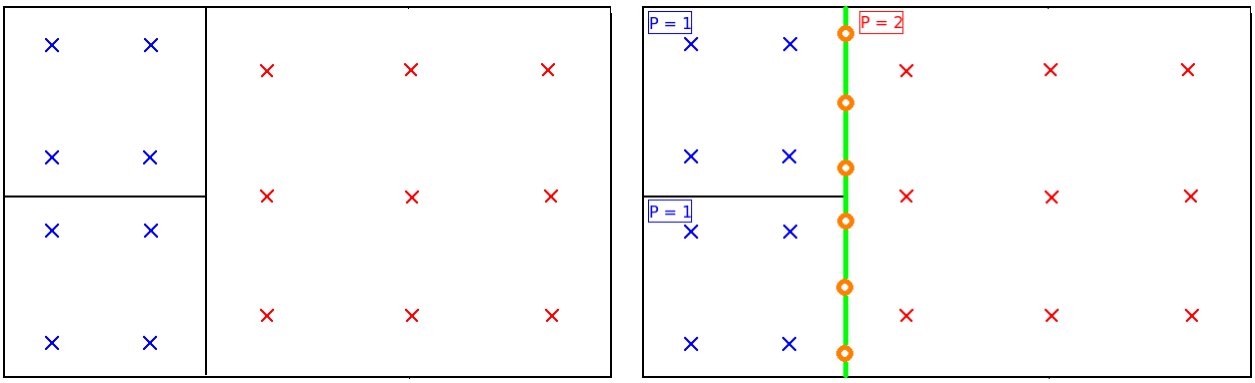
(b) solver time step execution

(c) fine-to-coarse level communication, and coarse level flux correction

Figure 4.2: Patch-based adaptive mesh refinement communication and solve procedure per computational time step.

structures. However, `p4est` implements a communication pattern similar to the discontinuous Galerkin finite-element method, meaning the communication pattern is composed of a nearest-neighbor stencil (`p4est` implements other extended communication patterns as well). When parallel communication is invoked, the nearest-neighbor elements on a mesh-partition boundary are communicated to the respective neighboring processor. This can be viewed as an exchange of cell solutions at the mesh-partition boundary.

In contrast, the patch-based AMR framework SAMRAI implements parallel communication patterns which are composed of data transfers between mesh level interfaces. These data communication transfers first occur in a coarse-to-fine level procedure, which requires a refinement operator. In the solution process, the flow solver is executed for one computational time step on each mesh level, then the solution on the fine level is transferred down to the coarser levels via a coarsen operator. Fig. (4.2) demonstrates the patch-based AMR communication and flow solve execution pattern. Additionally, a flux correction is required for conservation as the flux calculated on the fine level is different from the flux calculated on the coarse level. Thus, addition communication of the flux calculated on the fine level to the coarse level is required, which is used to correct the coarse level solution. Further, the coarsen operator adds extraneous computations at every computational time step. Many of the patch-based AMR frameworks are implemented using this communication protocol,



(a) hanging element face (b) mortar element interface

Figure 4.3: Hanging mesh elements requiring a mortar element for flux calculation.

which can be difficult for finite-element method implementation. If the parallel communication pattern of direct solution is used, patch-based AMR would be perfectly acceptable for use and ease of development for finite-element methods.

4.1 Numerical Operators for AMR

The use of adaptive mesh refinement requires special numerical operations which are used to communicate solutions between mesh levels and at fine-coarse element boundary interfaces. The two operators required are a refinement operator and a coarsen operator. Two occurrences require the use of a refinement operator: coarse-fine element boundary interpolation, and refinement of coarse cells for newly formed refined cells. The former occurs at cell interfaces with hanging element faces where flux calculations are required. The coarsening operator is utilized when a collection of element solutions does not meet a feature-based refinement criterion, which are then used to form a single coarse element solution.

4.1.1 Mortar Element Strategy

For coarse-fine element boundary interpolation, a common element interface, called a mortar element, is required. As shown in Fig. (4.3), a mortar element is a common face onto which solutions are interpolated, and flux calculations are performed. The number of points on the mortar element is chosen to be $2 \cdot (p_{\max} + 1)$, where $p_{\max} = \max(p_{\text{left}}, p_{\text{right}})$ with p_{left}

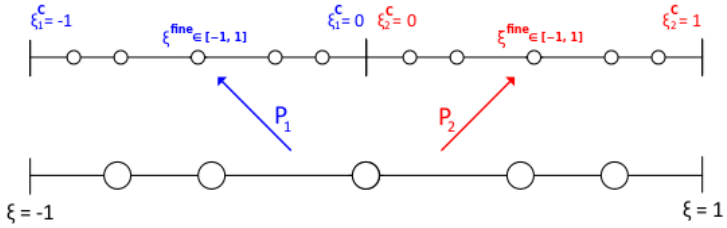


Figure 4.4: One-dimensional refine operator via Galerkin projection.

and p_{right} being the left and right element solution polynomial degrees. For example, in the figure demonstrating the mortar element, the left polynomial degrees are $p = 1$, and the right polynomial degree is $p = 2$. Thus, $p_{\max} = 2$ gives the number of flux points on the mortar face as $2 \cdot (2 + 1) = 6$.

4.1.2 Refinement Operators

As shown in Fig. (4.4), the coarse element coordinates are represented by $\xi_{\text{coarse}} \in [-1, 1]$, and the coarse element solution is represented as Q_j. The fine elements bordering the coarse element each have local reference coordinates $\xi_{\text{fine}} \in [-1, 1]$, which are mapped to the coarse element coordinate systems $\xi_1^C \in [-1, 0]$ and $\xi_2^C \in [0, 1]$. To obtain the refined left solution, U^L, a coordinate transformation $\xi_{\text{fine}} \mapsto \xi_1^C : \ \xi_1^C = \frac{1}{2}(\xi_{\text{fine}} - 1)$ is performed, and evaluated using the coarse element basis functions ϕ_j, giving the projection P_1 operator as follows:

$$U^L(\xi_{\text{fine}}) = \sum_{j=1}^{N} \phi_j \left(\frac{1}{2}(\xi_{\text{fine}} - 1) \right) Q_j \tag{4.1}$$

Similarly, to obtain the refined right solution U^R, a coordinate transformation $\xi_{\text{fine}} \mapsto \xi_2^C$: $\xi_2^C = \frac{1}{2}(\xi_{\text{fine}} + 1)$ is performed giving the projection P_2 operator as follows:

$$U^R(\xi_{\text{fine}}) = \sum_{j=1}^{N} \phi_j \left(\frac{1}{2}(\xi_{\text{fine}} + 1) \right) Q_j \tag{4.2}$$

In three dimensions, transformation of each coordinate direction is performed to the respec-

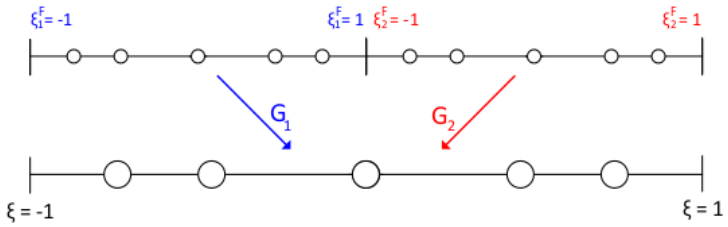


Figure 4.5: One-dimensional coarsen operator via mass matrix Galerkin projection.

tive coarse subset coordinates, ξ_1^C or ξ_2^C, then a Galerkin projection is performed using the coarse element basis functions evaluated at the transformed coordinates.

4.1.3 Coarsen Operators

Conversely to the refinement operator, an interpolation mechanism is needed to transfer data from a set of fine element solutions to one coarse element solution. To maintain discrete conservation of the conservative variables, the employment of a mass matrix Galerkin projection approach is used for the coarsening operator. To form a coarse element solution Q_j from two fine element solutions, as shown in Fig. (4.5), a mass matrix Galerkin projection must satisfy the following:

$$\sum_{j=1}^{N} \phi_j(\xi) Q_j = U^{\text{fine}}(\xi), \quad \xi \in [-1, 1] \tag{4.3}$$

where $U^{\text{fine}}(\xi)$ is the composite vector of the fine element solutions $\left[U^L(\xi), U^R(\xi)\right]^T$, such that:

$$U^{\text{fine}}(\xi) = \begin{cases} U^L(\xi), & \xi \in [-1, 0] \\ U^R(\xi), & \xi \in [0, +1] \end{cases} \tag{4.4}$$

Further, we note the following definitions of the fine element solutions using the coarse element basis functions, ϕ_j:

$$U^L(\xi) = \sum_{j=1}^{N} \phi_j(2\xi + 1)Q_j^L, \quad \xi \in [-1, 0] \tag{4.5}$$

$$U^R(\xi) = \sum_{j=1}^{N} \phi_j(2\xi - 1)Q_j^R, \quad \xi \in [0, +1] \tag{4.6}$$

where Q^L and Q^R are the solution coefficients for the fine level solutions U^L and U^R, respectively. For nodal basis functions, the coefficients are the same as the solutions, thus, $Q_i^L = U^L(\xi_i)$, and $Q_i^R = U^R(\xi_i)$. Next, we define the mass matrix, $[\mathbf{M}]$, and mixed basis matrices,$[\mathbf{C}_1]$ and $[\mathbf{C}_2]$, as follows:

$$[\mathbf{M}] = M_{ij} = \int_{-1}^{1} \phi_i(\xi)\phi_j(\xi)\mathrm{d}\xi, \quad 1 \leq i, j \leq N \tag{4.7}$$

$$[\mathbf{C}_1] = C_{ij}^1 = \int_{-1}^{1} \phi_i(\xi)\phi_j(2\xi + 1)\mathrm{d}\xi, \quad 1 \leq i, j \leq N \tag{4.8}$$

$$[\mathbf{C}_2] = C_{ij}^2 = \int_{-1}^{1} \phi_i(\xi)\phi_j(2\xi - 1)\mathrm{d}\xi, \quad 1 \leq i, j \leq N \tag{4.9}$$

Next, multiplying both sides of Eqn. (4.3) by ϕ_i, and integrating over the standard element:

$$\begin{aligned}
\int_{-1}^{1}\left(\sum_{j=1}^{N}\phi_i\left(\xi\right)\phi_j\left(\xi\right)Q_j\right)\mathrm{d}\xi &= \int_{-1}^{1}\phi_i\left(\xi\right)U^{\text{fine}}(\xi)\mathrm{d}\xi \\
\int_{-1}^{1}\left(\sum_{j=1}^{N}\phi_i\left(\xi\right)\phi_j\left(\xi\right)Q_j\right)\mathrm{d}\xi &= \int_{-1}^{0}\phi_i\left(\xi\right)U^{L}(\xi)\mathrm{d}\xi + \int_{0}^{1}\phi_i\left(\xi\right)U^{R}(\xi)\mathrm{d}\xi \quad \boxed{\text{by: Eqn. (4.4)}} \\
\sum_{j=1}^{N}\left(\int_{-1}^{1}\phi_i\left(\xi\right)\phi_j\left(\xi\right)\mathrm{d}\xi\right)Q_j &= \sum_{j=1}^{N}\left(\int_{-1}^{0}\phi_i\left(\xi\right)\phi_j(2\xi+1)\mathrm{d}\xi\right)Q_j^L \\
&+ \sum_{j=1}^{N}\left(\int_{0}^{1}\phi_i\left(\xi\right)\phi_j(2\xi-1)\mathrm{d}\xi\right)Q_j^R \\
&\quad \boxed{\text{by: Fubini's Theorem [146], Eqns. (4.5) and (4.6)}} \\
\sum_{j=1}^{N}\left(M_{ij}\right)Q_j &= \sum_{j=1}^{N}\left(C_{ij}^{1}\right)Q_j^L \\
&+ \sum_{j=1}^{N}\left(C_{ij}^{2}\right)Q_j^R \qquad \boxed{\text{by: Eqns. (4.7), (4.8), (4.9)}} \\
\left[\mathbf{M}\right]\vec{Q} &= \left[\mathbf{C}_1\right]\vec{Q}^L + \left[\mathbf{C}_2\right]\vec{Q}^R \\
\vec{Q} &= \left[\mathbf{M}\right]^{-1}\left(\left[\mathbf{C}_1\right]\vec{Q}^L + \left[\mathbf{C}_2\right]\vec{Q}^R\right) \qquad (4.10)
\end{aligned}$$

A composite matrix, $[\mathbf{G}]$, can be formed as follows:

$$\left[\mathbf{G}\right] = \left[\mathbf{M}\right]^{-1}\left[\left[\mathbf{C}_1\right]\left[\mathbf{C}_2\right]\right] \qquad (4.11)$$

Finally, the coarse element solution, $\vec{Q}$, is obtained from the fine element solutions, $\vec{U}^L$ and $\vec{U}^R$, as follows:

$$\vec{Q} = \left[\mathbf{G}\right]\begin{bmatrix}\vec{U}^L \\ \vec{U}^R\end{bmatrix} \qquad (4.12)$$

4.2 Adaptive Mesh Refinement Results

This section presents results using adaptive mesh refinement for two problems. First, a mesh refinement study is performed to demonstrate the correct order-of-accuracy asymptotic error rates under the presence of hanging faces. Second, a study of the Taylor Green Vortex problem is investigated to show the dynamic adaption, and ability to capture flow features accurately while reducing the computational cost.

4.2.1 Ringleb Flow Mesh Resolution Study

As previously mentioned in Chapter 2, Ringleb flow is an analytic solution of the two-dimensional inviscid Euler equations. This can be used to determine if the numerical discretization is obtaining correct asymptotic error rates as a mesh is refined in element size. Fig. (4.6)(a) demonstrates the density contours of the analytic flow using a coarse base mesh with a section of refinement in the interior of the computational domain.

To verify the asymptotic error rates of the numerical discretization associated with the adaptive mesh refinement operators, two studies are performed: a global mesh convergence study using a single polynomial degree with two levels of mesh refinement, and a global mesh convergence study using two polynomial degrees with two levels of mesh refinement. In the latter case, the polynomial degrees are chosen to be in sequence, i.e. p-degree and $(p+1)$-degree. The L_2-error is expected to have slope $Ch^{(p+1)}$ as the mesh is refined. The solution error slopes for $p = 1 - 5$ are shown in Table (4.1). Note that for $p = 5$, the final converged error on the finest mesh is at machine precision, therefore, the error cannot reduce at the expected asymptotic rate as observed on the coarser meshes. Verification of the hanging face mortar element operators of the same solution order are demonstrated in Figure 4.7(a) for polynomial degrees $p = 1 - 5$.

The second mesh resolution study uses two polynomial degrees, namely p and $p + 1$, on two mesh levels. The higher polynomial degree is placed on the coarser mesh, while the lower polynomial degree is placed on the finer mesh level. Fig. (4.6)(b) demonstrates the computational domain using $p = 1$ in the center on the finer mesh level, and $p = 2$ placed on the outer, coarser mesh level. The asymptotic error convergence rate is expected to be limited by the lowest polynomial degree, but have a lower overall L_2-error since the coarser mesh level has a higher solution polynomial degree, namely $p+1$. Fig. (4.7)(b) demonstrates the expected behavior.

L_2-error slopes: $p = 1$						
Mesh Size	Conservative Variable Error Slopes					
	ρ	ρu	ρv	ρw	ρE	Total
h	–	–	–	–	–	—
h/2	1.97	2.34	2.10	–	1.96	2.04
h/4	2.01	2.17	2.29	–	2.01	2.07
h/8	2.01	2.14	2.26	–	2.01	2.05

L_2-error slopes: $p = 2$						
Mesh Size	Conservative Variable Error Slopes					
	ρ	ρu	ρv	ρw	ρE	Total
h	–	–	–	–	–	—
h/2	2.73	2.49	2.26	–	2.74	2.58
h/4	2.85	2.67	2.44	–	2.84	2.69
h/8	2.94	2.85	2.71	–	2.92	2.84

L_2-error slopes: $p = 3$						
Mesh Size	Conservative Variable Error Slopes					
	ρ	ρu	ρv	ρw	ρE	Total
h	–	–	–	–	–	—
h/2	4.20	4.36	4.33	–	4.19	4.24
h/4	4.09	4.42	4.46	–	4.09	4.20
h/8	4.06	4.13	4.28	–	4.05	4.10

L_2-error slopes: $p = 4$						
Mesh Size	Conservative Variable Error Slopes					
	ρ	ρu	ρv	ρw	ρE	Total
h	–	–	–	–	–	—
h/2	4.65	4.39	4.31	–	4.60	4.47
h/4	4.83	4.70	4.68	–	4.78	4.73
h/8	4.94	4.85	4.89	–	4.89	4.89

L_2-error slopes: $p = 5$						
5 Mesh Size	Conservative Variable Error Slopes					
	ρ	ρu	ρv	ρw	ρE	Total
h	–	–	–	–	–	—
h/2	6.22	6.47	6.40	–	6.17	6.25
h/4	6.19	6.19	6.19	–	6.16	6.16
h/8	3.56	3.64	3.84	–	4.49	3.88

Table 4.1: L_2-error slopes using Ringleb flow as reference solution.

4.2.2 Taylor Green Vortex Study

The Taylor-Green Vortex problem is simulated by solving the compressible Navier-Stokes equations using adaptive mesh refinement. The flow is solved on an isotropic domain which spans $[-\pi L, \pi L]$ in each coordinate direction where L is the characteristic length. The initial conditions are given as:

$$\begin{aligned} u &= \quad V_0 \ \sin(x/L)\cos(y/L)\cos(z/L) \\ v &= -V_0 \ \cos(x/L)\sin(y/L)\cos(z/L) \\ w &= 0 \\ p &= \rho_0 V_0^2 \left[\frac{1}{\gamma M_0^2} + \frac{1}{16}\left(\cos(2x)+\sin(2y)\right)\left(\cos(2z)+2\right)\right] \end{aligned} \tag{4.13}$$

where u, v, and w are the components of the velocity in the x-, y- and z-directions, p is the pressure and ρ is the density. The flow is initialized to be isothermal, $\left(\frac{p}{\rho} = \frac{p_0}{\rho_0} = RT_0\right)$, with initial density $\rho_0 = 1.0$, initial velocity $V_0 = 0.1$, and initial pressure $p_0 = 1/\gamma$. For this case the Reynolds number is $Re = 1,600$, the ratio of specific heats is $\gamma = 1.4$, and the Prandtl number is $Pr = 0.71$. The boundary conditions are periodic in all coordinate directions.

Three polynomial degrees, $p = 1, 3, 7$, are simulated with and without adaptive mesh refinement. Simulations without AMR include a fine mesh and a medium mesh, corresponding to 256^3 degrees-of-freedom and 128^3 degrees-of-freedom (DOF), respectively. The AMR simulations use three levels of mesh refinement where the coarsest level has 64^3 DOF, and the finest mesh level corresponds to the finest spatial resolution used in the single mesh simulations, i.e. 256^3. All cases are performed using uniform p-order discretizations. The adaptive refinement process uses a feature-based tagging criterion, where cells are refined if they contain a vorticity magnitude value at or greater than a specified level τ. Three tagging thresholds are examined in this study: $\tau = 0.3$, 0.5, or 1.0. The tagging refinement criterion does not constitute an error estimate, however, this feature-based approach has proven successful for capturing vortical structures for rotorcraft [53] and wind energy problems [39].

The computed dissipation of kinetic energy, number of DOF, total CPU-hours, and L_2-errors are shown in Fig. (4.8) for $p = 1$ simulations, Fig. (4.9) for the $p = 3$ simulations, and Fig. (4.10) for the $p = 7$ simulations. The dissipation in these cases is obtained by calculating the change of kinetic-energy at each time step:

$$\epsilon = -\frac{\mathrm{d}\mathcal{E}_k}{\mathrm{d}t}$$

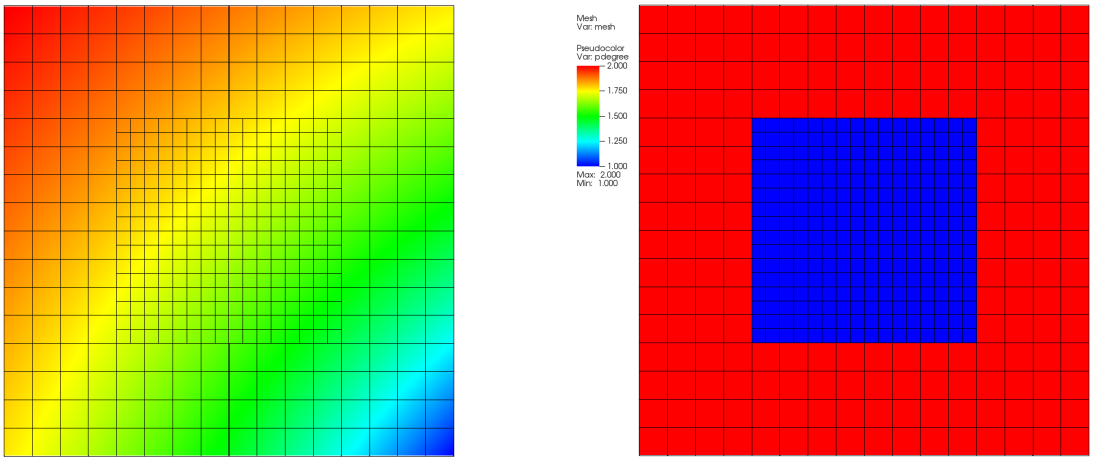
where $\mathcal{E}_k$ is the kinetic energy. The L_2-error is a cumulative error summed over the time history of dissipation, which is computed as:

$$E_t = \sqrt{\left(\sum_{j=0}^{t}\left(\epsilon_j^{fine} - \epsilon_j\right)^2\right)}$$

where ϵ^{fine} is the dissipation calculated using the single fine mesh, and ϵ is the dissipation for the particular case under study.

The major savings for AMR occur in the beginning of the simulation when the turbulent scales are large, and near the end of the simulation when the small turbulent scales have been mostly dissipated to internal energy. For $p = 1$, the highest threshold tagging, $\tau = 1.0$, took the least amount of CPU-hours, followed by the static medium mesh case, while the fine mesh simulation required the most CPU-hours. The simulations with tagging thresholds of $\tau = 0.3$ and 0.5 were more accurate than the medium mesh case, and cheaper in terms of CPU-hours than the fine mesh case. For $p = 3$, the static medium mesh case used the least amount of CPU-hours, and the static fine mesh case took the most CPU-hours. The AMR case with tagging threshold $\tau = 0.3$ was more accurate than the static medium mesh case, and cheaper in terms of CPU-hours than the static fine mesh case.

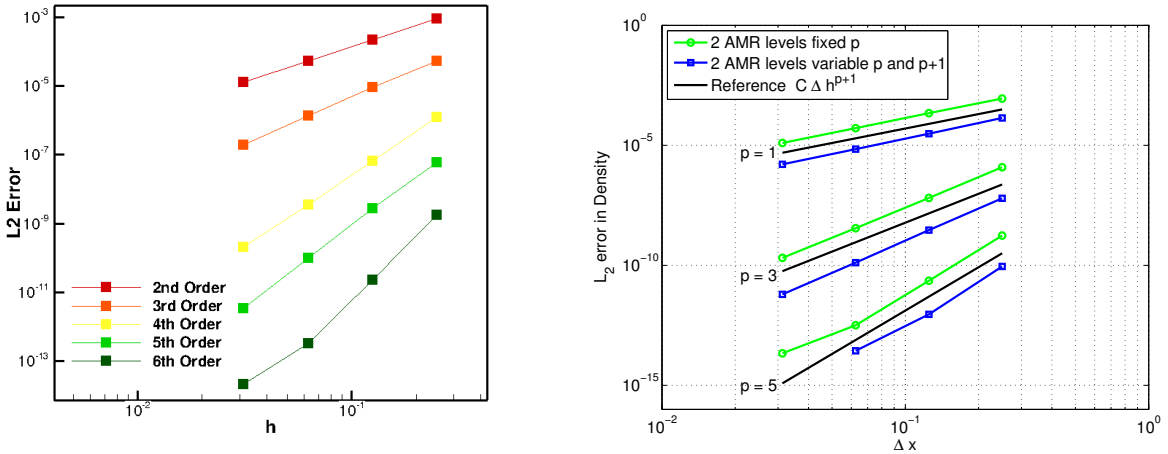
A time series of the velocity magnitude for the Taylor-Green Vortex problem is shown in Figure 4.11 for a single slice located at $z = 1$ in the xy-plane. As the vorticity magnitude increases, the mesh refines dynamically to those regions. As the dissipation begins to weaken, the vorticity magnitude also weakens, causing the mesh to coarsen.



(a) Density contours of the analytic solution.

(b) Two polynomial degrees on two mesh levels.

Figure 4.6: Ringleb flow mesh resolution study for two mesh levels.

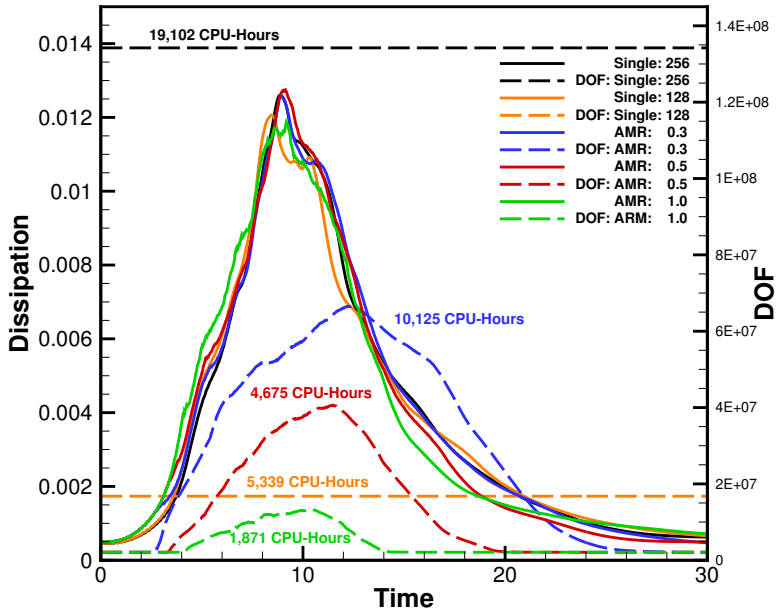


(a) L_2-error for one polynomial degree with two mesh levels.

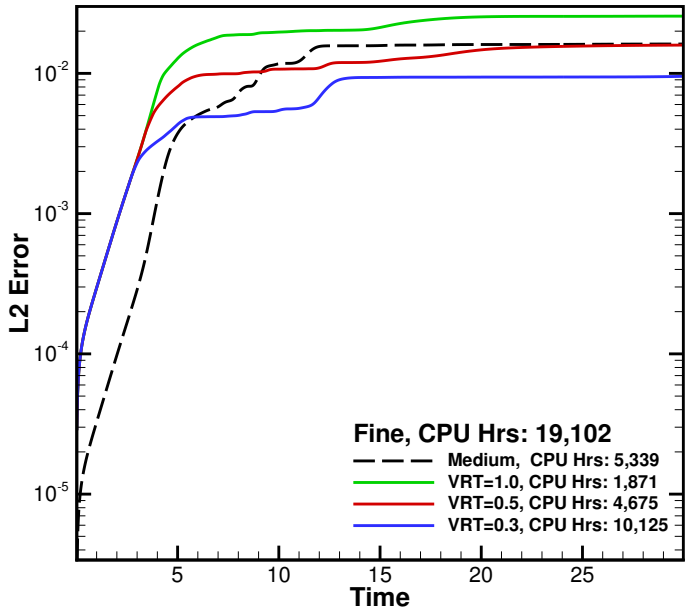
(b) L_2-error of two polynomial degrees with two mesh levels.

Figure 4.7: L_2-error rates for Ringleb flow mesh resolution study.

$$p = 1$$



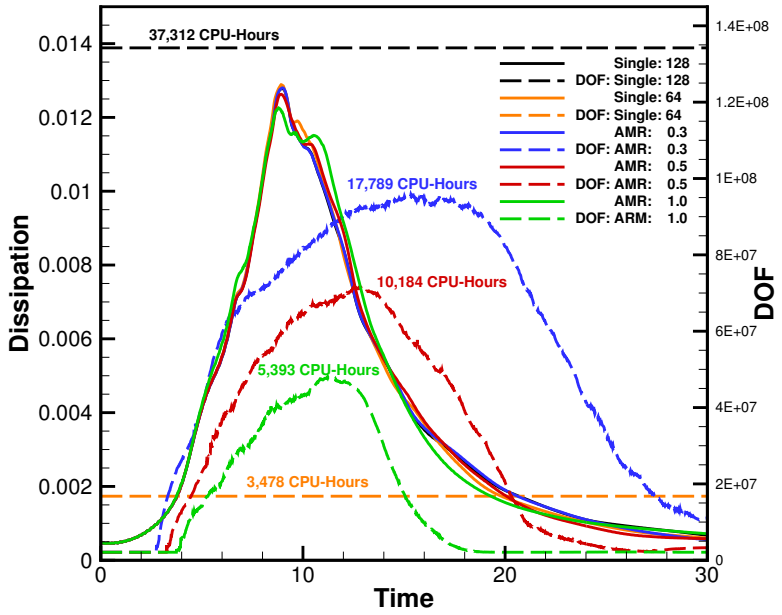
(a) Degrees-of-freedom



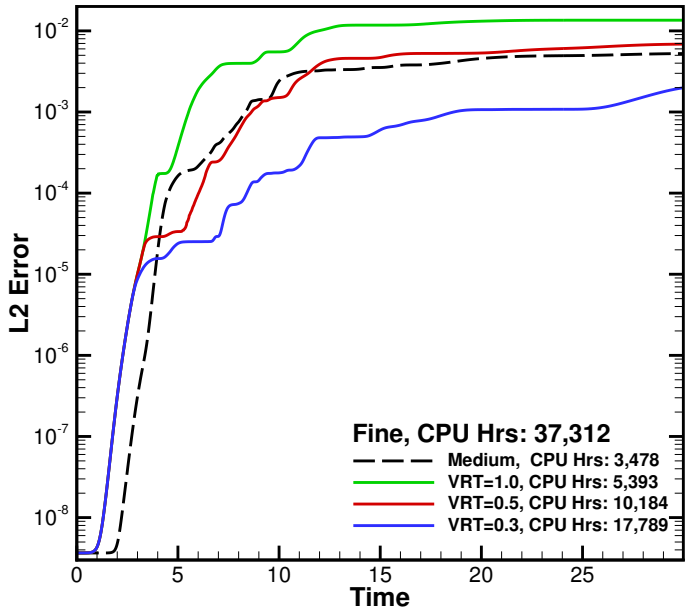
(b) L_2-error

Figure 4.8: Taylor-Green vortex dissipation and L_2-error compared to fixed fine mesh simulation using polynomial degree $p = 1$.

$$\boxed{p = 3}$$



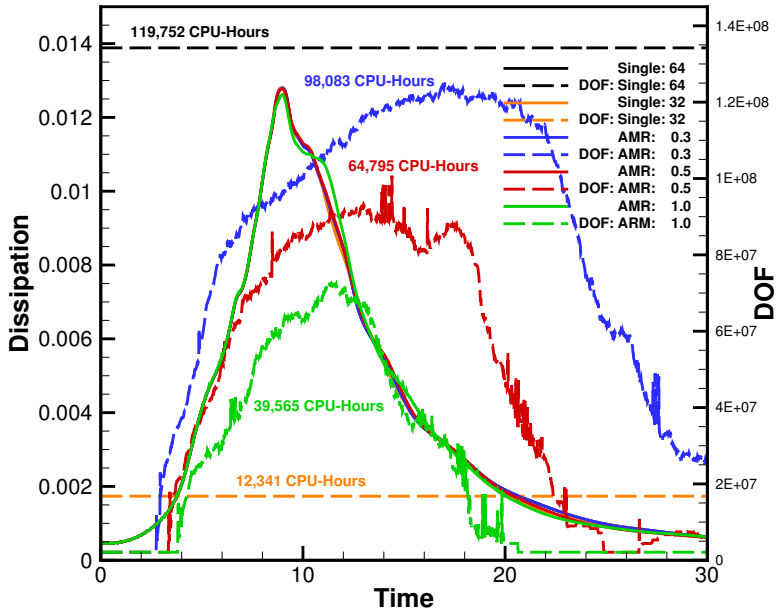
(a) Degrees-of-freedom



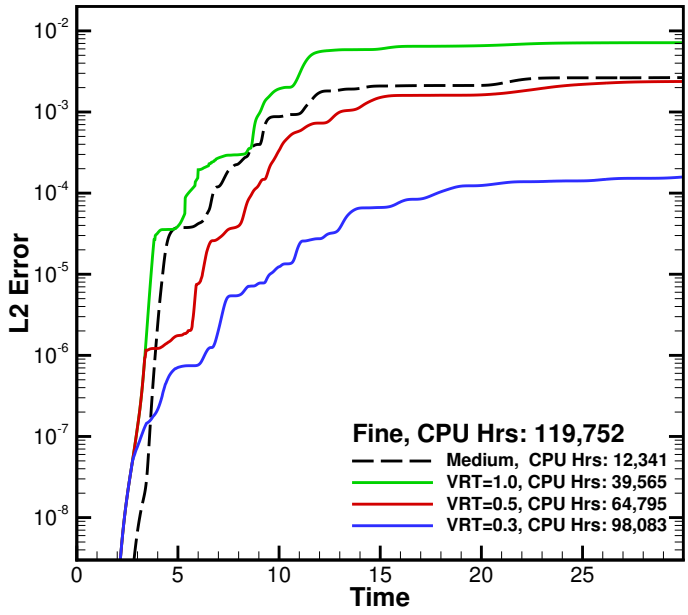
(b) L_2-error

Figure 4.9: Taylor-Green vortex dissipation and L_2-error compared to fixed fine mesh simulation using polynomial degree $p = 3$.

$$\boxed{p = 7}$$



(a) Degrees-of-freedom



(b) L_2-error

Figure 4.10: Taylor-Green vortex dissipation and L_2-error compared to fixed fine mesh simulation using polynomial degree $p = 7$.

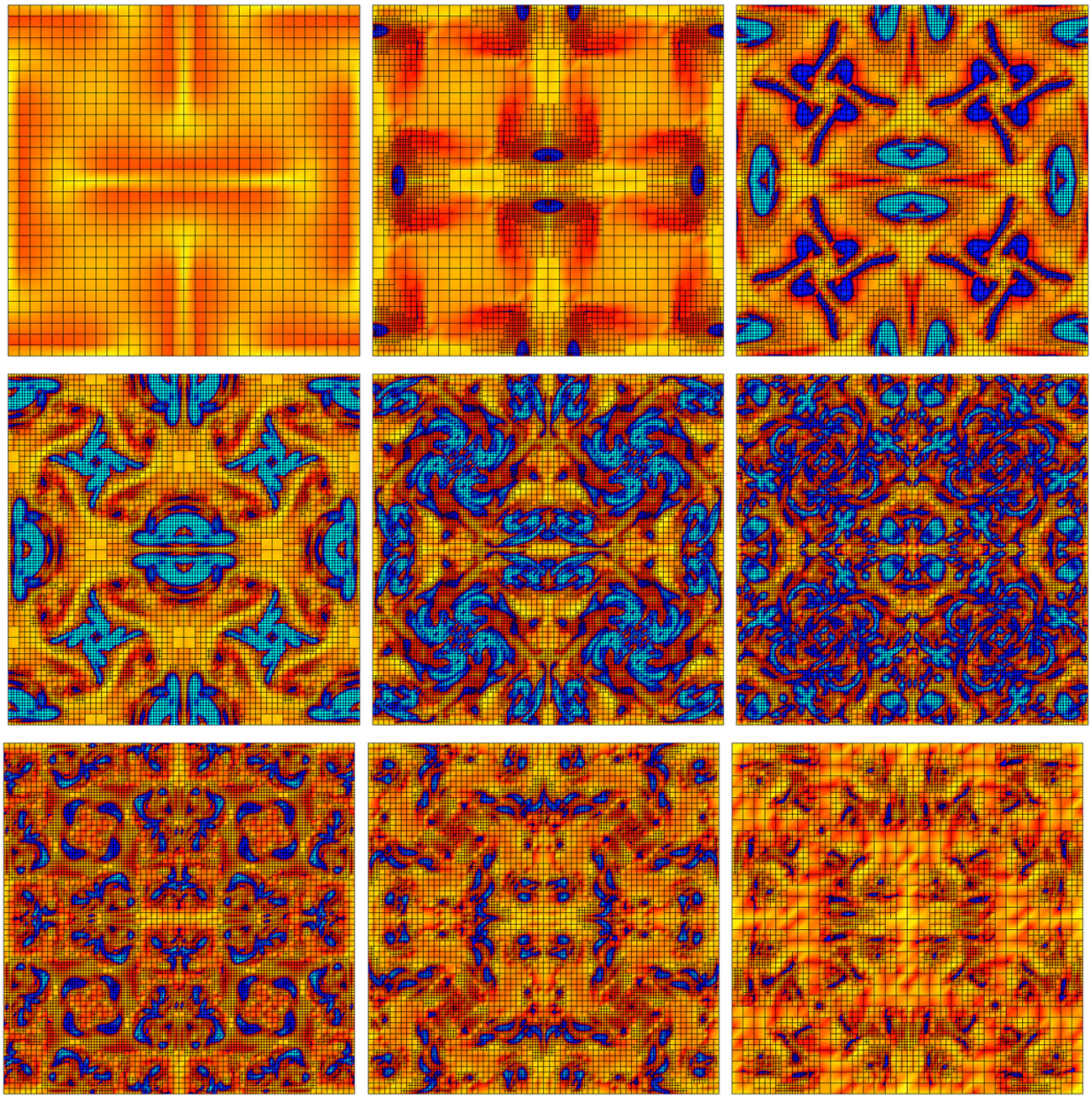


Figure 4.11: Time history of the Taylor-Green Vortex using three levels of adaptive mesh refinement. Contours of vorticity magnitude are shown with the adaptive mesh.

Chapter 5

Computational Simulation Framework

The work herein is embedded inside a computational framework that enables a broad set of problems not limited to Cartesian mesh systems. This chapter introduces the computational framework and all of the components within, along with results from simulation of a sphere at low Reynolds number and a simulation of the three-dimensional NACA0015 wing.

5.1 Computational Methodology

The computational framework deployed in this work is known as the **W**yoming **W**ind and **A**erospace **A**pplications **K**omputation **E**nvironment (W^2A^2KE3D). The framework is derived to have a flexible solver and mesh system paradigm in order to perform simulations for a large class of problems in aerospace and wind energy.

W^2A^2KE3D is designed to support a dynamic overset mesh system using multiple flow solvers and multiple computational meshes. The mesh system generally consists of a collection of *near-body* and *off-body* meshes. The near-body meshes are inherently unstructured and highly anisotropic to model complex geometry and resolve aerodynamic boundary layers. The off-body mesh is a dynamically adaptive Cartesian grid system, via the work discussed in Chapter 3. The use of Cartesian meshes in the off-body region allows for efficient flow solvers, efficient storage, and ease of dynamic solution-based mesh adaption. This multiple mesh paradigm allows for effective use of solver and mesh technologies in variable flow con-

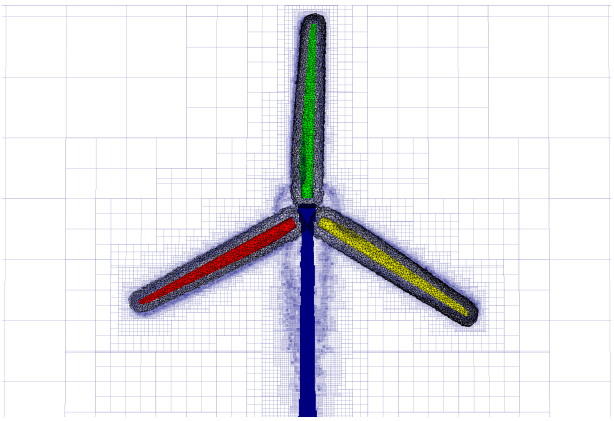


Figure 5.1: NREL 5MW wind turbine overset mesh system. One turbine blade unstructured mesh is replicated three times, rotated and translated to the initial positions. A fourth unstructured mesh is used to represent the tower and nacelle. The off-body adaptive mesh is visualized in the background.

ditions. This hypothesis holds strongly for wind energy applications that require capturing boundary layer phenomenon as well as wake dynamics. Figure 5.1 demonstrates an overset mesh system of a traditional three-bladed wind turbine with a tower and nacelle. Each blade is represented by a single mesh that is replicated, transitioned, and rotated to the correct starting position. The tower geometry is fitted with an independent unstructured near-body mesh component.

The W^2A^2KE3D framework allows for multiple CFD solvers individually optimized for their respective mesh system in the multiple-mesh paradigm. The use of multiple meshes and multiple flow solvers introduces the requirement of coordination. A C-programming-language based driver program choreographs all flow solvers and all mesh systems. This driver allows for solvers to run on disjoint groups of CPU cores, allowing for variable amounts of computational resources to be allocated appropriately where needed. This is particularly important for the off-body solver which uses a dynamically adaptive mesh. During the evolution of a wind turbine simulation, the flow features of interest, such as the wake, require additional mesh resolution. As the propagation of the wake grows over time, more computational resources are required in the off-body region. The flow solvers present in the framework can be redistributed to different numbers of cores at the beginning of restarted simulations. This allows for long run-time simulations to be moderately load balanced.

Additionally, once a solver is implemented into the framework, flow visualization and analysis is provided to the solver through the use of in-situ visualization. Pointers to the flow solver's data and mesh are held by the driver program which is fed to the in-situ flow visualization software implemented directly into the driver.

5.1.1 Near-Body Flow Solver

The near-body flow solver utilized in W^{2}A^{2}KE3D is NSU3D (**N**avier-**S**tokes **U**nstructured) [147,148]. NSU3D is a well-established Unsteady Reynolds Averaged Navier-Stokes (URANS) solver for unstructured meshes. The discretization is based on a vertex-centered finite-volume method with matrix-based artificial dissipation providing second-order spatial accuracy. The solver uses automatic agglomeration multigrid along with line-implicit preconditioning for accelerated solution convergence [149]. NSU3D contains several turbulence models for various aerodynamics problems. This includes the Spalart-Allmaras (SA) [150], K-Omega [151], and the Delayed Detached Eddy Simulation (DDES) [152] turbulence models with rotation/curvature correction [153]. The SA and the DDES turbulence models with rotation correction are the primary methods employed in this work.

NSU3D has been demonstrated on multiple aerodynamics problems and has been a regular participant in the AIAA High Lift Prediction Workshop [154] and the AIAA Drag Prediction Workshop [155] series. The solver has been demonstrated to have good strong scalability on the NCAR supercomputer NWSC-1 Yellowstone up to 32,768 CPU cores. Additionally, NSU3D has served as a near-body flow solver in the CREATE-AV HELIOS [53] software. W^{2}A^{2}KE3D is analogous to HELIOS in that they each utilize the solution strategy composed of a near-body mesh and flow solver, an off-body mesh and flow-solver, and an overset mesh connectivity component.

5.1.2 Off-Body Flow Solver

W^{2}A^{2}KE3D utilizes the variable-order discontinuous Galerkin (DG) finite-element method developed and implemented for this work in the dynamic adaptive mesh refinement (AMR) framework discussed in Chapters 2 and 3. The off-body solver is known herein as **dg4est**.

5.1.3 Overset and Domain Connectivity Assembler

By adopting a multiple-mesh, multiple-solver paradigm in an overset framework, domain connectivity and solution interpolation is required. To fulfill this requirement, the **T**opology **I**ndependent **O**verset **G**rid **A**ssembler (TIOGA) [86, 87, 156] is utilized. TIOGA relies on an efficient parallel implementation of the Alternating Digital Tree (ADT) algorithm in order to handle point-in-cell inclusion tests to determine connectivity. TIOGA determines the donor-receptor patterns of overlapping mesh grids, and performs the solution interpolation using the appropriate solution accuracy orders required by the respective flow solvers. To perform high-order solution interpolation, several call-back functions are provided in the TIOGA API [86]. These functions include: receptor node list generation, high-order donor inclusion test, high-order interpolation weight generation, and interpolated solution conversion to high-order solution coefficients. For high-order methods, the use of multiple points inside each cell is required. Thus the flow solver must provide a list of the node locations inside a cell to be interpolated. Additionally, for high-order methods, the use of high-order mesh geometries may be used, which include curved cells and faces, therefore requiring a high-order approach to provide a donor-inclusion test. The donor-inclusion test for high-order methods maps physical point coordinates to natural coordinates in the standard isoparametric-reference space using a geometric basis function mapping. This transformation forms a system of nonlinear equations which are solved via a Newton-Rhaphson method. Once the natural coordinates are found, it is trivial to test if the point is inside the cell. Once donor cells are identified, the solution interpolation order of accuracy is required to be of the same order as the solution order of accuracy. Lastly, if the high-order numerical method is a modal-based finite element solver, then the interpolated solution requires conversion to solution coefficients, which can be done using either a mass matrix or a Vandermode matrix approach [86]. TIOGA is agnostic to mesh element types and numerical discretizations. Therefore mixed-element meshes can be used concurrently with any combination of numerical discretizations, such as a finite-volume solver with a high-order finite-element method. TIOGA has also been utilized in high-order solution techniques of intersecting Hamiltonian path and strand mesh grids [157].

5.1.4 Micro-Scale Atmospheric Inflow Coupler

Faithful representation of wind plants through simulation requires capturing all fluid scales and physical environments. This introduces complex terrain and atmospheric inflow conditions thus requiring meteorological micro-scale flow conditions. To achieve this, the large fluid scales are introduced through a one-way coupling between precursor atmospheric turbulence solvers and the off-body flow solver. The off-body flow solver then transfers these atmospheric conditions to the near-body CFD flow solver via the overset assembler and interpolator. The flow coupler incorporates a choice of two precursor atmospheric solvers: the National Center for Atmospheric Research's (NCAR) **W**eather **R**esearch and **F**orecasting (WRF) [158] model and the National Renewable Energy Laboratory's (NREL) **S**imulation f**O**r **W**ind **F**arm **A**pplications (SOWFA) [64] model. SOWFA has been extensively used in simulation of wind plant modeling in various atmospheric conditions [65] and complex terrains [159–161].

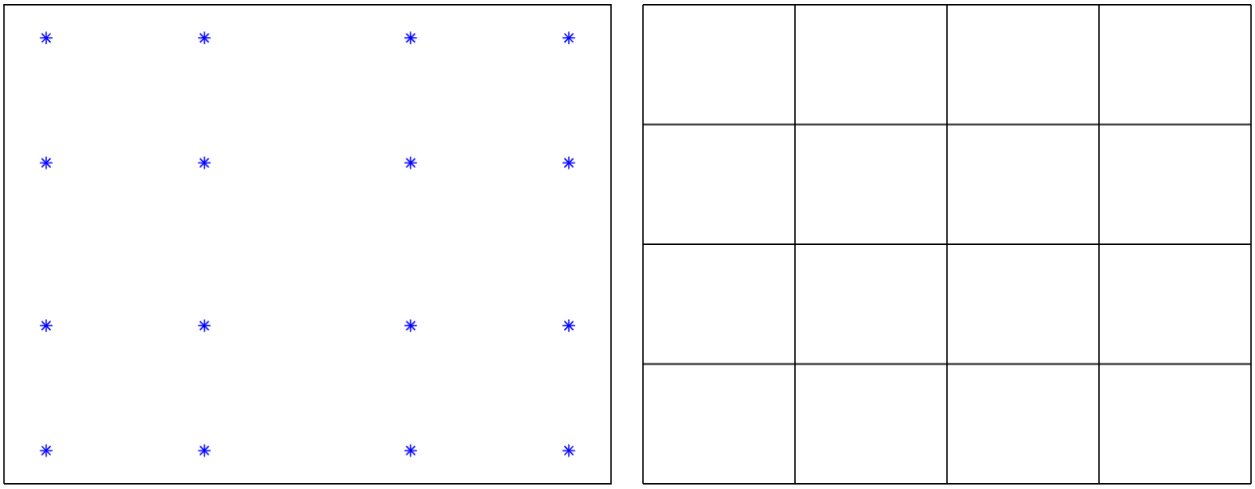
The atmospheric solvers are run as precursor simulations prior to the CFD simulation to accurately capture the atmospheric boundary layer (ABL). These precursor simulations generate the initial and boundary flow field conditions for the CFD flow solvers. An intermediary pseudo-flow solver reads in the precursor atmospheric flow mesh and data over the duration of the wind plant simulation. This pseudo-flow solver is treated in similar fashion as a regular CFD solver to the overset grid connectivity assembler. The mesh and data are registered with TIOGA as its own mesh system from which the atmospheric data is interpolated on to the CFD solver mesh system to establish the entire initial condition, and on to the CFD mesh system boundaries at each subsequent time step of the simulation. Between precursor atmospheric flow solutions, which are written in files, the pseudo-solver performs linear time interpolation of the data. TIOGA then spatially interpolates the atmospheric data, in a one-way coupling manner, to the CFD flow solver. Additionally, the ABL flow solver can be run simultaneously with the CFD simulation.

5.1.5 Flow Visualization and Post-Processing

Data sets generated by high-fidelity numerical simulations of wind energy applications that resolve the blade aerodynamics can span 10-12 orders of magnitude in spatial scales and 4-6 orders of magnitude in temporal scales. Saving frequent volume data-sets can quickly accumulate to hundreds or thousands of terabytes of data. Therefore, the ability to post-process flow visualization becomes intractable due to the shear amount of data. To alleviate the big-data issue, flow visualization and data analysis are no longer performed as a post-processing step by reading in data written to disk; analysis and visualization are performed while the data is being generated in the simulation, known as *in-situ* visualization and analysis. In-situ analysis techniques allow for data extraction and collection at 10x to 1000x data reduction.

This work utilizes the VisIt Libsim [162] in-situ library, which is implemented directly into the driver program that coordinates all flow solvers within the W^2A^2KE3D framework. This procedure is tightly coupled as the driver program has access to all flow solver solution pointers, which are passed directly through to the Libsim interface. Thus any CFD solver implemented into the framework gains access to in-situ visualization and data analysis capabilities automatically. The Libsim interface is directed by input scripts which are dynamically read allowing for changes during run-time. For example, without stopping and restarting the simulation, the user may change any parameters related to the in-situ visualization such as file output type and frequency, iso-contour values or variables, or cut-plane locations or variables.

A data adapter is developed to convert simulation data to and from the Libsim data model. This data adapter is indifferent with finite-volume and finite-element method data; the data model assumes a point-wise data structure with linear elements between solution points. In order to achieve full resolution visualization for the high-order finite elements, subdivision of the finite-element cell solution is performed to generate multiple linear sub-cells. This subdivision is flexible in the choice of the number of linear sub-cells. Practically, we subdivide a p-degree high-order element into $p+1$ uniform sub-cells in each coordinate direction as shown in Fig. (5.2). This technique enables output of higher-order iso-surfaces



(a) High-order element solution points. (b) Subdivided element into linear sub-elements.

Figure 5.2: High-order element subdivision for higher-order plotting.

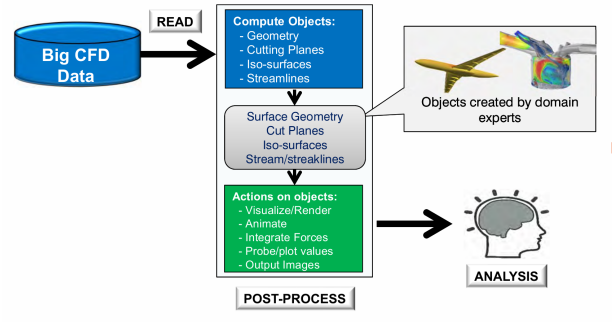
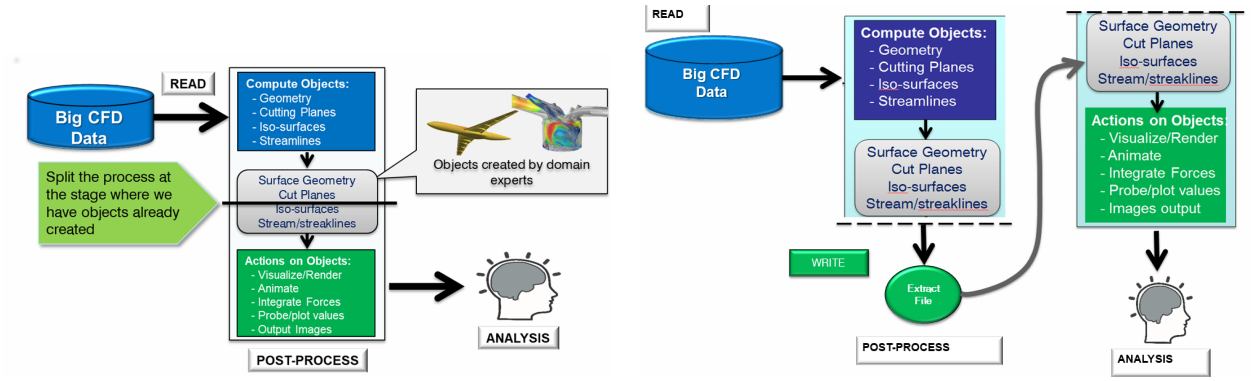


Figure 5.3: Traditional post-processing workflows used for scientific analysis.

and cut-planes at run-time which provides locally lossless data reduction by avoiding output of large CFD volume data.

Traditional CFD workflows write large volume data sets to disk for later post analysis. Fig. (5.3) demonstrates the traditional post-processing workflow starting with CFD data being read into a visualization software where objects are created, such as cutting planes, iso-surfaces, and streamlines, followed by actions performed on the newly formed objects, including visualization, animation, and image exportation. This procedure may be costly and time-consuming for object construction, which at the completion of the post-processing session, is discarded.



(a) Splitting the post-processing workflow

(b) Extract-based workflow

Figure 5.4: Extract-based workflows allow for continued and repeated analysis on data extracts.

To combat this issue and increase productivity, the post-processing workflow can be split into multiple stages as shown in Fig. (5.4)(a). After the data objects are created in the scientific post-processing software, they are extracted and stored for repeated use, as demonstrated in Fig. (5.4)(b), and, further, new data objects can be created from the data extracts. One such extraction-based workflow is the FieldView eXtract DataBase (XDB) approach. In an XDB workflow, shown in Fig. (5.5), the post-processing objects are created and saved in FieldView as an XDB file, which can be used repeatedly.

Further, an in-situ workflow can be adopted where the data objects that are usually extracted during post-processing are extracted during run-time instead. This increases productivity substantially by avoiding costly data writing and transferring of large volume datasets. Fig. (5.6) shows a combined in-situ XDB workflow which starts with data extraction using VisIt Libsim, then directly exports XDB files.

A hybrid in-situ workflow is adopted using VisIt Libsim for the run-time data extraction of data objects. The data can be written as XDB files or Silo [163] files, which can be visualized in parallel via remote-hosting with FieldView or VisIt, respectively. Using the remote-host capability, the extracted data objects remain on the computing host, thereby removing data movement and, thus, increasing productivity. Using VisIt, imported data can be exported as a different file type; VisIt currently supports 12 export formats [162] including FieldView XDB.

Figure 5.5: FieldView eXtract DataBase (XDB) workflow.

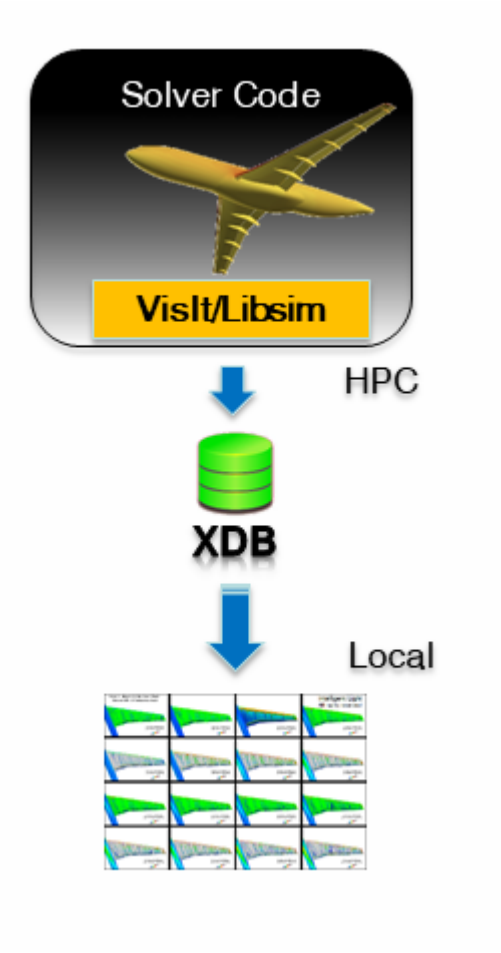


Figure 5.6: In-situ XDB workflow: VisIt Libsim can directly output XDB formatted files.

5.1.6 Driver

The driver software is responsible for controlling all component solvers embedded in a multiple-mesh and multiple-solver, overset framework. Inevitably, different meshes and independent flow solver speeds introduce variable amounts of computational work and efficiency. In a parallel computing environment, the software developer is presented with a few options to make computational load balancing more amenable: (i) place all flow solvers on all CPU cores, (ii) allocate disjoint groups of CPU cores to each flow solver. In the former solution, all flow solvers are partitioned across all CPU cores and execution of the flow solvers are serialized with respect to each other. The latter solution allows for flow solvers to execute in parallel and allows for each flow solver to have different numbers of CPU cores. In W^2A^2KE3D, option (ii) is chosen for the flexibility it provides regarding different solver requirements and scalability. However, this flexibility can add development and algorithmic complexity. As a simulation evolves, the off-body solver may be dynamically adapting which can introduce more overall degrees-of-freedom. This can lead to a load imbalance of the flow simulation. To alleviate this problem, redistribution of the problem can be performed between simulation campaign restarts for more effective use of computational resources. That is, more CPU cores are allocated to the flow solver that requires more computational resources as the solution evolves. Each flow solver component in the W^2A^2KE3D framework has this capability: NSU3D needs to be manually redistributed by repartitioning the restart file before execution resumes, but `dg4est` has automatic redistribution when provided more CPU cores for a restarted solution.

A sample driver software work flow is portrayed in Fig. (5.7) which illustrates a scenario with two near-body mesh groups and one off-body mesh group. The sample demonstrates each flow solver is independent to solve their respective time-dependent problem in parallel without dependence of other solvers or meshes. During a simulation, a global time orchestrates when data is exchanged between the near-body mesh and the off-body mesh. When an unsteady time step is executed, the near-body and off-body flow solvers iterate in time in an uncoupled manner for a Δt time step which we refer to as the global time step. When the global time step is completed, data is exchanged between the near-body and the off-body

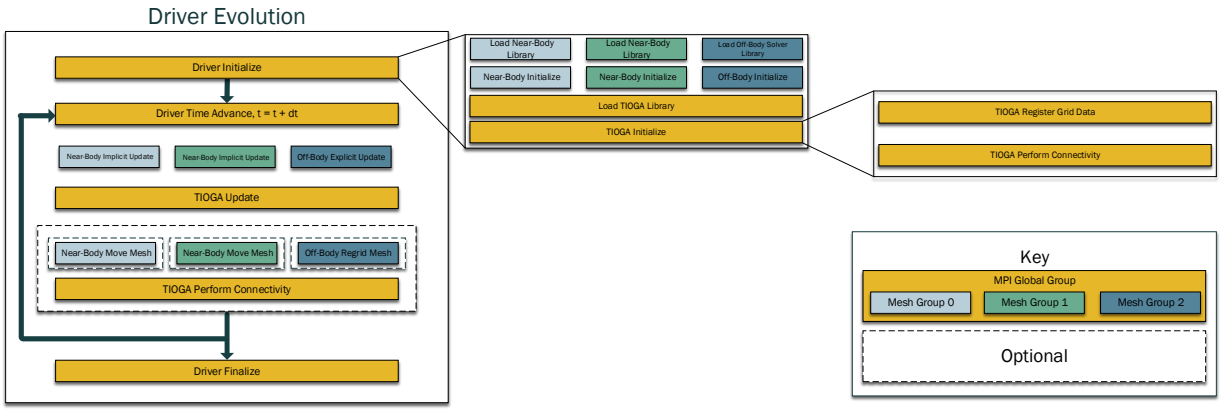


Figure 5.7: A driver code is used to choreograph all flow solvers, mesh movement and adaption, overset data update and grid connectivity, and in-situ visualization. All flow solvers are allocated disjoint groups of CPU cores for parallel flow solution updates.

solvers through the overset interface. Cells receiving data through the overset interpolation are known as receptor cells. These receptor cells zero their respective residual values and their solution is provided by the solver's counterpart mesh solution (near-body solution from off-body solution, and vice-versa). This receptor interpolated solution is then held fixed during a global time step therefore serving as a *boundary condition* for surrounding mesh elements. Thus the global time step dictates the duration of interpolated solution which serves as a fixed value boundary condition to surrounding elements. Following the time step update of the solvers, the overset assembler performs a data update in which the solutions are exchanged between meshes. The flow solvers then have the ability to perform auxiliary functions such as mesh motion or mesh adaption. If any auxiliary meshing routines are called, the new grid information is registered and processed with the overset assembler.

5.2 Computational Framework Validation

This section presents results for the computational framework using the overset paradigm with a mix of higher-order solution methods with finite-volume methods for unsteady problems.

5.2.1 Sphere

The first case is a simulation of an unsteady, low Reynolds number, DNS flow over a sphere. Flow over a sphere is a good validation case because it has been studied extensively in both experiments and numerical simulations. For this simulation, the free-stream Mach number is $M = 0.3$, and the Reynolds number is $Re_D = 1000$, based on sphere diameter. This case combines a near-body unstructured DG solver with the off-body AMR DG solver using an overset mesh. The near-body mesh is a strand grid which consists of prismatic cells. This grid is created by extruding (radially outward) an unstructured triangular mesh on the surface of the sphere. The near-body DG solver is third-order accurate ($p = 2$) in space and is solved on 64 cores. The off-body DG solver is fourth-order accurate ($p = 3$) in space and solved on 256 cores. Four levels of grid refinement are used in this simulation. Fig. (5.8) shows the wake created by the sphere as iso-contours of vorticity magnitude. Also shown is the adaptive off-body grid which closely tracks the vortices in the wake as they travel downstream.

Validation of this case is performed by comparing the drag coefficient on the sphere to both experimental data and other simulations. Since the flow is unsteady, the drag coefficient is time averaged. The time history of the drag coefficient is shown in Fig. (5.9) along with a running average. Also shown in Fig. (5.10) is a close view of the final running average. The running average is calculated using a moving average with a bin size of 2.5×10^5 time steps. This averaging procedure ensures that the initial transients do not adversely affect the time-averaged drag coefficient of the fully developed flow.

The final time averaged drag coefficient is shown in Table (5.1) along with experimental data, simulation data, and data correlation. The correlation for the drag coefficient in

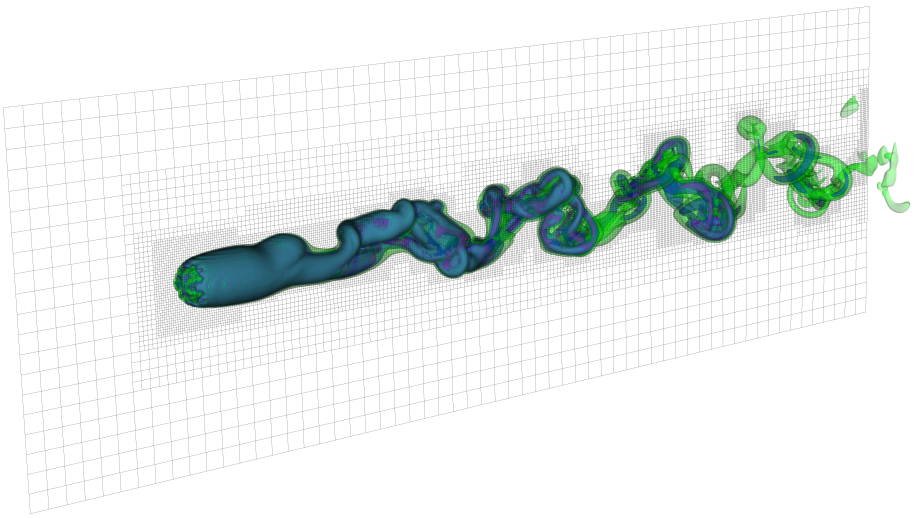


Figure 5.8: Iso-contours of vorticity magnitude for flow over a sphere with $Re_D = 1000$.

Table 5.1: Drag coefficient compared with data from literature.

Re_D	Present	Ref. [164]	Ref. [33]	Eqn. (5.1)
1000	0.475	0.4818	0.476	0.4841

uniform flow around a sphere is given by the following equation:

$$C_D = \frac{24}{Re_D} + \frac{2.6\left(\frac{5.0}{Re_D}\right)}{1+\left(\frac{5.0}{Re_D}\right)^{1.52}} + \frac{0.411\left(\frac{Re_D}{263{,}000}\right)^{-7.94}}{1+\left(\frac{Re_D}{263{,}000}\right)^{-8.0}} + \left(\frac{Re_D^{0.80}}{461{,}000}\right) \tag{5.1}$$

where Re_D is the Reynolds number based on diameter of the sphere. The presented drag coefficient is very similar to the referenced numerical simulation [33] and reasonably close to the experimental data [164]. This demonstrates that the DG solvers combined in an overset framework are capable of accurately simulating unsteady, low Reynolds number flow around simple geometries. Also, high-order methods combined with AMR result in a very large reduction in the number of degrees-of-freedom necessary to solve problems of this type.

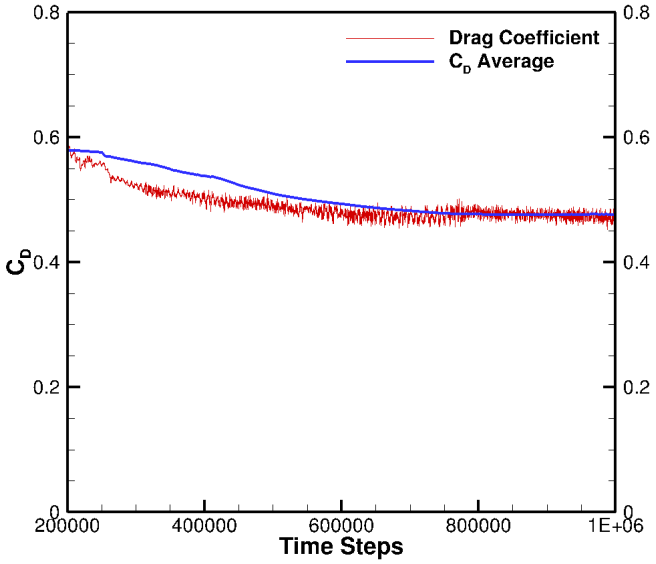


Figure 5.9: Time history of drag coefficient and running average.

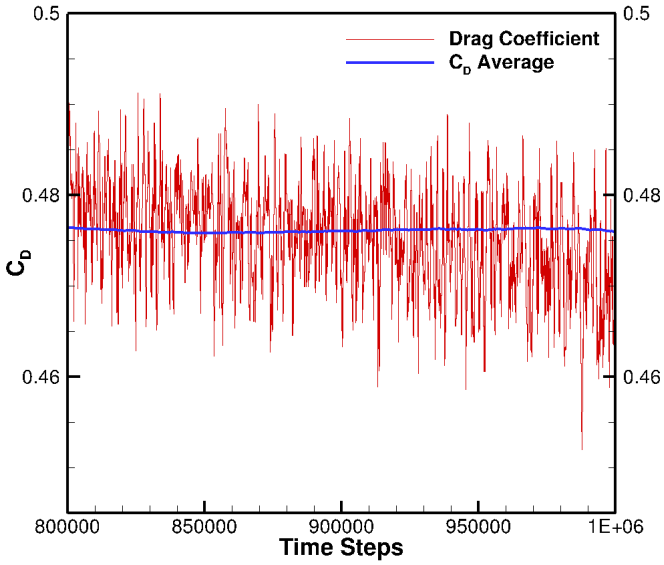


Figure 5.10: Close view of running average.

Figure 5.11: Drag history of flow over a sphere, $Re_D = 1000$ and Mach= 0.3.

5.2.2 NACA0015

The second simulation consists of the study of flow over a wing based on a NACA 0015 airfoil. The NACA 0015 wing has been studied experimentally by McAlister and Takhashi [165]. Computational studies have been performed by Wissink [166], Sitaraman and Baeder [167], and by Hariharan and Sankar [168]. This case demonstrates the capability of simulating unsteady, turbulent flow over a rectangular square-tipped lifting wing. The wing is based on a constant cross-section NACA0015 airfoil, and has a finite span of 6.6 chords. The Mach number is 0.1235, the angle of attack is $\alpha = 12^{\circ}$, and the Reynolds number is 1.5 million.

NSU3D is used for the near-body solver, and the AMR DG solver is used for the off-body solver. The overset mesh configuration is shown in Fig. (5.12). This case is used to demonstrate the ability to combine a node-based finite-volume discretization with a cell-based, high-order, DG discretization in an overset framework, and to accurately capture wing-tip vortices. NSU3D solves the unsteady RANS equations closed by the Spalart-Allmaras with Rotation Correction (SA-RC) turbulence model [153]. NSU3D employs a steady-state implicit solver on a mesh with 4.4 million nodes. The off-body AMR solver runs explicitly using a fourth-order Runge-Kutta time discretization with a non-dimensional time step of $\Delta t u_{\infty}/c = 0.036$. Fig. (5.13) shows iso-contours of vorticity equal to 1.0 and 3.0 and slices of AMR grids to 48 chords downstream.

Three ranges of polynomial degrees are studied in this case for the off-body DG discretization. The first is uniform $p = 1$, the second is $p = 2$ to attach to the near-body and $p = 3$ in the wake, and the third is $p = 3$ to attach to the near-body and grows to $p = 5$ in the wake. Off-body refinement occurs when vorticity magnitude w_c is greater than a tolerance of $\tau = 0.4$. The non-linear residual convergence and lift and drag coefficients are shown in Fig. (5.14) for only the $p = 3 - 5$ case but show similar trends to the other cases. A residual reduction of four orders of magnitude for the near-body solver is obtained although full convergence to steady-state is inhibited due to the time dependent nature of the off-body flow solver. The lift and drag time histories exhibit small oscillations and a moving average with a window of 10,000 non-linear near-body solution updates is used to estimate the lift and drag as shown in Table (5.2).

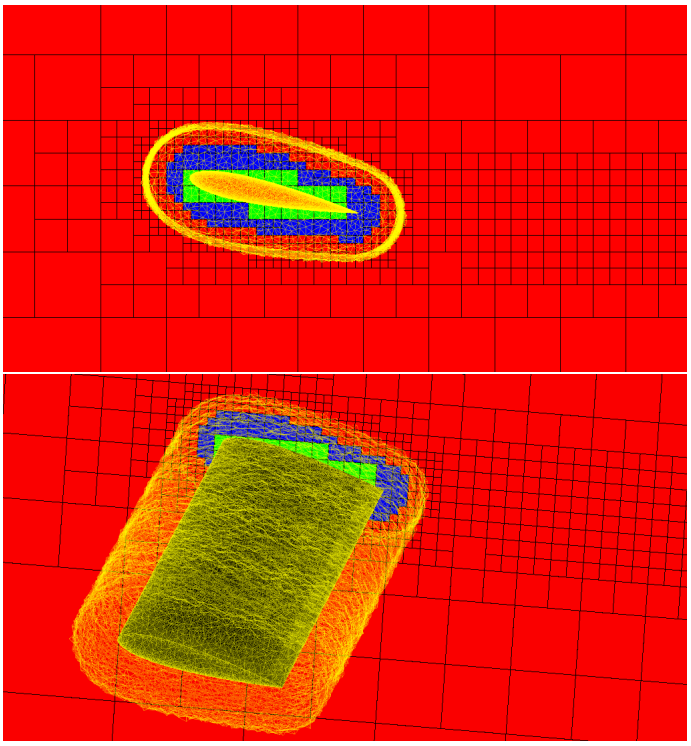


Figure 5.12: Overset meshes for the near-body and off-body NACA 0015 wing.

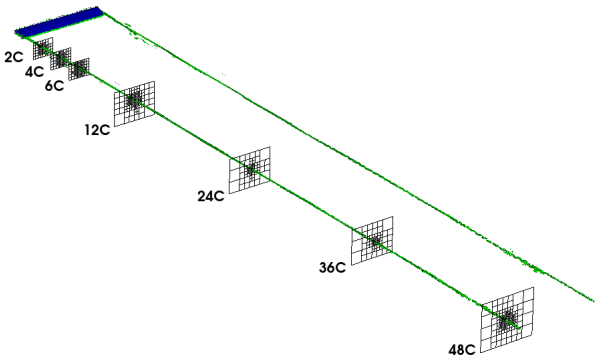


Figure 5.13: Iso contours of vorticity on NACA 0015 wing for $p = 2 - 3$.

The lift and drag for each of these cases shown in Table 5.2 are also compared to experimental data [165], and HELIOS simulation results taken from reference [169]. HELIOS uses the same near-body solver, but uses a fifth-order finite-difference method implemented into the SAMRAI adaptive mesh refinement framework for the off-body solver. The HELIOS simulations used the same near-body resolution as the current results. HELIOS has improved these results recently [170, 171], but, at the same time, modified the near-body mesh. Therefore, comparison to HELIOS results in reference [169] is performed.

Table 5.2: Coefficient of lift and drag for NACA 0015 at $\alpha = 12^\circ$ and $Re = 1.5 \times 10^6$

	Experiment	HELIOS	$p = 1$	$p = 2 - 3$	$p = 3 - 5$
C_L	1.04	0.91950	0.91790	0.91895	0.91881
C_D	0.049	0.0568	0.06223	0.06013	0.06019

Fig. (5.15) shows wake profiles downstream from the wing at 1, 2, 4, and 6 chords. The three polynomial degree ranges are shown where the higher polynomial degrees resolve the wakes more accurately.

Fig. (5.16) shows the wake profiles of the best case ($p = 3-5$) compared to HELIOS and the experimental data at 2, 4, and 6 chords. Also shown is a comparison between HELIOS and the overset case herein with $p = 3 - 5$ at 12 chords downstream. The results show slightly better vortex resolution compared to the HELIOS results from reference [169].

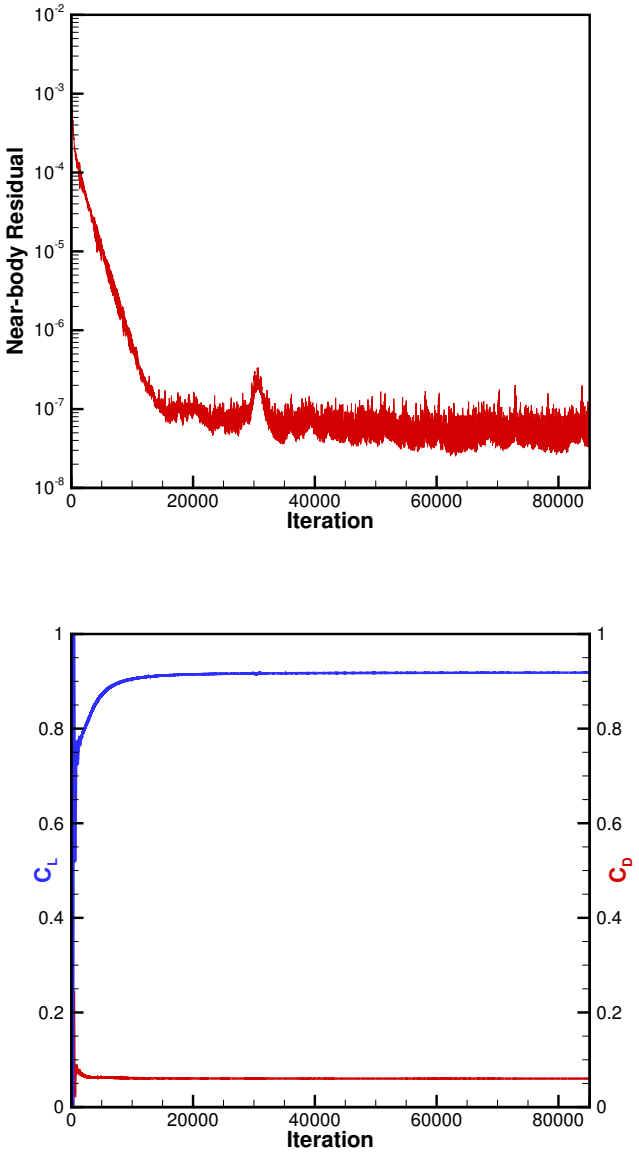
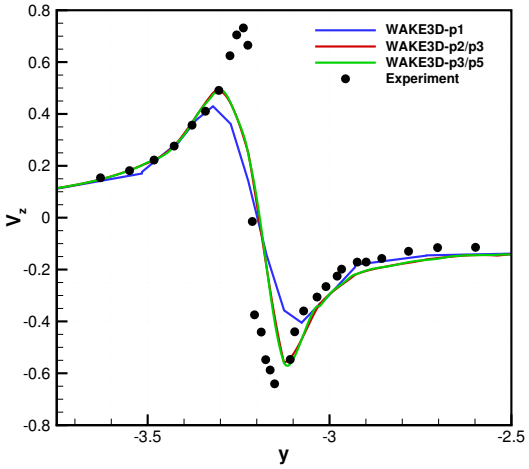
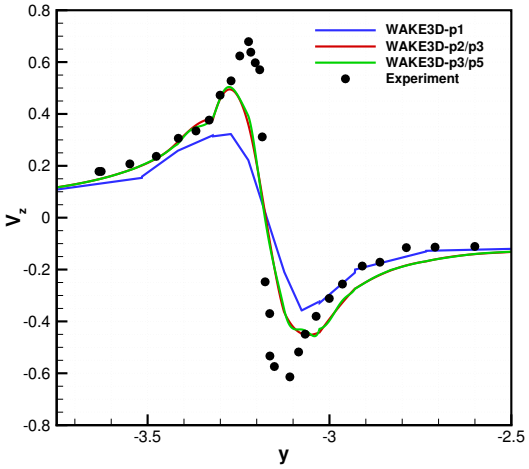


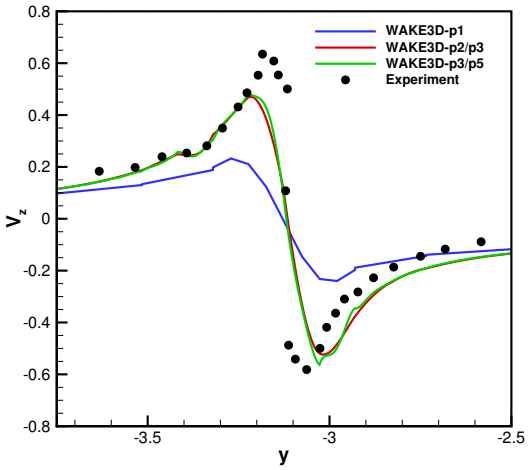
Figure 5.14: Residual convergence (top) and coefficient of lift and drag time history for NACA0015 wing using $p = 3 - 5$.



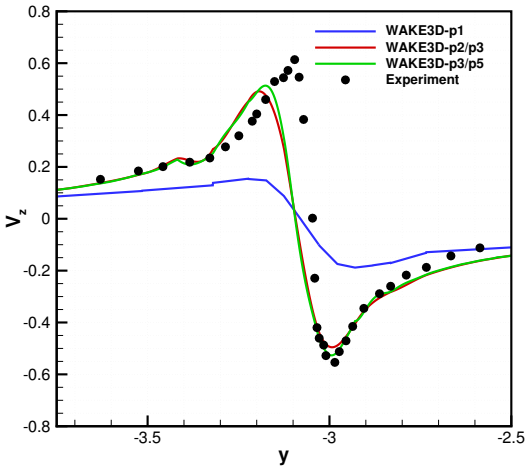
(a) 1 chord downstream



(b) 2 chords downstreams

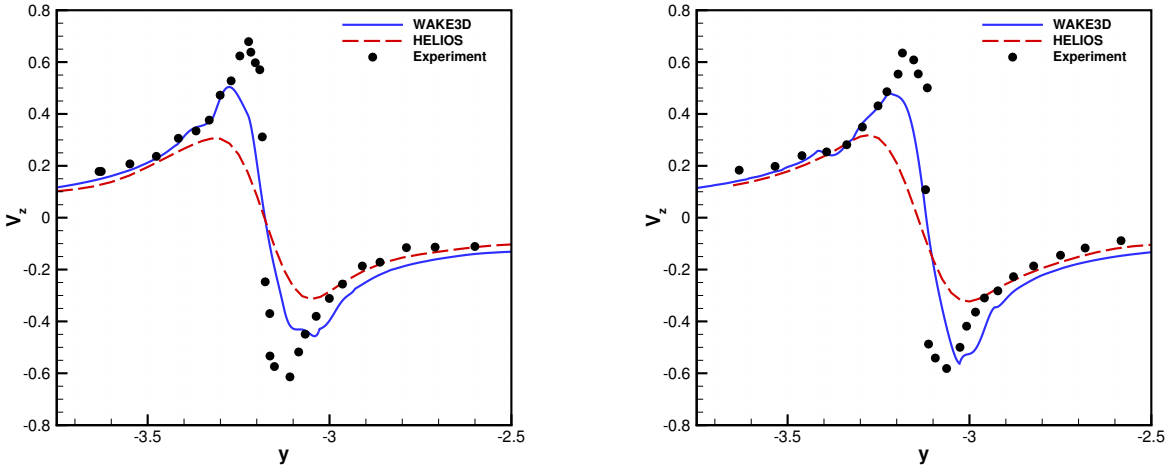


(c) 4 chords downstreams



(d) 6 chords downstreams

Figure 5.15: Velocity wake profile downstream from wing for polynomial degrees $p = 1$, $p = 2 - 3$, and $p = 3 - 5$.



(a) 2 chords downstreams

(b) 4 chords downstreams

(c) 6 chords downstreams

(d) 12 chords downstreams

Figure 5.16: Velocity wake profile downstream from NACA0015 for polynomial degrees $p = 3 - 5$ compared to HELIOS.

Chapter 6

Single Wind Turbine Simulation Results

In this chapter, the computational framework W^2A^2KE3D is demonstrated for four wind turbines: (i) NREL 5MW, (ii) NREL Phase VI, (iii) Siemens SWT-2.3-93, (iv) NREL WindPact-1.5MW. For the NREL 5MW turbine, mesh resolution, time refinement, and sub-iteration studies are performed for the near-body time-accurate solver. Analysis of single turbine performance for multiple uniform inflow velocities is performed for the NREL 5MW and for the NREL Phase VI wind turbine. The NREL WindPACT-1.5MW wind turbine case studies the wake quantitatively using Reynolds stress and Proper Orthogonal Decomposition analysis. Lastly, demonstration of the micro-scale atmospheric and CFD coupling using the WRF and SOWFA solvers is performed.

The near-body meshes are constructed for stand-alone simulations and tested for accuracy. Once the mesh is validated in a simulation using the near-body solver in stand-alone mode, the mesh is trimmed to a user specified distance from the surface of the body and overset with the background Cartesian mesh. For all overset simulations, the off-body mesh system uses as many refined mesh levels as necessary to match the grid resolution of the near-body unstructured mesh cells located at the trimmed mesh boundary. The finest level in the AMR mesh system for the off-body mesh uses $p = 1$, second-order spatial solution accuracy and all coarser levels use higher polynomial degrees. Thus off-body cells that serve

as donors and receptors to the near-body mesh match the second-order spatial accuracy of the near-body discretization, but away from these areas on coarser mesh levels, the solution order of accuracy is increased. This solution accuracy strategy is adopted to mitigate the explicit CFL time step restriction for the off-body solver. The time step restriction of the small $p = 1$ cells is roughly the same as using larger elements with higher polynomial degrees. For all simulations, all solvers are evolved using an unsteady formulation with time-accurate methods. The near-body solver employs the implicit BDF-2 method, and the off-body solver executes the classical explicit Runge-Kutta four-stage method (RK4). Each global time step is loosely coupled; at the end of each global time step, the flow solutions on each mesh are exchanged and interpolated to their counterpart receptor cells.

6.1 NREL 5MW

The NREL 5MW turbine is a concept design aimed at assessing offshore wind turbine technologies. The wind turbine design is a conventional three-bladed upwind variable-speed blade-pitch-to-feather-controlled turbine [172]. This model turbine is used for reference to standardize baseline offshore wind turbine specifications. The NREL 5MW turbine has a blade radius of 63.0 meters with coning angle of 2.5° and a shaft angle of 5°. The simulations assume rigid blades and a rigid tower with nacelle of height 90.0 m. The rated rotor speed is 12.1 revolutions per minute at nominal conditions.

6.1.1 Mesh Resolution Study

We perform a mesh convergence study using two coarse meshes, one medium mesh, and one fine mesh. Table (6.1) outlines the mesh statistics for each of the meshes. The coarsest mesh contains approximately 360,000 mesh nodes per blade where are the fine mesh contains nearly 2.88 million mesh nodes per blade. The tower mesh for all mesh resolution cases is fixed at just over 500,000 nodes. For a full turbine configuration, the total node count can vary from 1.58 million to nearly 9.12 million. We note that the coarse, medium, and fine meshes are representatives of a family of meshes where the coarse and fine meshes are derived

from the medium mesh, while the mesh denoted by coarse* is generated independently.

Mesh	Mesh Points	Tetrahedra	Pyramids	Prisms
Coarse*	474,383	780,283	8,926	661,274
Coarse	360,148	473,747	9,557	539,715
Medium	927,701	1,922,304	12,928	1,162,586
Fine	2,873,862	6,898,579	28,751	3,306,509

Table 6.1: Mesh statistics used in the mesh convergence study of the NREL 5MW wind turbine blade. Each blade mesh is replicated and placed into the correct starting position at the beginning of the simulation. The coarse, medium, and fine meshes are a family of meshes; the coarse and fine meshes are derived from the medium mesh. The coarse* mesh is constructed independently.

The rated power of this wind turbine, as the name suggests, is 5 mega-watts at the nominal inflow rate of 11.4 m/s. We perform the mesh convergence study using the nominal inflow velocity with a time step corresponding to `1/4`° of turbine rotation. The near-body solver uses 50 sub-iterations for the BDF-2 time step. The explicit method, used by the off-body solver, is limited by the CFL number. Thus, it performs sub-cycles using its maximal stable time step until the global time step corresponding to `1/4`° of rotation is reached.

Fig. (6.4)(a) shows the power time history convergence at inflow velocity 11.4 m/s for each mesh over the evolution of rotor revolutions, and Fig. (6.4)(b) shows their respective thrust convergence histories. The target thrust value at 11.4 m/s is 730,000 Newtons. As demonstrated in the mesh convergence study figures, the need for at least medium refined meshes is required at nearly one million nodes per blade to capture the power correctly. The thrust convergence shows slightly more variation between the medium and fine meshes, although all values are clearly converging with additional mesh resolution. This provides an estimate that over two million mesh points per blade are required to accurately capture the aerodynamic forces on the wind turbine. Results for the medium and the fine meshes are close in both power and thrust forces in comparison to the two coarse meshes. Even though the mesh denoted coarse* has more mesh elements and nodes in comparison to the standard coarse mesh, the mesh nodes in the Coarse mesh are more appropriately placed along the blade edges and tips leading to better results. A noticeable difference in the power prediction between the two coarse meshes is shown in Fig. (6.4)(a) while the thrust prediction is only

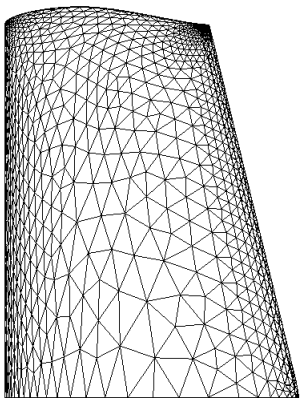


Figure 6.1: NREL 5MW coarse mesh: 360,148 points.

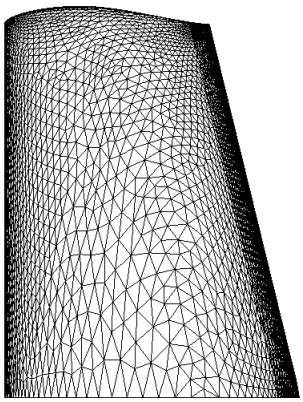


Figure 6.2: NREL 5MW medium mesh: 927,701 points.

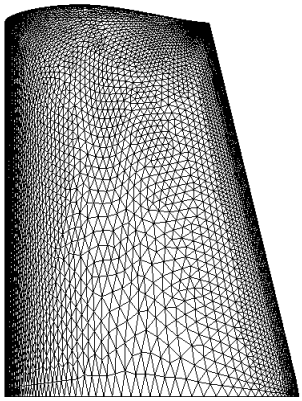
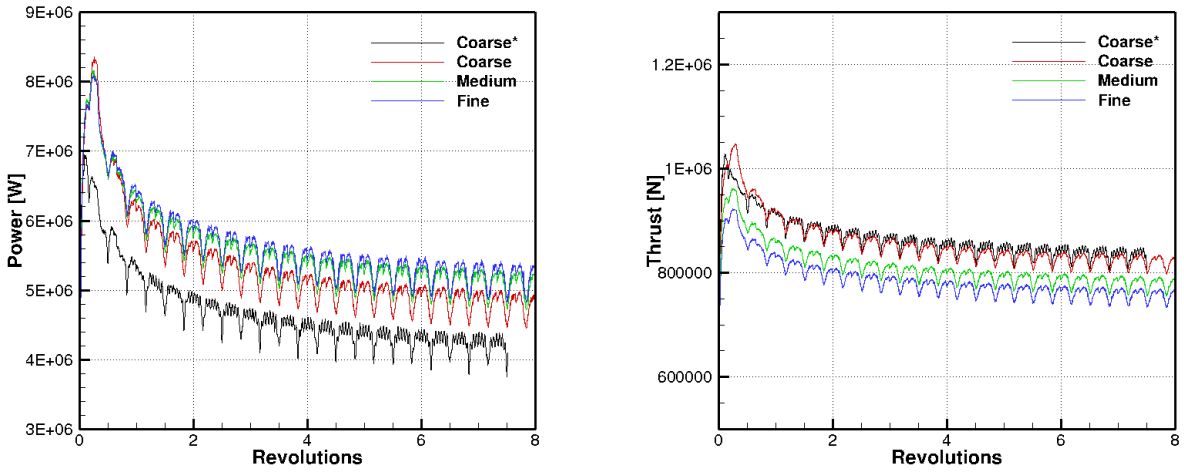


Figure 6.3: NREL 5MW fine mesh: 2,873,862 points.



(a) Power

(b) Thrust

Figure 6.4: NREL 5MW power and thrust simulation results for the mesh resolution study for inflow velocity 11.4 m/s. Each simulation uses a time step corresponding to a 1/4° rotation. Each time step was solved with BDF-2 using 50 sub-iterations for the near-body flow solver.

marginally effected in Fig. (6.4)(b). Highly oscillatory convergence features are also noticed in the coarse* mesh compared to the coarse, medium, and fine meshes. Recalling the coarse, medium, and fine meshes are a family, we see the high frequency content in both the power and thrust curves have the same characteristics in contrast to the coarse* mesh. The low frequency dips in the force histories are caused by the wind turbine blade passing the tower on the downswing of rotation resulting in three dips per revolution.

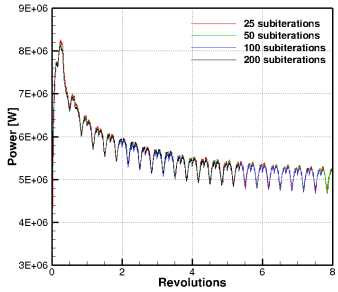
6.1.2 Linear Sub-Iteration Convergence Study

Time-step and sub-iteration convergence studies are performed in Fig. (6.5). Row 1 shows the power prediction histories and row 2 shows the thrust prediction histories. Fig. (6.5)(a) demonstrates the result of using more sub-iterations for the near-body solver in the BDF-2 time stepping scheme, and Fig. (6.5)(b) shows the detailed high frequency content by zooming into the 4-5 revolution time frame. The results indicate using more sub-iterations in the time step smooths the highly oscillatory content in the simulation. Overall the mean values of power prediction using more sub-iterations remain the same as using fewer sub-iterations.

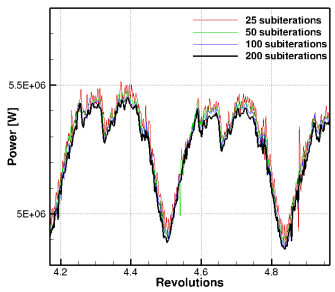
6.1.3 Time Step Convergence Study

In contrast to the sub-iteration convergence study, the time-step study demonstrates a significant influence of the global time step size for force prediction. Fig. (6.5)(c) and (f) show the power and thrust time histories for three sizes of the global time step, respectively. For this study, we choose global time steps corresponding to 1/4°, 1/2°, and 1° of rotor rotation. We can define a local blade tip CFL number as the product of blade tip speed of sound and the global time step divided by the finest mesh element size in the off-body mesh system. The local CFL numbers are 1.02, 2.05, and 4.09 for global time steps corresponding to 1/4°, 1/2°, and 1°, respectively. This local CFL represents the cell distances an acoustic wave can travel in a global time step. Thus for the 1/4° time step, an acoustic wave may travel an entire cell width before the near-body and off-body solutions are updated between the two mesh systems. We see that when we choose a large time step corresponding to a large local CFL, the initial solution transients are higher than using a smaller time step. As the step size is decreased, the solution converges to a refined time-step solution. From Fig. (6.5)(c) and (f), it is suggested that the values of power and thrust will be over predicted unless a sufficiently small global time step of the order of 1/4° is used.

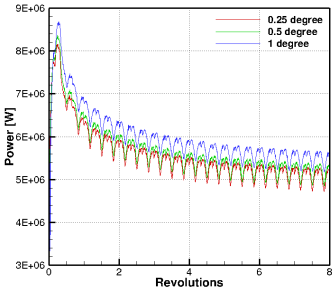
Recall that the data exchange between the two mesh systems is loosely coupled by the global time step, and the receptor solution nodes in each mesh are held at constant values during the sub-iterations/sub-cycling of the time iterations. Particularly for the off-body receptor cells, flow information about the turbine blade is not properly disseminated therefore not 'informing' the donor cells, the cells that are used to fill the near-body receptor cells, to correctly adjust for the moving turbine blades. The average donor cell values will likely be closer to free-stream flow conditions. Thus when the donor cells are used to interpolate the solution from the off-body mesh to the near-body mesh, the receptor cells will receive flow variables containing higher energy content and, hence, result in higher force prediction.



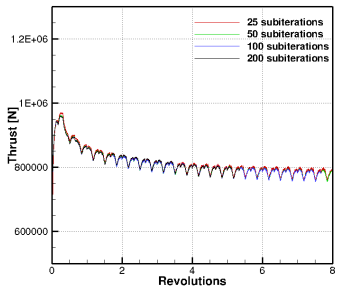
(a) Power convergence history for 25-200 sub-iterations with time step corresponding to 1/4° rotation.



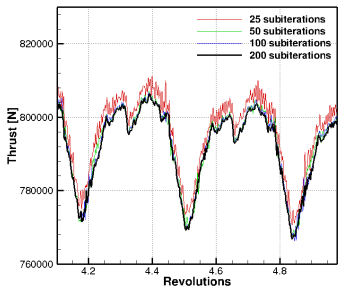
(b) Zoomed view of power history at 4-5 revolutions with time step corresponding to 1/4° rotation.



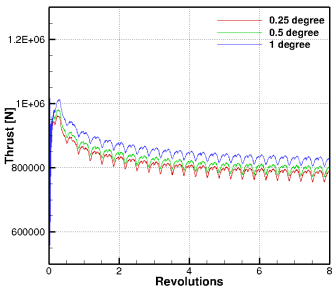
(c) Power history for time step sizes corresponding to 1/4°, 1/2°, and 1° of rotation using 50 sub-iterations.



(d) Thrust convergence history for 25-200 sub-iterations with time step corresponding to 1/4° rotation.



(e) Zoomed view of thrust history at 4-5 revolutions with time step corresponding to 1/4° rotation.



(f) Thrust history for time step sizes corresponding to 1/4°, 1/2°, and 1° of rotation using 50 sub-iterations.

Figure 6.5: NREL 5MW force histories using BDF-2 time stepping for the near-body flow solver. All results performed on the medium refined mesh are presented in Table (6.1).

Fig. (6.6) shows the power and thrust predictions for inflow velocities 6-11.4 m/s using the medium mesh compared to the NREL FAST [172] reference solution. The reference solution uses blade element momentum theory with a fluid-structure interface to model structural wake effects [172]. The power predicted from W^{2}A^{2}KE3D agrees well with FAST, and the thrust is slightly under predicted for inflow velocities less than the nominal inflow velocity of 11.4 m/s. We note as the velocity increases, the power becomes slightly under predicted using the medium refined mesh but notice the power is improved when using the fine mesh. We also note that the W^{2}A^{2}KE3D framework results do not contain blade elastic structural deflection responses.

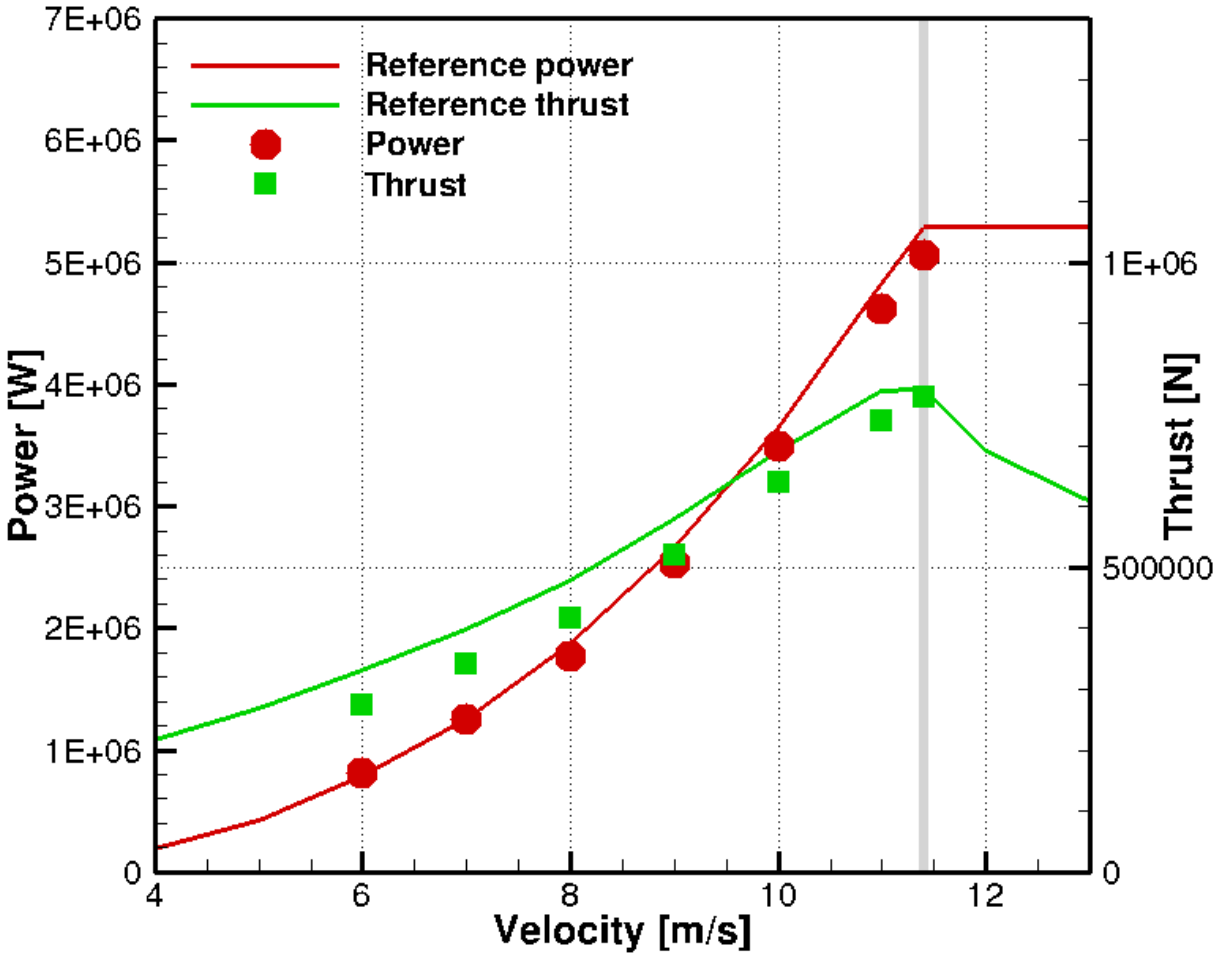


Figure 6.6: NREL 5MW power and thrust simulation results using a time step corresponding to a 1/4° rotation. Each time was solved with BDF-2 using 50 sub-iterations for the near-body flow solver on the Medium mesh. Reference solution data provided by the NREL FAST software.

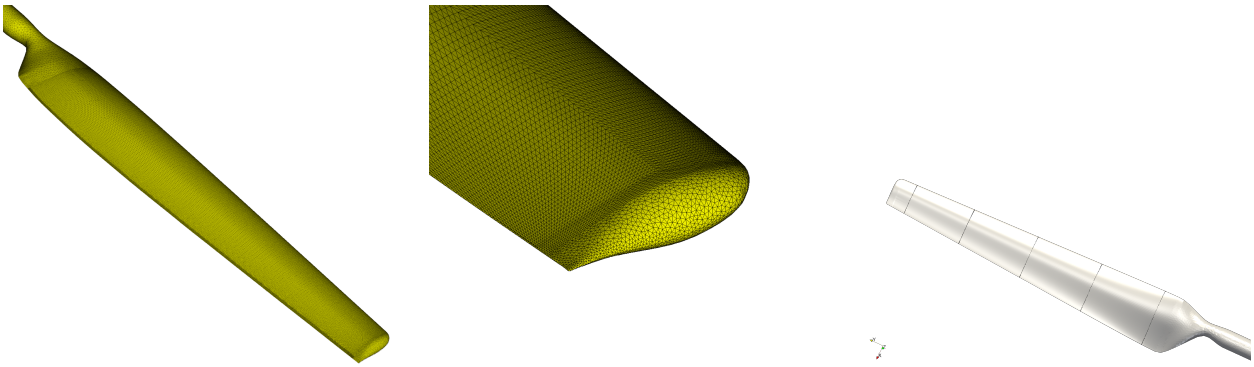


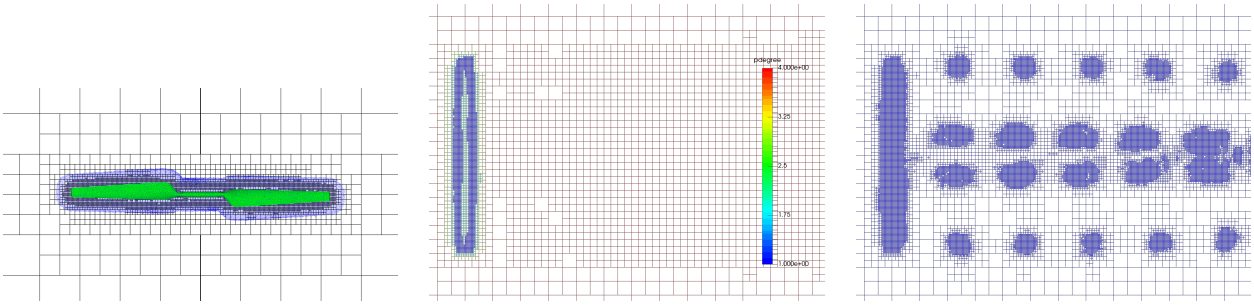
Figure 6.7: NREL Phase VI computational near-body mesh containing 7 million elements and 3 million nodes. The right figure shows the span-wise stations used for pressure coefficient measurements.

6.2 NREL Phase VI

The NREL Unsteady Aerodynamics Experiment Phase VI is a wind turbine that has been studied experimentally [173–176]. The wind turbine was studied at NASA Ames Research Center in the 80 ft x 120 ft (24.4 m x 36.6 m) wind tunnel. The experiment of the Phase VI wind turbine is regarded as one of the most extensive studies performed for a wind turbine.

The Phase VI turbine has a blade radius of 5.029 m and the rotor is assumed to be rigid with a blade pitch angle of 3°, a yaw angle of 0°, and a cone angle of 0° for this computational study. The blade geometry is constructed from a single NREL S809 airfoil [173]. The rotation rate is prescribed at 72 revolutions per minute. The tower and nacelle are excluded for this case. The near-body mesh used in this simulation contains approximately seven million elements and three million nodes which extend one chord length from the surface of the blade. Fig. (6.7) shows the near-body surface mesh and Fig. (6.8)(a) depicts the near-body and off-body mesh system.

The inflow conditions vary with velocities ranging from 7-15 m/s and a Reynolds number of 2.5 million based on the chord length of the wind turbine blade. The global time step is set to `1/4`° of rotation, and the near-body flow solver uses 25 sub-iterations per BDF-2 time step. The off-body mesh domain is 1000 m with the mesh system composed of 11 levels of refinement. The finest AMR level uses $p = 1$, second-order spatial accuracy and the coarser levels transition to $p = 4$, fifth-order spatial accuracy elements. To do this transition,



(a) Near-body mesh trimmed to one chord length shown in blue with the surface of the blade shown in green.

(b) Off-body mesh adaption colored by spatial solution order of accuracy with $p = 1$, 2nd-order, near the wind turbine and growing to $p = 4$, 5th-order accuracy.

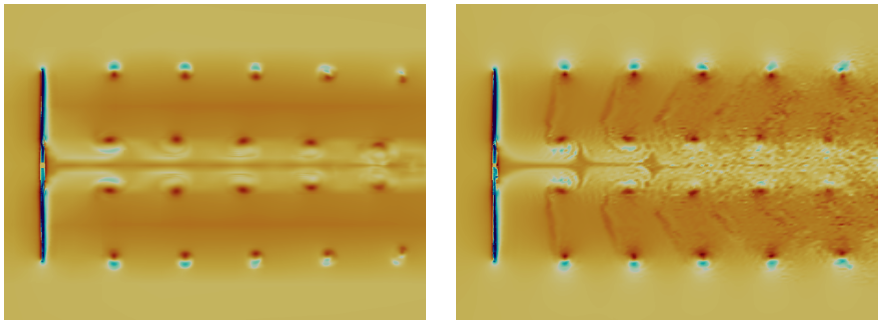
(c) Off-body mesh adaption of $p = 1$, second-order, spatial solution order of accuracy.

Figure 6.8: NREL Phase VI overset mesh system with wake mesh adaption.

each subsequent level from the finest mesh level increases its spatial order of accuracy by one polynomial degree until the maximal polynomial degree of $p = 4$ is achieved. For this particular study, level 11 uses $p = 1$, level 10 uses $p = 2$, level 9 uses $p = 3$, and all coarser levels use $p = 4$. Thus higher order spacial accuracy is used in the regions away from the unstructured mesh particularly in the wake region. Fig. (6.8)(b) demonstrates the spatial order of accuracy in different regions of the off-body mesh showing fewer elements are needed in the wake region when higher polynomial degrees are used. Fig. (6.8)(c) shows the difference in adaptive mesh refinement when only $p = 1$, second-order accurate elements are used.

Fig. (6.9) demonstrates the use of second-order spatial accuracy in the wake region in comparison to fifth-order spatial accuracy. The fifth-order accurate solution is able to capture finer turbulence scales whereas the second-order accurate solution smears the details of the wake structure. The fifth-order accurate solution requires three less AMR levels in the wake region therefore reducing the overall element count.

Fig. (6.10) shows the power and thrust predictions of W^2A^2KE3D. Good agreement for velocities of 7-10 m/s is demonstrated for all flow solvers in comparison to experimental data. For velocities 11-15 m/s, delayed blade stalling is present in the overset simulations in comparison to the HELIOS solver and experimental data resulting in over prediction of the power and thrust. However, the W^2A^2KE3D results for power are significantly more



(a) Velocity mag. using $p = 1$. (b) Velocity mag. using $p = 4$.

Figure 6.9: NREL Phase VI wake comparison of 2nd- and 5th-order spatial discretizations.

accurate than the stand-alone OVERFLOW and NSU3D results.

Fig. (6.11) shows the pressure coefficient at $30\%, 46.6\%, 63.3\%, 80\%$ and 95% span-wise stations of the blade for 7, 10, and 15 m/s uniform inflow velocities. In all inflow velocity cases at all sectional locations, the computed pressure coefficient values on the pressure side of the blade compare well with experimental data, as expected. On the suction suction side of the blade, good agreement with experimental data is observed at 7 m/s. However for higher inflow velocity cases, some of the experimental values show flat profiles indicative of blade stalling, while the computational results show higher suction peaks suggesting the flow remains attached. The discrepancy is most pronounced at 46.6% span for the 10 m/s inflow velocity case, and at the 30% span location for the 15 m/s inflow velocity case. Fig. (6.12) shows span-wise slides of the computed pressure coefficient for 11 m/s illustrating the fact the the flow remains mostly attached at this condition.

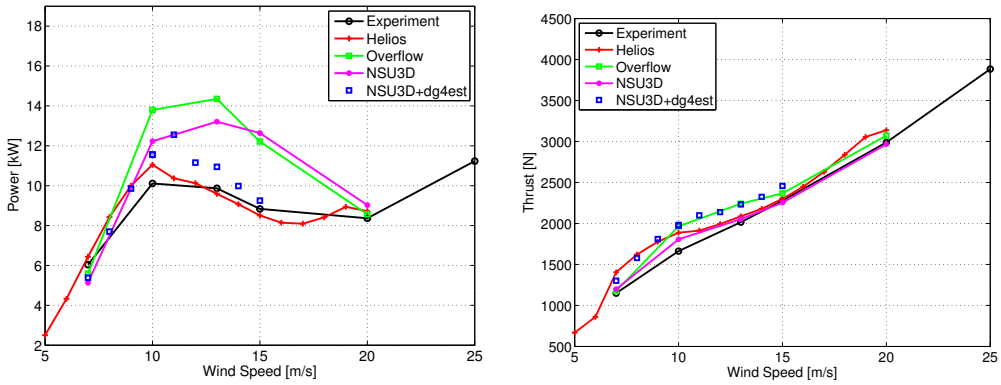


Figure 6.10: NREL Phase VI power and thrust for uniform inflow velocities of 7-15 m/s. Results are compared to the experimental values along with other numerical simulations: NSU3D in stand-alone, CREATE-AV HELIOS, and NASA Overflow.

6.3 Siemens SWT-2.3-93

A generic Siemens SWT-2.3.93 turbine model using specifications from the IAE Wind Task 31-Wakebench [177] is simulated. The geometry of the turbine blade is constructed from multiple cylinder and airfoil sections. The wind turbine contains three blades and a tower with a nacelle for a total of four near-body meshes per wind turbine. The nominal rotor rotation speed is 16 revolutions per minute. The rated power inflow velocity is 10.9 m/s generating a rated electrical power of 2.3 MW. The Siemens blade has a radius of 46.5 m and a low-speed shaft title angle of 6°, pre-cone angle of 2.5°, and nominal blade pitch of -1°. The tower has a height of 65 m. To accurately predict the power and thrust forces, as indicated by the mesh refinement study of the NREL 5MW wind turbine, a near-body blade mesh containing 2,219,940 nodes and a tower with nacelle mesh containing 504,960 nodes is used. The global time step is set to a corresponding rotation of `1/4`° and 25 non-linear sub-iterations are used to converge the BDF-2 time step for the near-body solver.

Our predicted power from the simulation framework at the nominal uniform inflow velocity of 10.9 m/s is 2.5 mega-watts and the torque is predicted at 373,000 Newtons which agrees well with the NREL FAST software (2.5 mega-watts is the aerodynamic power force before losses due to the generator which yields 2.3 mega-watts). The power and thrust convergence histories are shown in Fig. (6.13). Fig. (6.14) shows a volume rendering and an iso-surface visualization of the Siemens SWT-2.3-93 wind turbine.

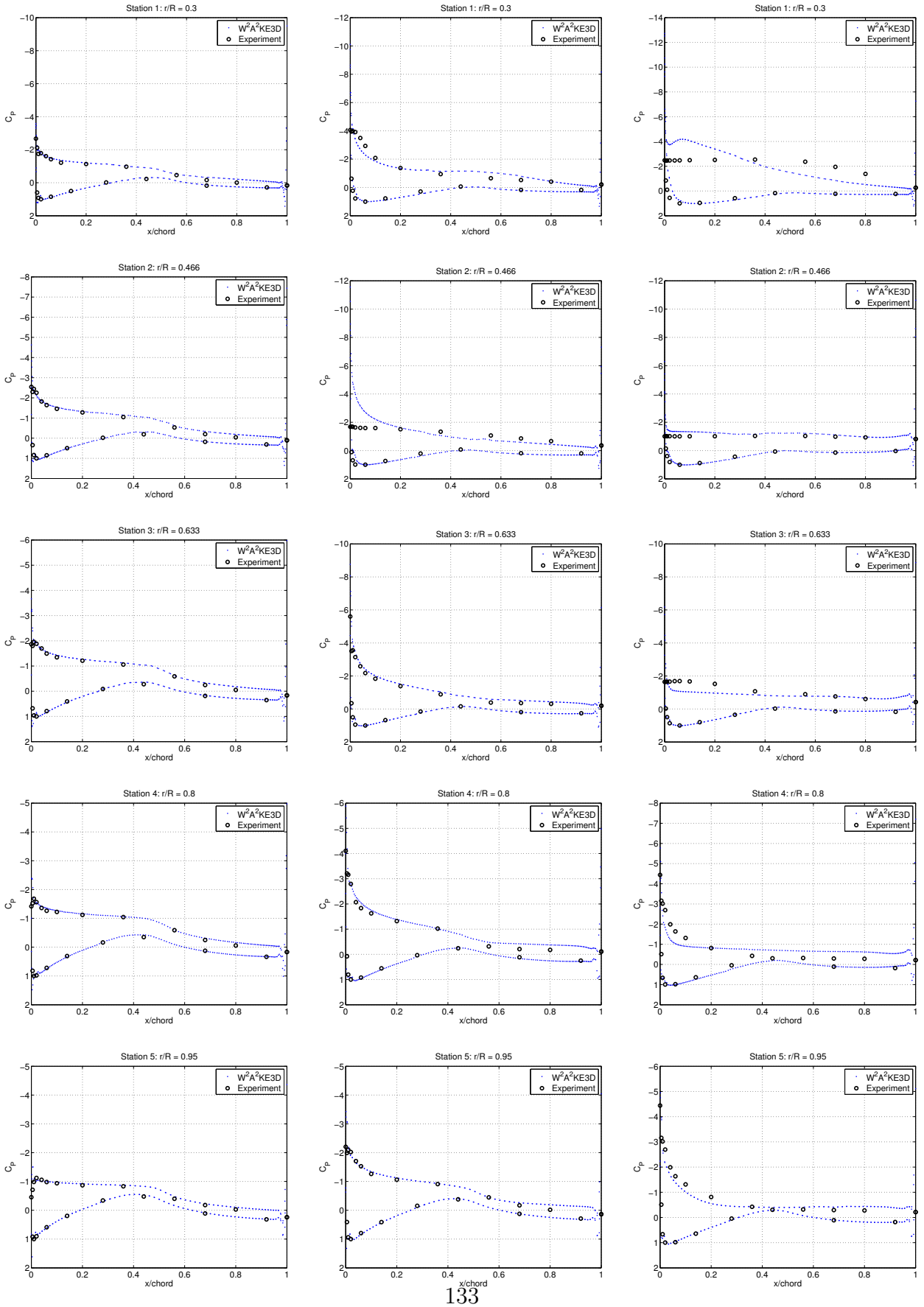


Figure 6.11: NREL Phase VI pressure coefficients at 30%, 46.6%, 63.3%, 80%, 95% spanwise stations for 7 m/s (column 1), 10 m/s (column 2), and 15 m/s (column 3) uniform axial inflow velocities. Predicted results of W^2A^2KE3D are plotted versus the experimental data.

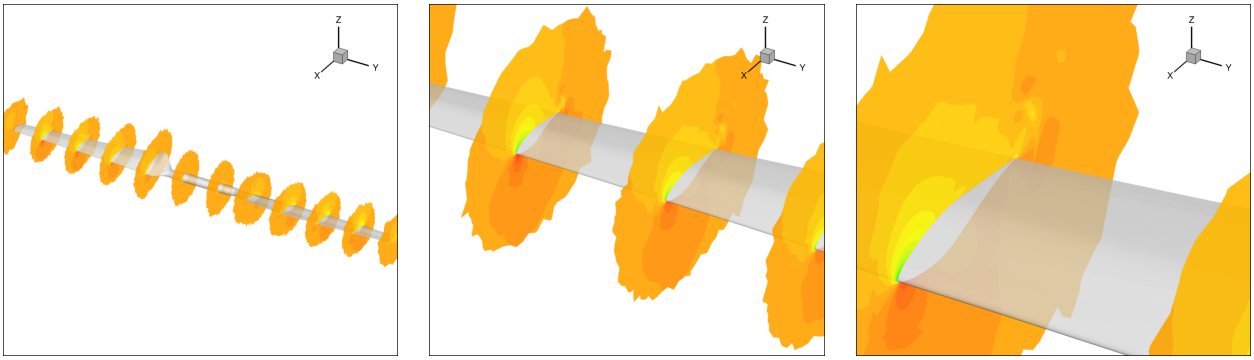


Figure 6.12: NREL Phase VI coefficient of pressure visualized for 11 m/s inflow velocity.

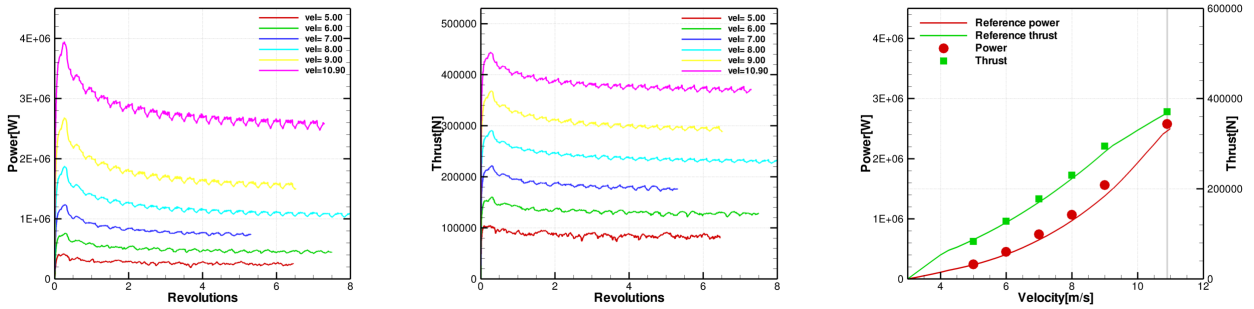
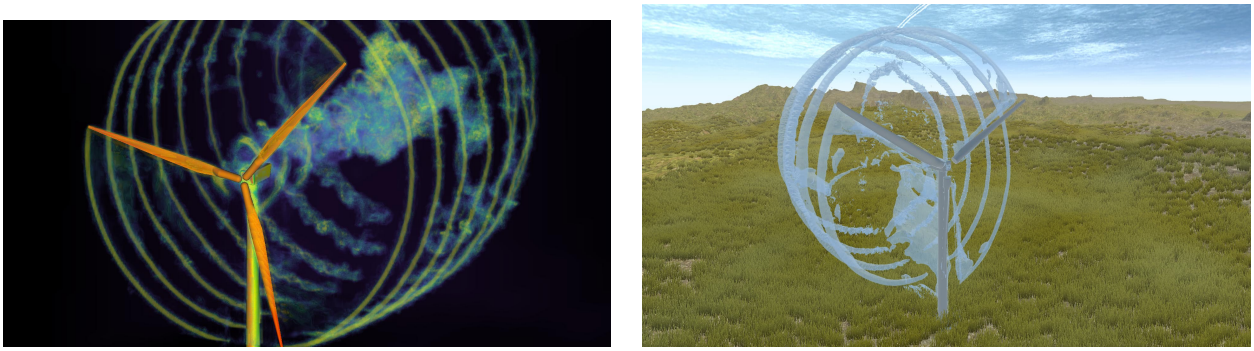


Figure 6.13: Siemens SWT-2.3-93 power and thrust simulation results using a time step corresponding to a 1/4° rotation. Each time was solved with BDF-2 using 25 sub-iterations for the near-body flow solver. Reference solution data provided by the NREL FAST software.



(a) Volume rendering of vorticity.

(b) Iso-surface of velocity magnitude.

Figure 6.14: Siemens SWT-2.3-93 wind turbine.

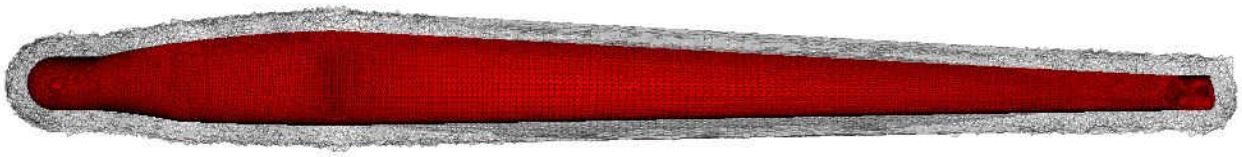


Figure 6.15: NREL WindPACT-1.5MW unstructured blade mesh with 3.24 million nodes.

6.4 WindPACT-1.5MW

This problem studies a 1.5MW wind turbine from the NREL Wind Partnership for Advanced Component Technologies [178] (WindPACT) Project, which was used for a turbine rotor study [179]. The WindPACT-1.5MW wind turbine is a three-bladed design with a rotor diameter of 70 m, rotation rate of 20.5 revolutions per minute, and an axial velocity of 10.7338 m/s corresponding to a tip-speed ratio of $\lambda = 7.0$. A blade pitch angle of 2.6° is applied.

The rotor geometry is composed of three identical unstructured blade meshes. The unstructured blade mesh is trimmed to approximately 1 meter from the blade surface, as shown in Fig. (6.15). The computational blade mesh is composed of approximately 3.24 million nodes with 687,965 tetrahedra, 49,061 pyramids, and 6,150,915 prisms. The smallest element width is 4.937 microns. The total rotor geometry mesh aggregates to approximately 9.72 million nodes. Each blade mesh is partitioned onto 144 cores, giving a total of 432 cores for the near-body solver.

The near-body mesh flow solver uses a Delayed Detached Eddy Simulation [152] turbulence model, and the off-body adaptive Cartesian mesh solver uses a Constant Smagorinsky Large Eddy Simulation [89] turbulence model. The off-body discontinuous Galerkin flow solver uses $p = 1$, second-order, polynomials in mesh cells near the blade surface and transitions to $p = 3$, fourth-order, polynomials in mesh cells away from the surface with a layer of $p = 2$ mesh cells in between to smoothly transition the solution. The flow solvers use a global time step corresponding to 1/3° rotor rotation, which is followed by flow solution interpolation between the near-body and off-body meshes performed by the overset mesh assembler. The near-body solver performs implicit time steps using the BDF-2 method, and the off-body uses multiple explicit time steps using the RK4 method.

Table 6.2: Single wind turbine data reductions obtained via in-situ workflow. A total of 72,008 cut-planes were written over 50 rotor revolutions in place of 9001 volume data files.

Volume Files	Volume Size (GB)	Cut-Planes	Cut-Plane Size (GB)	Data Reduction
9001	235,955.4	72,008	960	246.2x

The simulation is performed without the presence of the wind turbine tower or nacelle. The simulation conditions use uniform inflow, with the fluid parameters set to an ideal fluid (air) of density 1.225 kg/m^3, and a kinematic viscosity of $1.5 \cdot 10^{-5} m^2/s$. The simulation is performed for 50 rotor revolutions starting with 600,000 degrees-of-freedom in the off-body mesh and grows to over 600 million as the mesh adapts to the wake. The number of cores used for the off-body solver grows to 10,800.

The off-body volume data grew linearly over the simulation starting at a size of 629 MB and ending at 51.8 GB. If the volume data had to be output at the same 2° rotor revolution frequency over 50 rotor revolutions, a total of 235,955.4 GB (230.4 TB) of data would have been produced. Alternatively, using in-situ analysis outputting seven cut-planes at 0.0095 GB per plane and one center cut-plane at 0.04 GB per plane resulted in a total of 960 GB of data. This results in a data reduction factor of 246. Table (6.2) summarizes the results of the in-situ data reduction.

6.4.1 Analysis Approach

Simulation data from the high-fidelity numerical method is collected for 16 rotor revolutions starting at the time step corresponding to the beginning of the 31^{st} rotor revolution, with a temporal resolution based on a time step corresponding to 2° of revolution. This results in a total of 2,880 temporal samples. The spatial samples are taken from two-dimensional cross-wake planes at downstream stations shown in Fig. (6.16), where D denotes rotor diameter. Each plane has a spatial resolution of 400 by 400, which corresponds to a dimensional resolution of 40 cm by 40 cm.

The CFD simulation results are output in the Cartesian coordinate system. Since the simulation does not contain a tower or nacelle, and the inflow is uniform and perpendicular to

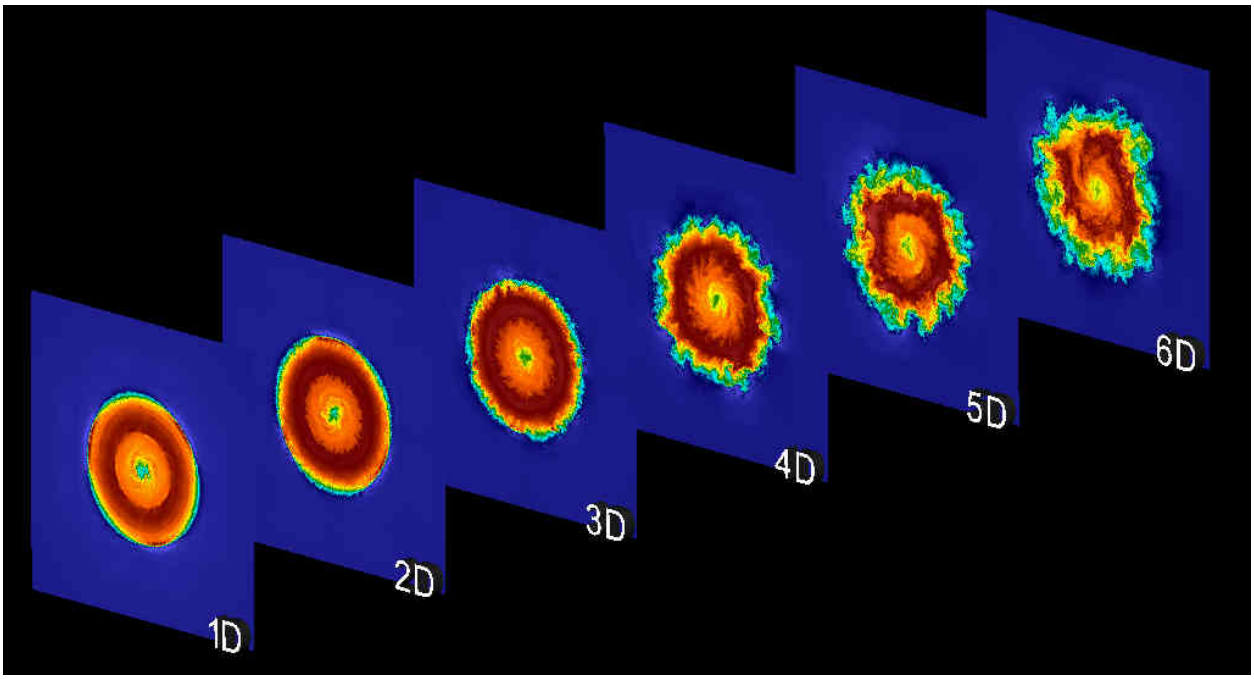


Figure 6.16: Instantaneous axial momentum at multiple downstream positions of the NREL WindPACT-1.5MW wind turbine.

the rotor plane, the flow is axisymmetric. Since the flow is axisymmetric, a coordinate transformation is applied to the simulation results, and analysis is applied in the polar coordinate system. In the polar reference frame, the velocity vector is composed of axial (denoted U), radial (denoted V), azimuthal (denoted W) components. All results and analysis presented herein are in the polar coordinate system.

6.4.2 Results

Fig. (6.16) demonstrates instantaneous axial momentum at multiple downstream stations as a function of rotor diameter (D). Qualitatively, the wake deficit structure is highly regular in the near-wake region starting at the turbine to two rotor diameters (2D) downstream. At station 3D, the wake begins to break down and transition to turbulence as shown in Fig. (6.17), where an instantaneous isocontour of velocity magnitude of 8.5 m/s colored by density demonstrates the tip vortex structure evolution. Between stations 2D and 3D, vortex merging and hopping occur implying instability of the vortex structures. Fig. (6.18) shows an instantaneous flow visualization of the normalized absolute tangential velocity wake.

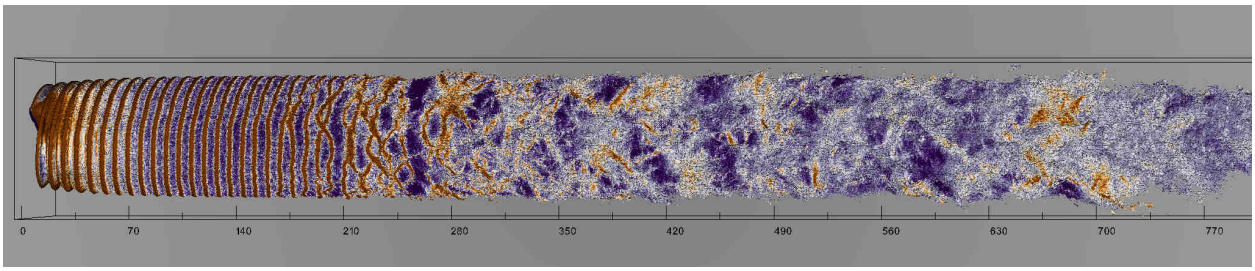


Figure 6.17: Instantaneous isocontour of the velocity magnitude of 8.5 m/s colored by density demonstrating the vortex structure evolution of the NREL WindPACT-1.5MW wind turbine.

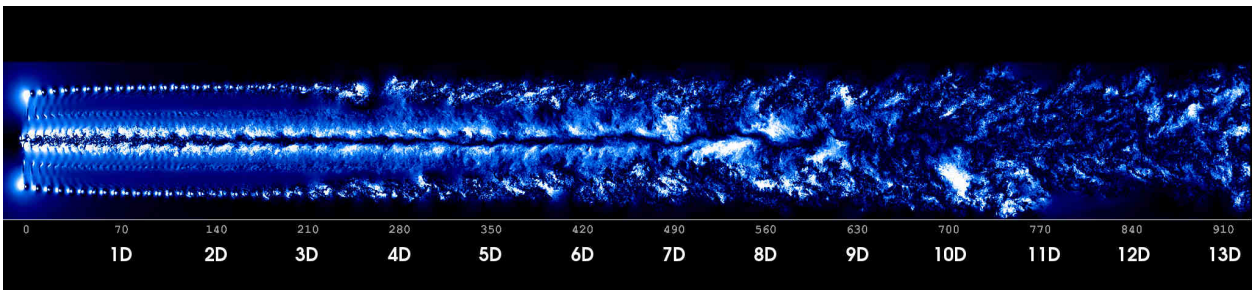


Figure 6.18: Instantaneous normalized absolute tangential flow velocity demonstrating the wake propagation downstream, annotated by rotor diameter lengths (D).

Fig. (6.19) shows the instantaneous polar velocity components: axial (U), radial (V), and azimuthal (W). The figure shows the flow structures as a function of downstream wake position. The radial velocity is highest near the blade tip regions, whereas the azimuthal velocity is uniformly distributed from the root region out to the blade tip indicating the flow axial induction. Fig. (6.20) demonstrates the time-averaged axial velocity at station x/D = 0.5. The temporal averaging occurred over 16 rotor revolutions of data. The wake is bordered by a sharp transition zone distinguishable by a thin annular area with steep velocity gradient, the shear region. Velocities in the center of the wake are slightly higher since no hub is modeled, and the overall shape of the wake is symmetric, as the turbine tower is also not modeled and the inflow velocity is uniform.

The axial velocity component is determined by a significant velocity deficit caused by the turbine, which gradually recovers to the incoming wind velocity by moving downstream as demonstrated in Fig. (6.21)(a). The wake recovery, which is strongly influenced by turbine performance [180] and the incoming turbulent flow [181], is an important feature for the estimation of the turbine separation distance within a wind farm. The largest velocity deficit

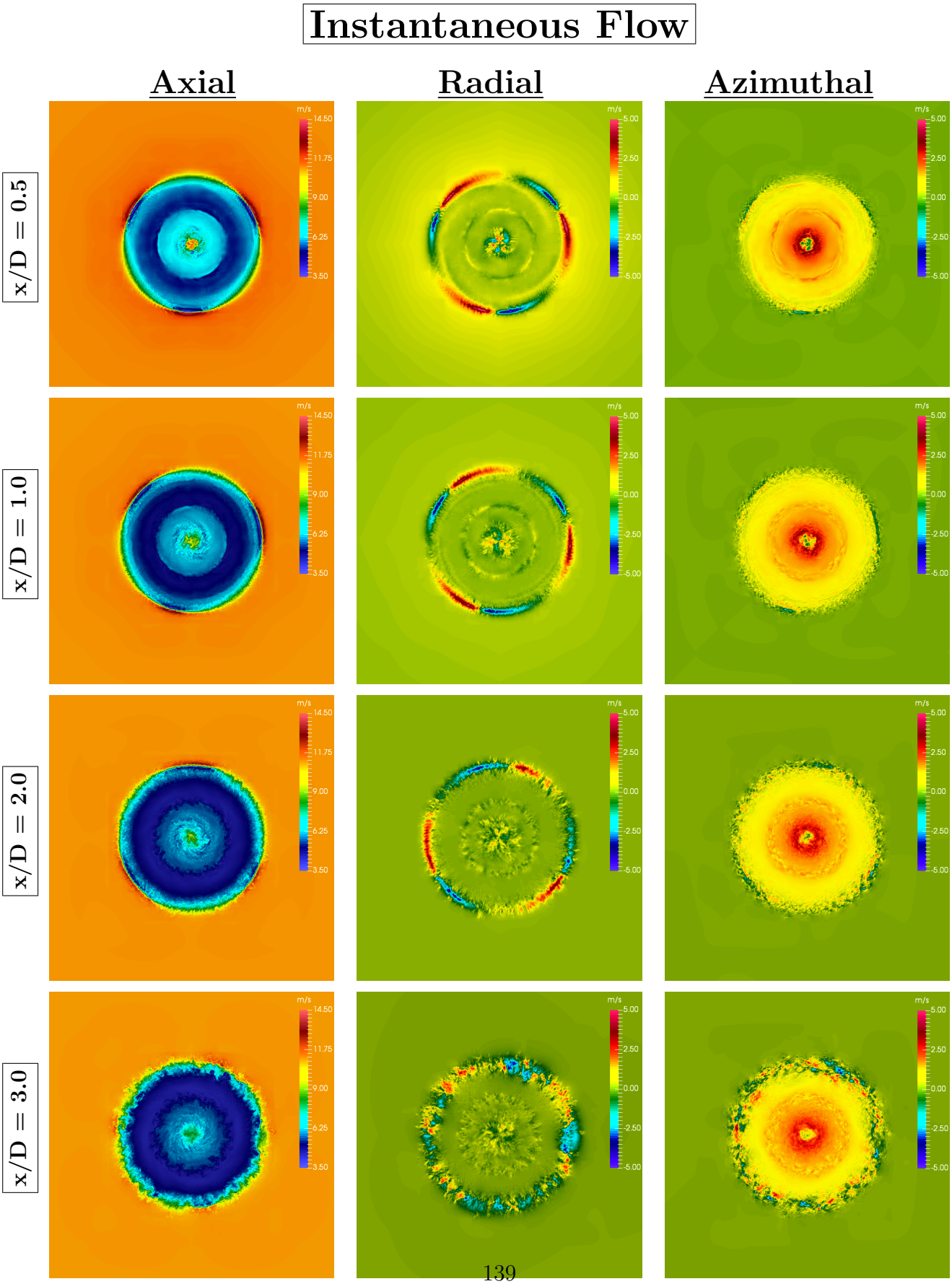


Figure 6.19: Instantaneous axial (U), radial (V), and azimuthal (W) velocity components at downstream wake positions: 0.5, 1.0, 2.0, and 3.0 rotor diameters (D).

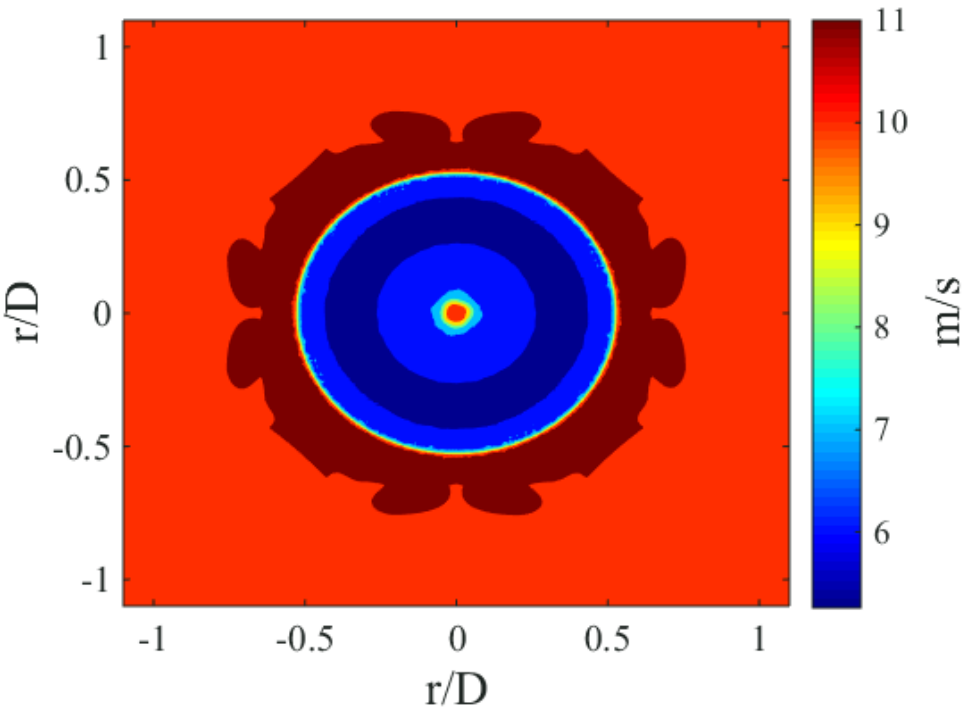
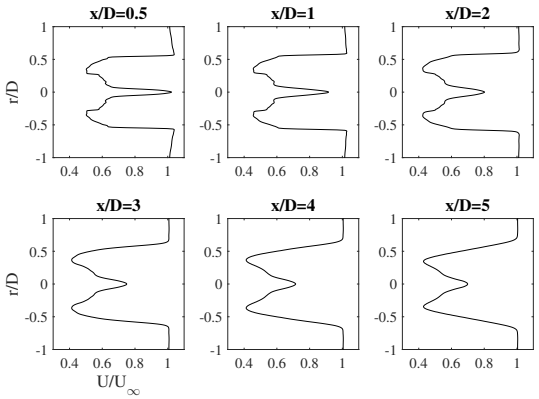


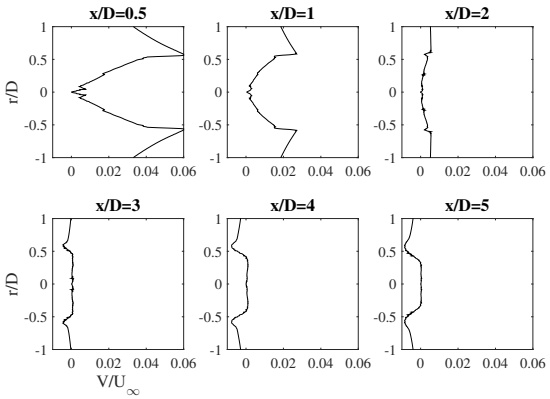
Figure 6.20: Temporally averaged axial velocity at x/D= 0.5 over 16 rotor revolutions.

is found at the blade tip, r/D = 0.5, where more energy is captured from the flow [180]. Further downstream, this region moves gradually towards the center of the wake. A general radial velocity component near the wake center is observed, which is in agreement with measurements by Medici [182].

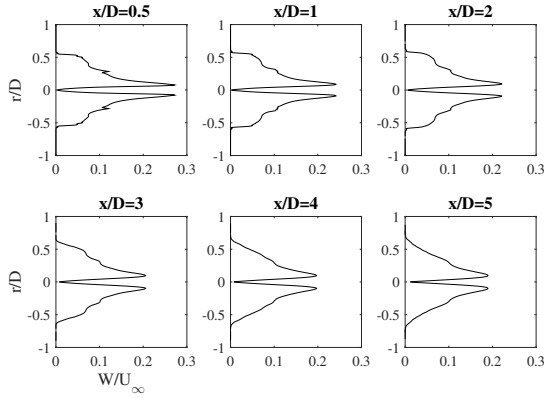
However, as seen in Fig. (6.21)(b), found by Medici [182] as well, a radial velocity component emerges from the center out to the freestream, which is caused by the centrifugal force applied to the flow in the rotor plane. The radial velocity component reverses towards the center approximately at x/D = 2.0 due to the entrainment from the freestream velocity to the wake. A significant peak of the azimuthal velocity is detected for radial positions r/D = 0.1 shown in Fig. (6.21)(c), which is related to the rotational velocity induced by a vorticity structure created in the blade root region. The azimuthal velocity diffuses very slowly as it moves downstream. A feathering is observed for the azimuthal velocity at the r/D = 0.5, influenced by the presence of the tip vortices, which gradually diffuse at downstream distances of approximately x/D = 4.0.



(a) Axial Velocity Deficit

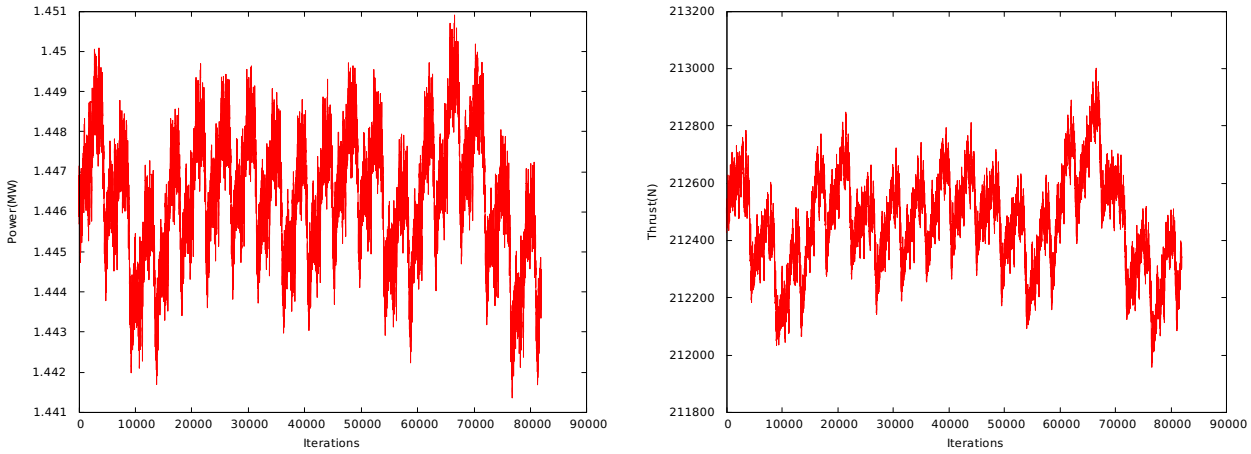


(b) Normalized Radial Velocity



(c) Normalized Azimuthal Velocity

Figure 6.21: Time-averaged wake velocity profiles normalized by the freestream velocity at different downstream locations.



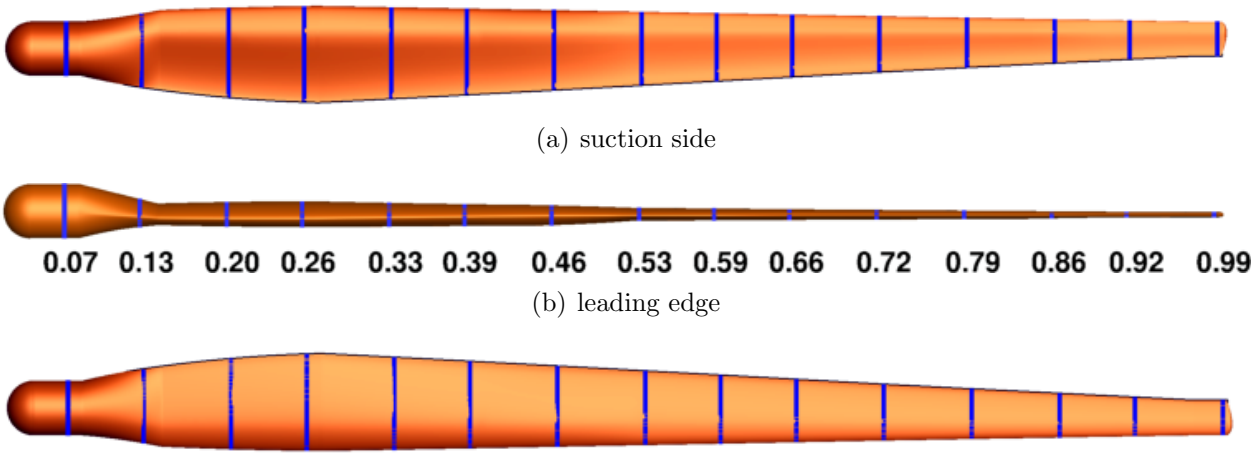
(a) Power (b) Thrust

Figure 6.22: Power and thrust prediction of the WindPACT-1.5MW wind turbine.

6.4.3 Blade Analysis

In this section, first-order statistics are analyzed at the wind turbine blade region. Fig. (6.22) demonstrates time history of the calculated dimensional power and thrust for the WindPACT-1.5MW wind turbine. A lower-fidelity model NREL FAST [183], which couples in an actuator line method, predicts 1.575 MW of power and 221,700 N of thrust. The current work predicts the power to be approximately 1.447 MW and 212,400 N of thrust.

Fig. (6.23) indicates the measurement stations used for blade loadings and coefficient of pressure measurements, which also illustrates the blade geometry. Fig. (6.24) shows the normal (F_n), radial (F_r), tangential (F_θ) loading forces on the blade as a function of normalized radial position. The radial blade forcing is significantly smaller in magnitude than the azimuthal and normal forces. The profile of the radial force transitions from a negative force to a positive force while traversing the first third of the blade radius. As seen in Fig. (6.23), the second station at r/R = 0.13 is positioned on transitioning geometry resulting in a spike in the radial force. The azimuthal force quickly ascends to a plateau from 20% to 90% of the blade radius then quickly falls off. Lastly, the normal blade loading linearly grows as a function of radius and then sharply falls resulting from the generated blade tip vortex. Note that the maximum normal axial force is approximately 8x the maximum azimuthal force and is 80x larger than the maximum radial force.



(a) suction side

(b) leading edge

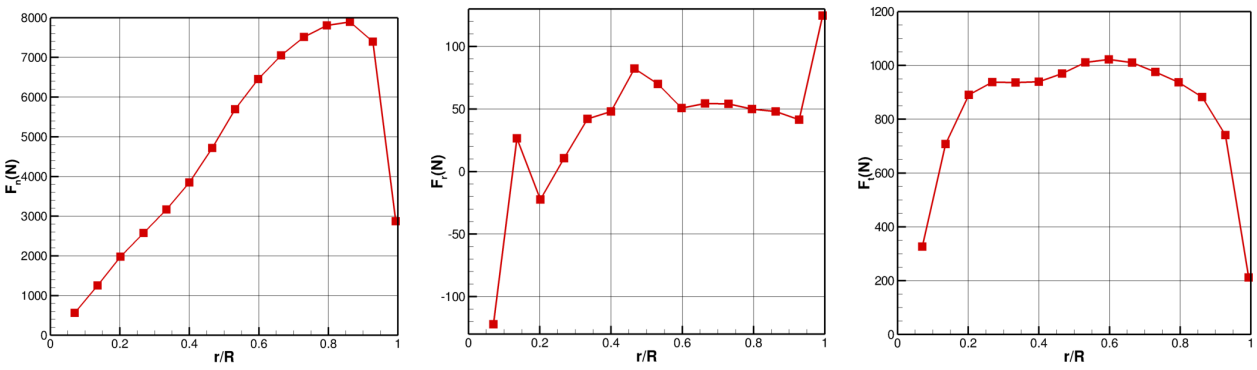
(c) pressure side

Figure 6.23: Measurement locations for loading forces and coefficient of pressure.

Fig. (6.25) shows the computed pressure coefficient on the blade surface, and Fig. (6.26) shows the individual station measurements. The sectional coefficient of pressure (C_p) is calculated as follows:

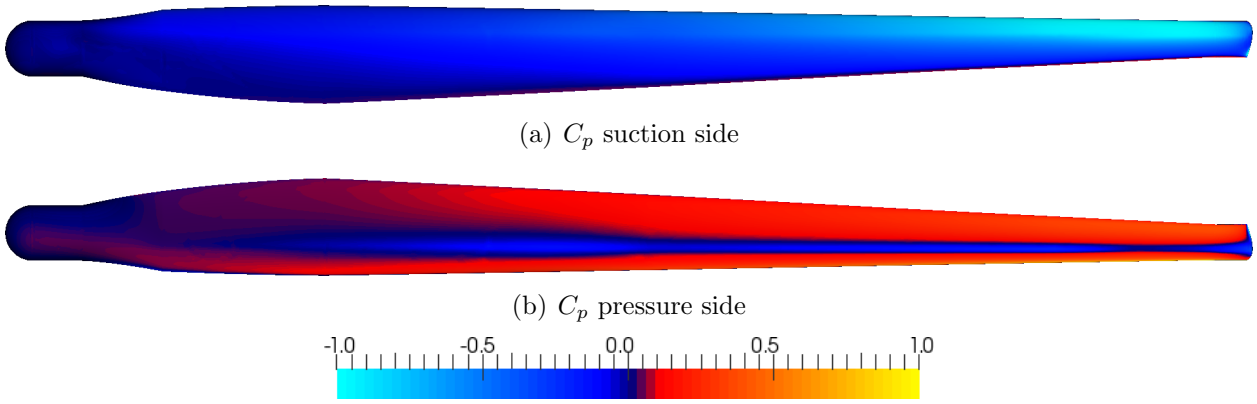
$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty \left(U_\infty^2 + (r\omega)^2\right)} \tag{6.1}$$

where ω is the rotation speed, and r is the sectional radius. The C_p plots indicate well-behaved values along the span of the blade. However, as seen in Figs. (6.25)(b) and (6.26), the pressure side of the wind turbine blade has a region spanning the entire blade where the C_p is negative just behind the leading edge indicating a favorable pressure gradient then transitioning into an adverse pressure gradient. Fig. (6.27) illustrates a blade-tip view of the pressure gradients.



(a) F_n (b) F_r (c) F_θ

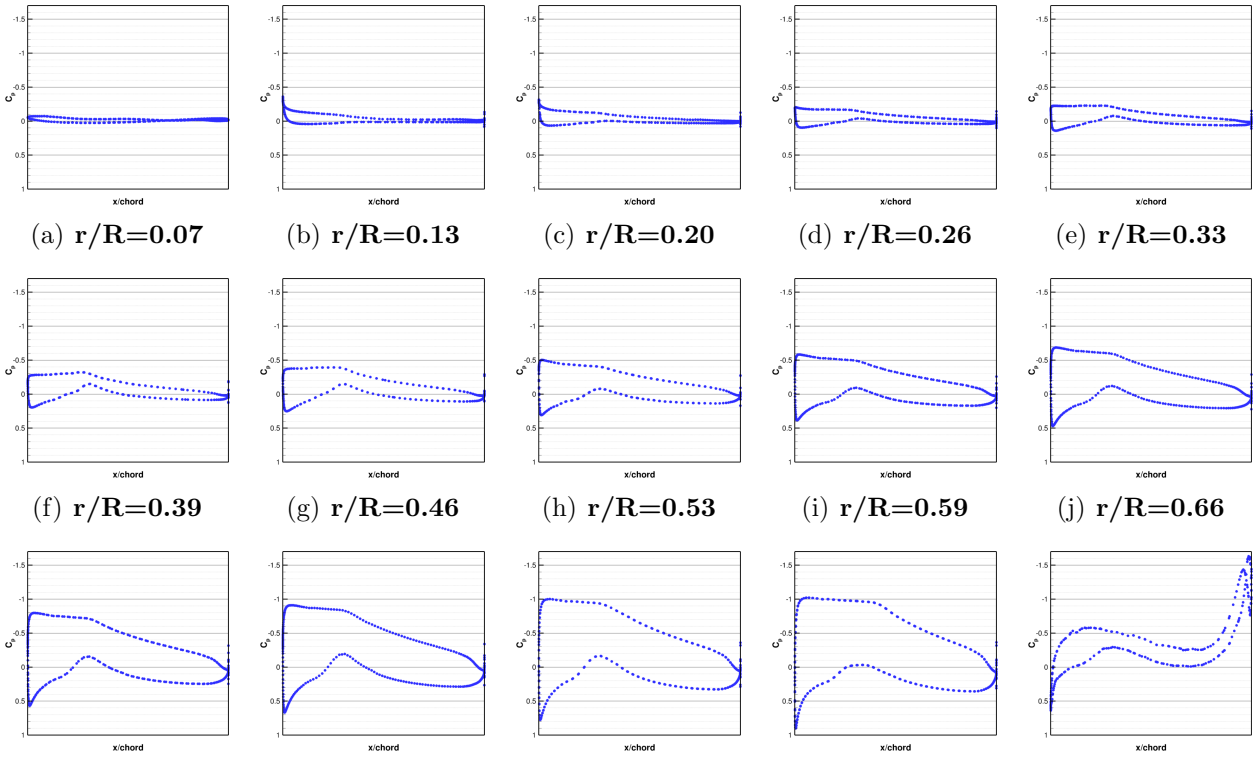
Figure 6.24: Normal [axial] (F_n), radial (F_r), and azimuthal (F_θ) force components distributed along the normalized blade radius.



(a) C_p suction side

(b) C_p pressure side

Figure 6.25: Coefficient of pressure on the blade surface. Pressure gradients are present on the pressure side along the span of the wind turbine blade.



(a) **r/R=0.07** (b) **r/R=0.13** (c) **r/R=0.20** (d) **r/R=0.26** (e) **r/R=0.33**

(f) **r/R=0.39** (g) **r/R=0.46** (h) **r/R=0.53** (i) **r/R=0.59** (j) **r/R=0.66**

(k) **r/R=0.72** (l) **r/R=0.79** (m) **r/R=0.86** (n) **r/R=0.92** (o) **r/R=0.99**

Figure 6.26: Coefficient of pressure at stations along the blade normalized blade radius.

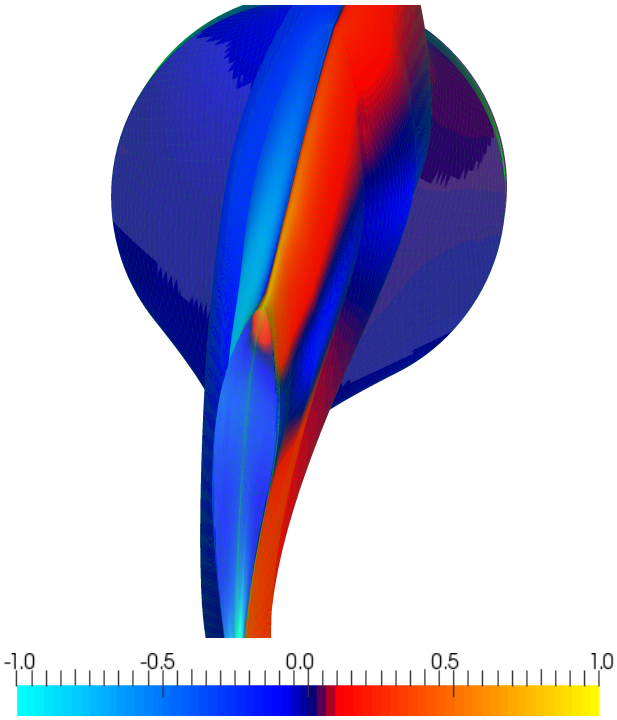


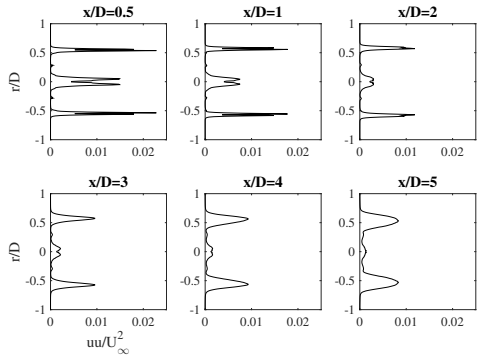
Figure 6.27: Blade tip view of coefficient of pressure showing the pressure gradients spanning the length of the wind turbine blade.

6.4.4 Reynolds Stress Analysis

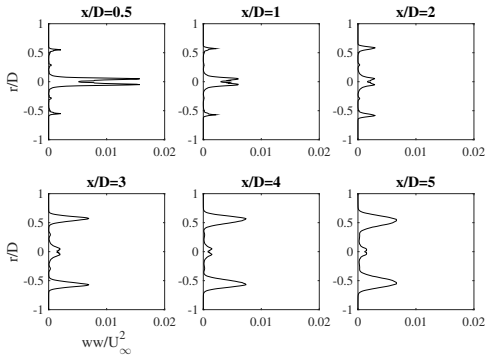
In this section, the wake is analyzed in terms of turbulence and Reynolds stresses. First, we denote the instantaneous velocity as $U(t)$ at $t = t_k$ and the temporally-averaged velocity as $\overline{u}$. The Reynolds stresses are averaged temporally over $n = 2,880$ time instances corresponding to 16 rotor revolutions. The Reynolds stresses are calculated as follows:

$$u_i u_j = \frac{\sum_{k=1}^{n} \left(U_i(t_k) - \overline{u}_i\right)\left(U_j(t_k) - \overline{u}_j\right)}{n}, \quad u_i = \{u, v, w\} \tag{6.2}$$

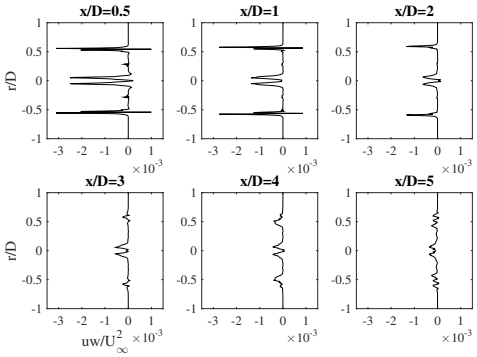
The normal Reynolds stress uu/U_∞^2, plotted in Fig. (6.28)(a), has an increased intensity in the near-wake, representing the mechanically-produced turbulence due to the presence of the wake velocity deficit. This wake turbulent energy gradually dissipates propagating downstream, while the wake diffuses and increases its transverse width. In addition, velocity fluctuations associated with the tip vortices are also detected from the normal stress for locations x/D = 0.5 and 1.0. However, a small increase occurs in the normalized axial stress at x/D = 4.0 and 5.0, which is due to the divergence of the second-order moments locally.



(a) Reynolds normal stress, uu/U_∞^2



(b) Reynolds normal stress, ww/U_∞^2



(c) Reynolds shear stress, uw/U_∞^2

Figure 6.28: Normalized Reynolds stresses at multiple downstream locations.

Conversely, in Fig. (6.28)(b), the normalized stress connected to the azimuthal velocity component, ww/U_∞^2, corresponding to the outboard of the blade increases downstream. However, the azimuthal normal stress associated with the blade root drops very rapidly in the near wake. Lastly, Fig. (6.28)(c) shows the Reynolds shear stress, uw/U_∞^2. This is an important quantity as it is related to the vertical transport of momentum, with negative values of shear indicating entrainment of the freestream flow momentum into the wake, which directly relates to the re-energizing process of the flow.

6.4.5 Proper Orthogonal Decomposition Analysis

Proper Orthogonal Decomposition (POD) is a common mathematical analysis technique known by other names from other fields (e.g. Principal Component Analysis) and is closely related to singular value decomposition. It is used as a statistical procedure to extract coherent structures within various flows. POD does this by constructing an eigenvector basis to build a modal decomposition from an ensemble of data signals. An eigenvalue problem is solved using a correlation matrix constructed from an autocovariance of the temporal ensemble of velocity snapshots in time, which results in a spatial autocovariance matrix.

First consider a collection of instantaneous flow field snapshots $U(\mathbf{x}, t)$ sampled at m spatial locations and n time instances. The fluctuating (u') and mean ($\overline{u}$) velocity fields are calculated from these n instances as follows:

$$u'(\mathbf{x}, t_j) = U(\mathbf{x}, t_j) - \overline{u}(\mathbf{x}), \quad \text{where } \overline{u}(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^{n} U(\mathbf{x}, t_j) \tag{6.3}$$

The fluctuating velocity matrix $\boldsymbol{M} \in \mathbb{R}^{(m\times n)}$ is constructed as follows:

$$M_{ij} = u'(x_i, t_j), \quad i = 1, \ldots, m,\ j = 1, \ldots, n \tag{6.4}$$

The traditional POD method formulates the correlation matrix $\tilde{\boldsymbol{C}} = \frac{1}{n}\boldsymbol{M}\boldsymbol{M}^T \in \mathbb{R}^{(m\times m)}$. From this, an eigenvalue problem is solved as follows:

$$\tilde{\mathrm{C}}\boldsymbol{\phi} = \boldsymbol{\phi}\boldsymbol{\lambda} \tag{6.5}$$

where $\boldsymbol{\phi} = \phi^l(x_i) \in \mathbb{R}^{(m\times m)}$ is a square matrix whose columns (index l) are the eigenvectors, and $\boldsymbol{\lambda} \in \mathbb{R}^{(m\times m)}$ is a diagonal matrix containing the eigenvalues λ_{ll}. $\boldsymbol{\phi} = \phi^l(x_i)$ are known as the POD modes. The eigenvalues represent the relative kinetic energy in each POD mode, which represent the dominant flow structures. The eigenvalues λ_{ll} are ordered such that:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_m \geq 0 \tag{6.6}$$

The instantaneous fluctuating velocity field can be represented as a series expansion of POD mode and POD coefficient products. The POD time-varying coefficients are calculated by projecting the velocity fluctuations onto the POD modes as follows:

$$a^l(t_j) = \frac{1}{m} \sum_{i=1}^{m} \phi^l(x_i) \cdot u'(x_i, t_j) \tag{6.7}$$

Finally, the instantaneous fluctuating velocity can be reconstructed as follows:

$$u'(x_i, t_j) = \sum_{l=1}^{m} a^l(t_j) \cdot \phi^l(x_i) \tag{6.8}$$

Note that for large data ensembles that have more temporal samples than spatial samples, such as in experimental studies, the traditional POD method is advantageous as the correlation matrix is size $m \times m$, which is smaller than $n \times n$, thus solving the eigenvalue problem in Equation 6.5 is smaller and easier. However, in computational data, there are generally more spatial samples than temporal samples. Thus, it is advantageous to use the snapshot POD method, which was introduced in 1987 by Sirovich [184]. Snapshot POD, alternatively, solves the eigenvalue problem using the correlation matrix $\boldsymbol{C} = \frac{1}{n}\boldsymbol{M}^T\boldsymbol{M} \in \mathbb{R}^{(n \times n)}$.

Recall, the singular value decomposition (SVD) of a matrix $\boldsymbol{M} \in \mathbb{R}^{(m \times n)}$: $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{(m \times m)}$ is a unitary matrix, i.e. $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{(m \times n)}$ is a diagonal matrix composed of the singular values of $\boldsymbol{M}$, and $\boldsymbol{V}^T \in \mathbb{R}^{(n \times n)}$ is also a unitary matrix. The matrix $\boldsymbol{U}$ is the set of the orthonormal eigenvectors of $\boldsymbol{M}\boldsymbol{M}^T$, and $\boldsymbol{V}$ is the set of orthornormal eigenvectors of $\boldsymbol{M}^T\boldsymbol{M}$. Further, the diagonal entries $\boldsymbol{\Sigma}$ are the square roots of the non-zero eigenvalues of both $\boldsymbol{M}\boldsymbol{M}^T$ and $\boldsymbol{M}^T\boldsymbol{M}$. Thus, the original POD method solves for the matrix $\boldsymbol{U}$ and the square values of the matrix $\boldsymbol{\Sigma}$. Alternatively, using the snapshot POD method, one can solve for $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$, then solve for the matrix $\boldsymbol{U}$ by the following procedure:

$$
\begin{aligned}
\boldsymbol{M} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\
\boldsymbol{M}\boldsymbol{V} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V} && \boxed{\text{right multiply } \boldsymbol{V}} \\
\boldsymbol{M}\boldsymbol{V} &= \boldsymbol{U}\boldsymbol{\Sigma} && \boxed{\boldsymbol{V} \text{ unitary}} \\
\boldsymbol{M}\boldsymbol{V}\boldsymbol{\Sigma}^{-1} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1} && \boxed{\text{right multiply } \boldsymbol{\Sigma}^{-1}} \\
\boldsymbol{M}\boldsymbol{V}\boldsymbol{\Sigma}^{-1} &= \boldsymbol{U} && (6.9)
\end{aligned}
$$

Note that $\boldsymbol{\Sigma}^{-1}$ is nonconventional notation as the matrix is non-square. However, $\boldsymbol{\Sigma}$ is a diagonal matrix with non-zero eigenvalues for the first $\min(m, n)$ entries, thus, reconstruction of the first $\min(m, n)$ columns of the $\boldsymbol{U}$ is as follows:

$$\boldsymbol{U}_l = \frac{1}{\sqrt{\lambda_{ll}}}\boldsymbol{M}\boldsymbol{V}_l \tag{6.10}$$

Through this process, we are able to solve the eigenproblem of size $\min(m^2, n^2)$ making high spatial resolution simulations tractable for POD using a small number of time samples. Once the eigenvalues and eigenvectors are reconstructed through Eqn. (6.10), the process of reconstructing the POD coefficients and velocity fluctuations is analogous to the procedure outlined in Eqns. (6.7) and (6.8).

Results and Analysis

Snapshot POD is applied to the WindPACT-1.5MW wind turbine simulation for an ensemble of data composed of 16 rotor revolutions. The autocovariance matrix is constructed using scalar velocity components, i.e. POD is applied to a single velocity component at a time. The spatial samples are taken from the two-dimensional planes at downstream stations at x/D = 0.5, 1.0, 2.0, and 3.0, as shown in Fig. (6.16).

The POD mode energy is shown in Fig. (6.29) for axial, radial, and azimuthal fluctuating velocity components at multiple downstream stations. The red bars correspond to the mode energy with the mode number as the abscissae. The blue curve represents the accumulating

energy as the mode energies are summed. For example, the first six modes for the axial fluctuating velocity at station x/D = 0.5 contain approximately 90% of the fluctuation total energy. The purple dashed line represents the total kinetic energy of the fluctuating velocity; one can compare the kinetic energy of the fluctuating velocity components by reading across each row. The axial fluctuating velocity at x/D = 0.5 has approximately half the energy as the radial fluctuation velocity, but nearly three times the energy of the azimuthal fluctuating velocity.

From Fig. (6.29), the axial and radial fluctuating mode energies are low-dimensional in the near-wake region up to station x/D = 2.0 as most energy is contained in the first six modes. Further, strong mode pairs exist (modes 1 and 2, modes 3 and 4) indicating strong flow structure coupling. The azimuthal mode energy is much more distributed as only 70% of the total energy is accounted for in the first 20 modes for station x/D = 0.5 and less than 30% for station x/D = 3.0. At the beginning of the results section, a qualitative observation of the wake appearing to break down in regions between x/D = 2.0 and 3.0 was stated. This observation is confirmed, quantitatively, through the mode energy evolution for all fluctuating velocity components; the mode energy has a significant redistribution from lower modes to higher modes between stations x/D = 2.0 and 3.0. Further, the total energy in the radial component starts very high in comparison to the azimuthal component but loses a third of its energy as the wake moves downstream where a significant increase in the azimuthal mode energy at station 3.0D occurs. This highlights not only a redistribution of energy within its own modes but to other fluctuating velocity components. This confirms the results from the Reynolds stress analysis demonstrating the growth of the azimuthal stress shown in Fig. (6.28)(b).

Fig. (6.30) illustrates the POD time-varying coefficients. The amplitudes of the time-varying coefficients demonstrate the mode energy, and the phases demonstrate the pairing. As seen in all fluctuation velocities, pairings between modes 1 and 2 and modes 3 and 4 are present as their respective amplitudes are approximately the same with an approximate 90° phase shift. To further demonstrate this pairing for the axial fluctuating velocity, Fig. (6.31) shows modes a_2 and a_4 plotted as functions of modes a_1 and a_3, respectively, at stations x/D

= 0.5 and 2.0. As the wake moves downstream, we observe this tight coupling between modes begin to deteriorate. This is especially true from station x/D = 2.0 to 3.0. More acutely, the axial time-varying coefficients show an early decoupling in mode pair 3 and 4 at station 2.0D. This decoupling may introduce the energy instability allowing vortex interactions, such as vortex pairing and hopping.

Fig. (6.32) shows a time series of mode 1 for the axial fluctuating velocity at station 0.5D which shows the largest energy containing structure over a period of one rotor revolution. The illustration shows the oscillation of mode 1 changing between positive and negative values at a 60° phase frequency. As a tip vortex passes through the x/D = 0.5 plane, the time-varying coefficient oscillates one full period over 120° of rotation. Thus, for three blades, there are three full periods passing through the wake station plane.

Next, the first 10 POD mode structures at stations 0.5D, 1.0D, 2.0D, and 3.0D are shown in Figs. (6.33), (6.34), (6.35), and (6.36), respectively. Similar mode pairing structures appear in the near-wake region for the axial and radial velocity fluctuations. Most of the structured content is found in the first four modes as asserted by the mode energies. Mode 3 at station 2.0D for the axial component begins to exhibit structural differences compared to its mode pair, mode 4, whereas the radial component still shows strong mode correlation for modes 3 and 4. Significant structure changes emerge at station 3.0D after the initial stages of wake breakdown. The strong asymmetry between axial modes 1 and 2 show a transfer of energy. Particularly, mode 1 of the radial component shows strong negative radial velocity, which indicates strong entrainment from the freestream velocity. More structures appear in higher modes for all three velocity components highlighting that the flow has taken on a higher dimensionality. However, since the wake has begun the transition from near-wake to mid-wake incorporating more turbulent effects, more data is required to assert convergence of the statistics.

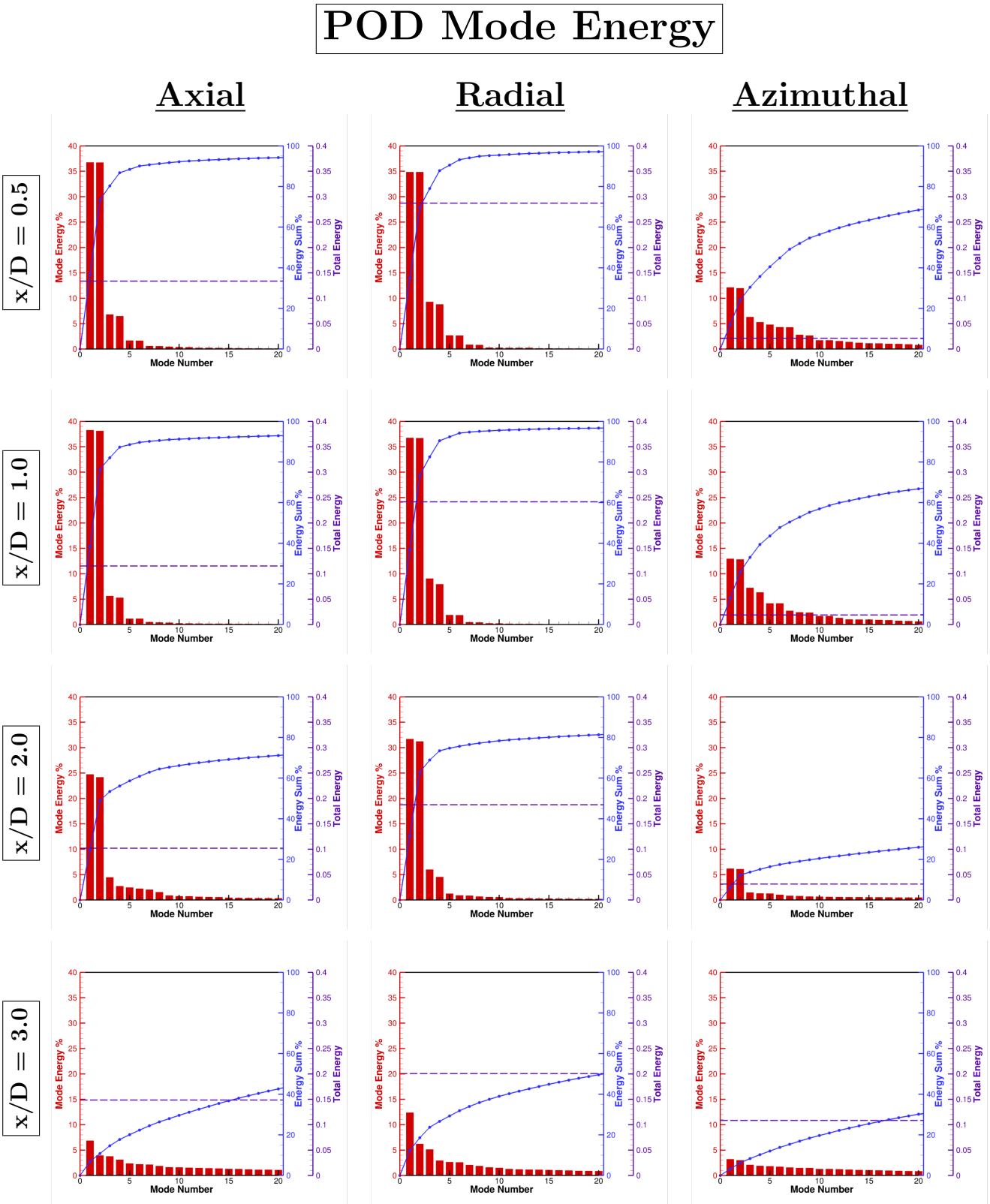


Figure 6.29: POD mode energies for axial (u'), radial (v'), and azimuthal (w') fluctuation velocities at downstream wake positions.

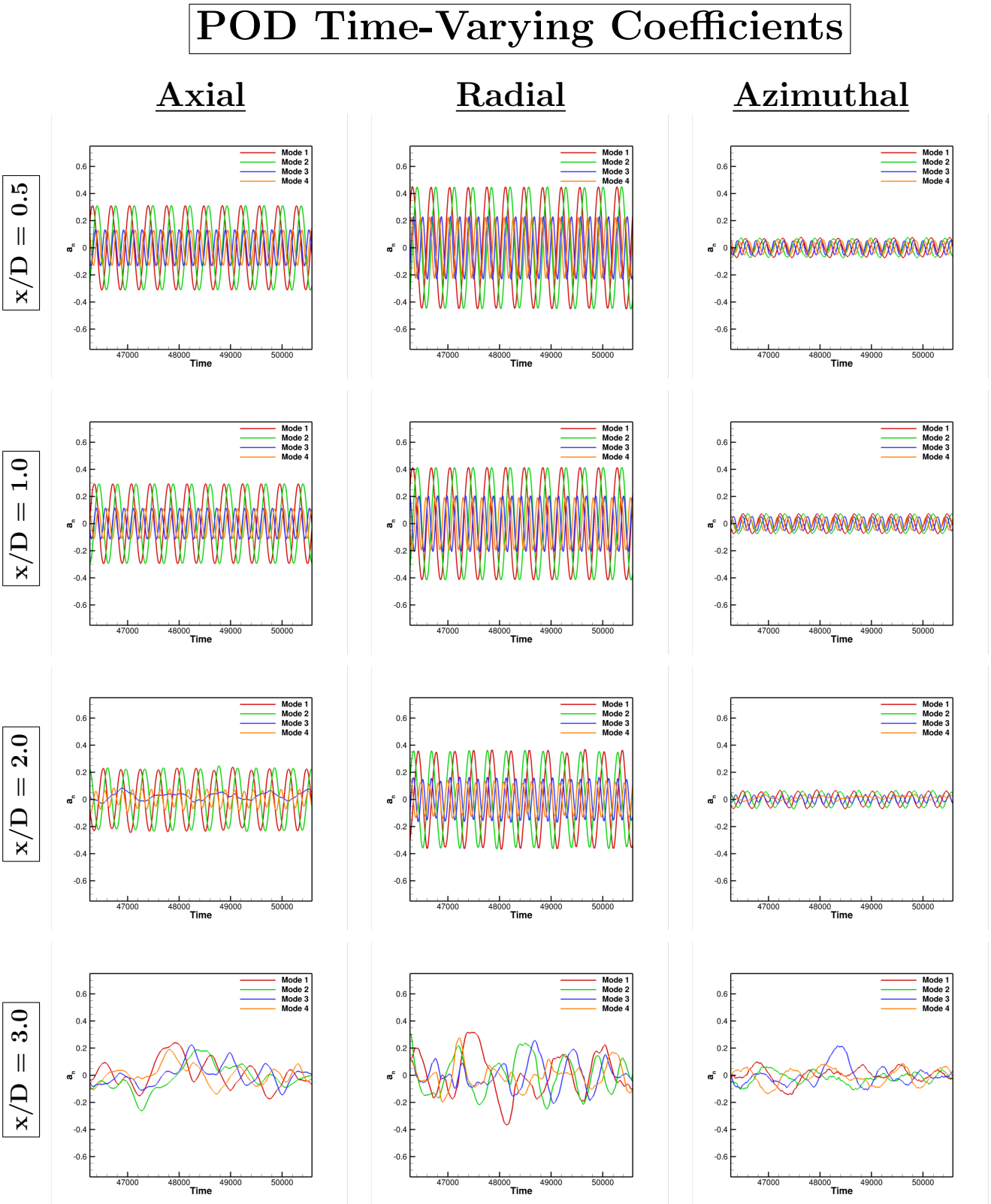
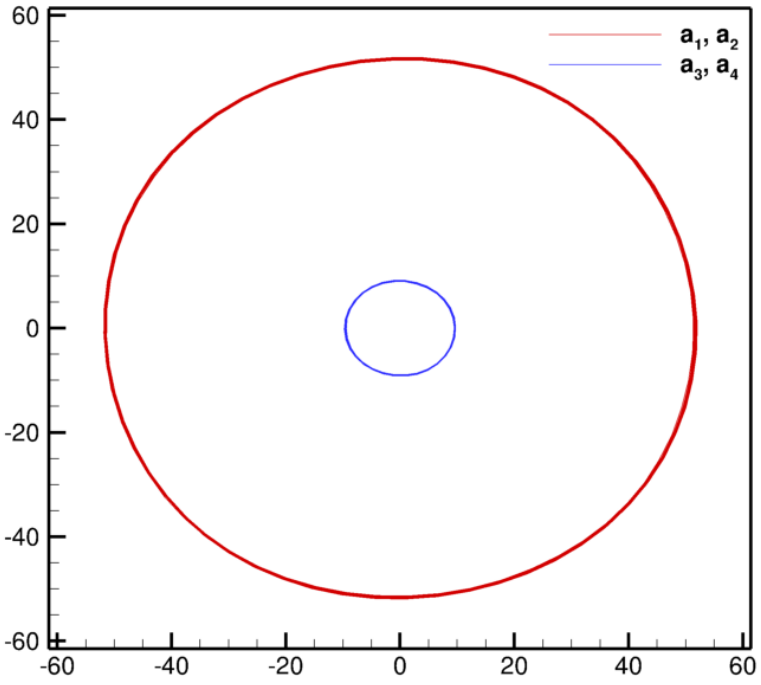
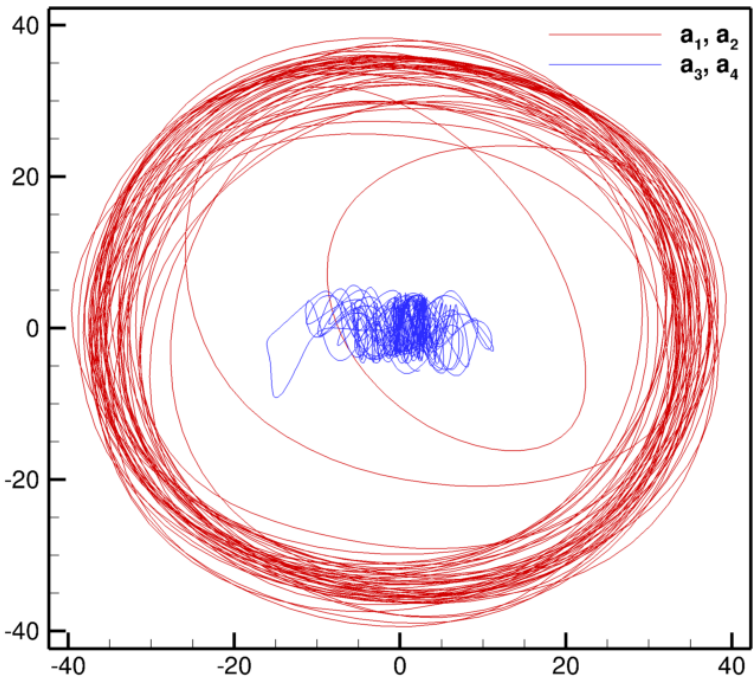


Figure 6.30: POD time-varying coefficients for u', v', and w' fluctuation velocities at downstream wake positions.



(a) **x/D = 0.5**



(b) **x/D = 2.0**

Figure 6.31: Axial fluctuation velocity time-varying coefficient pairings between modes 1 & 2 and modes 3 & 4.

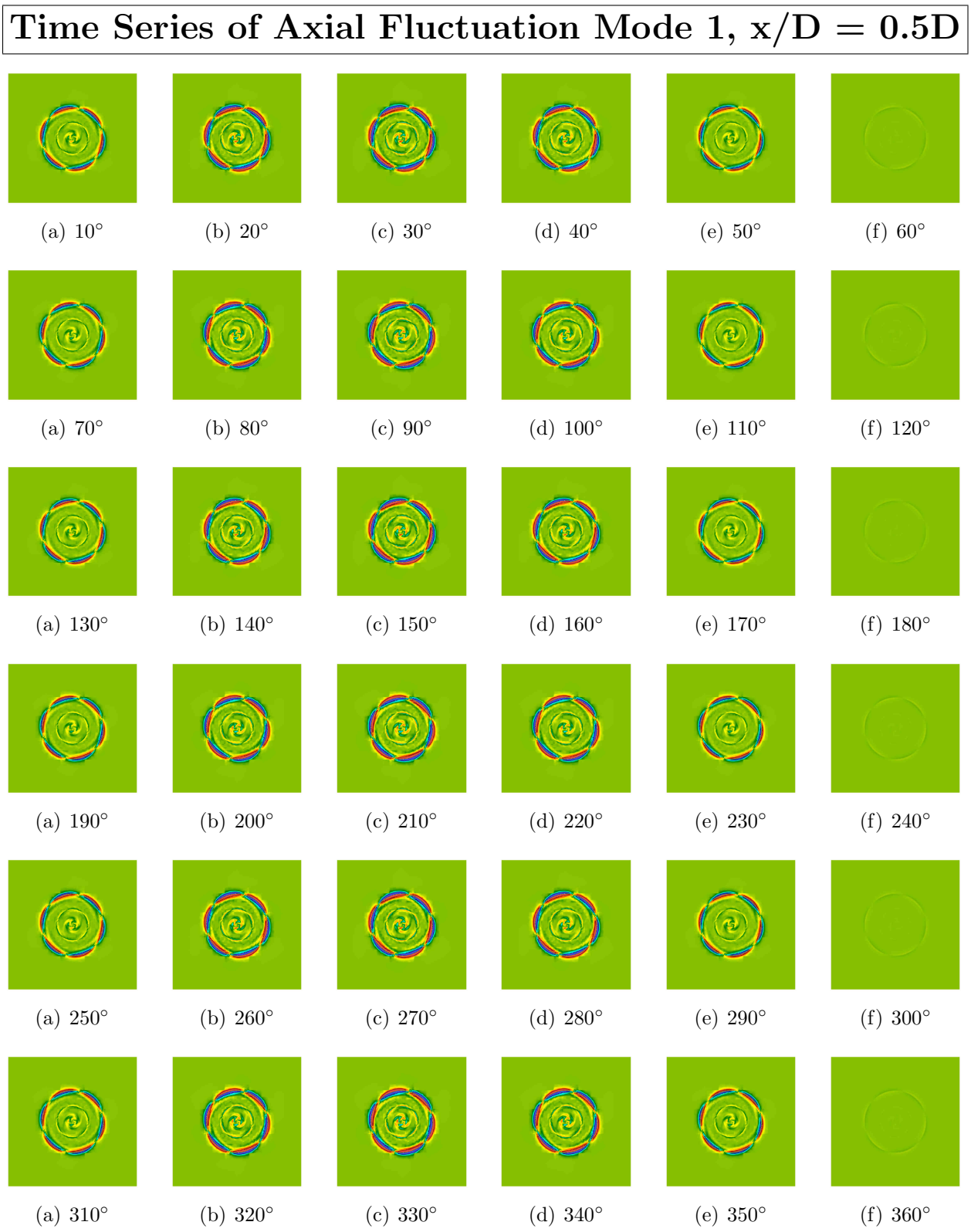


Figure 6.32: Time series of POD mode 1 for u' at x/D = 0.5D over the period of one rotor revolution.

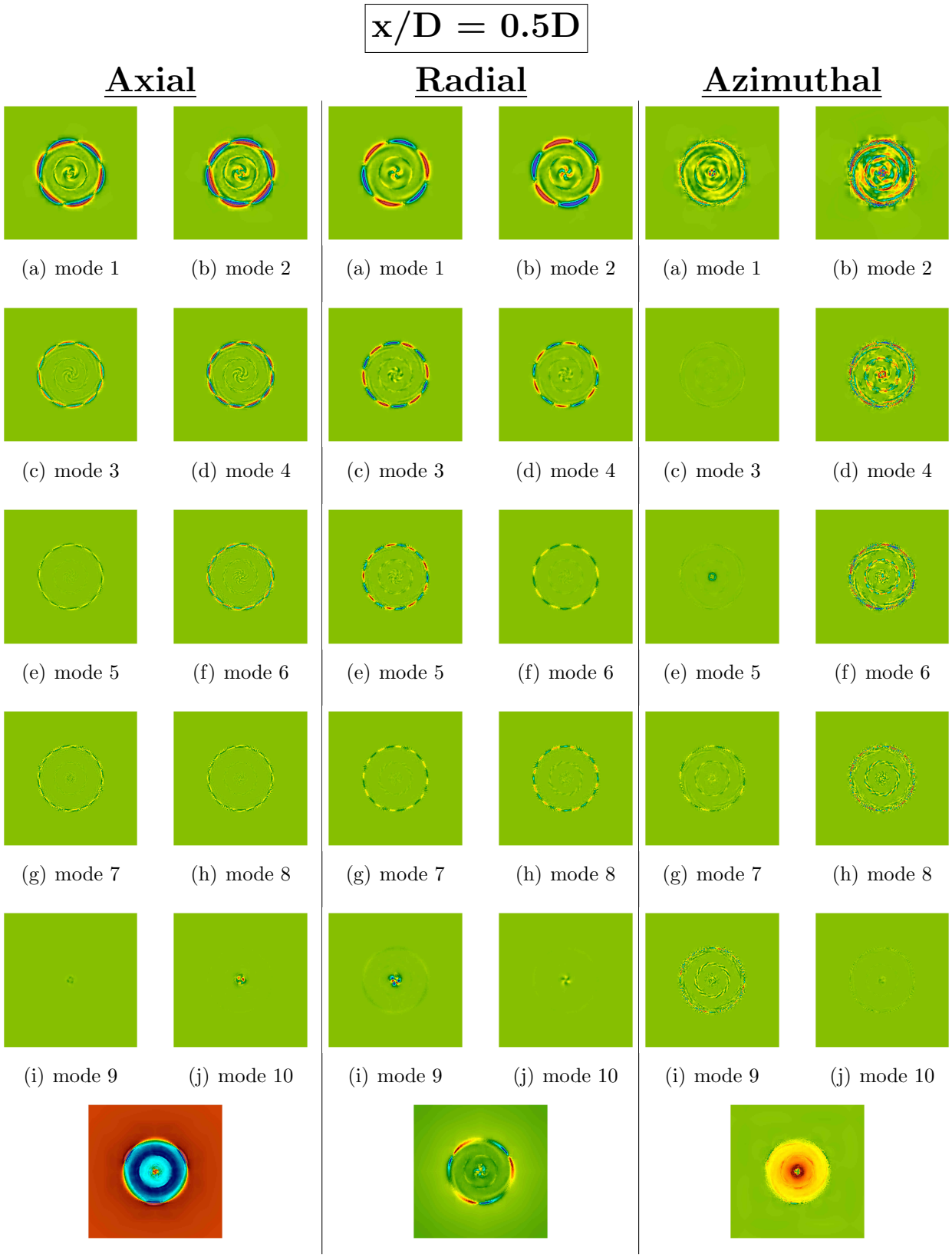


Figure 6.33: POD modal decomposition and instantaneous flow velocities (bottom) at downstream position x/D = 0.5D.

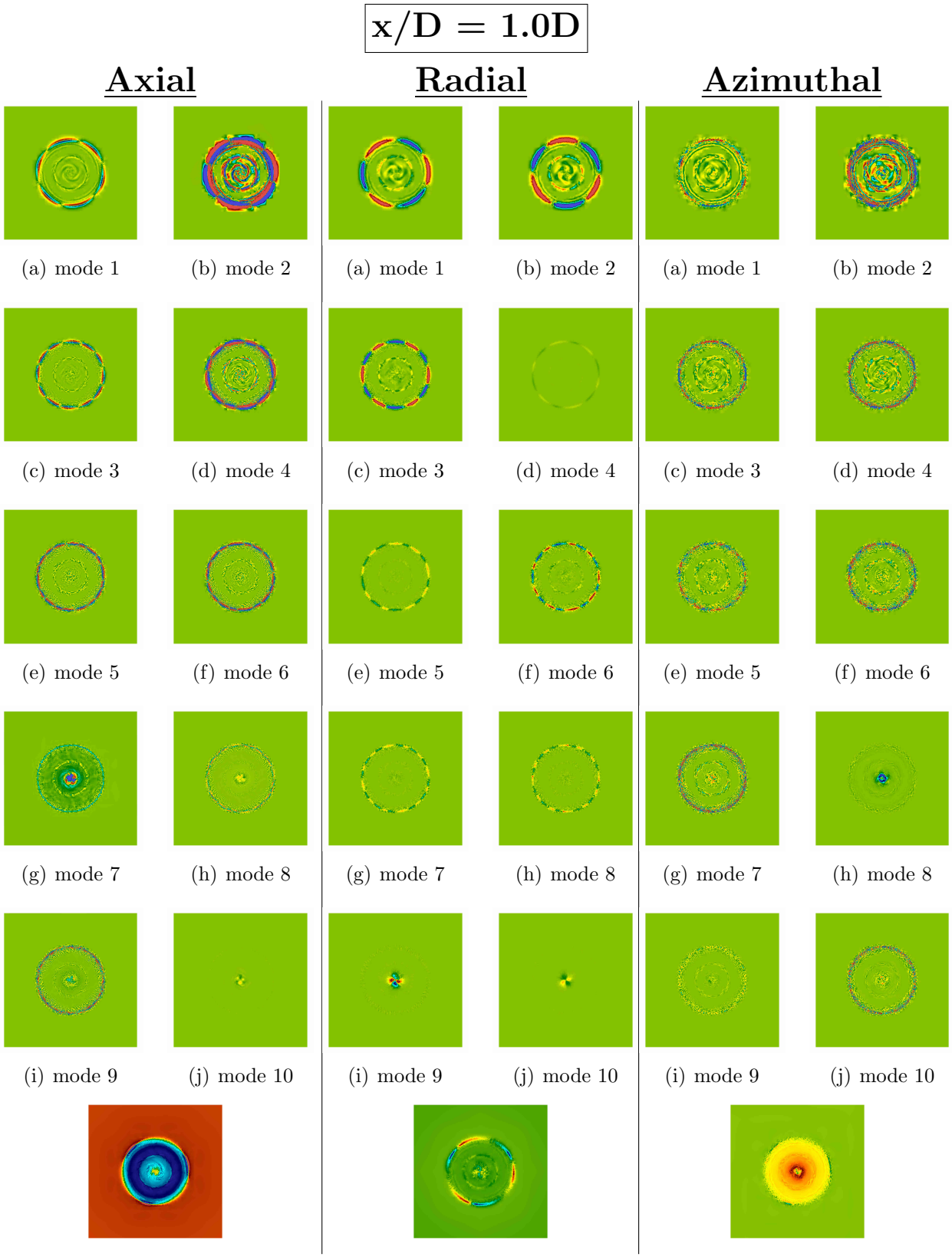


Figure 6.34: POD modal decomposition and instantaneous flow velocities (bottom) at downstream position x/D = 1.0D.

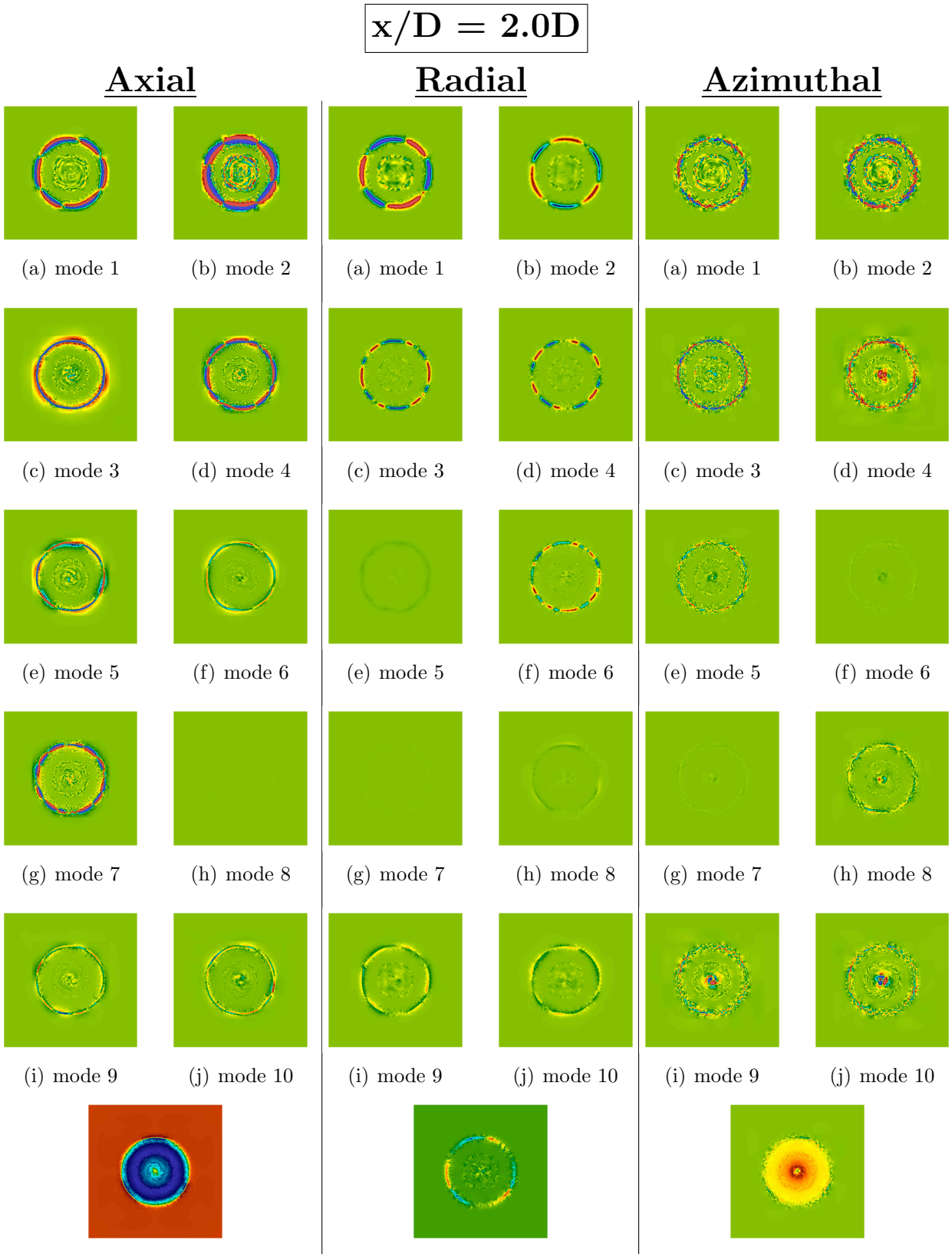


Figure 6.35: POD modal decomposition and instantaneous flow velocities (bottom) at downstream position x/D = 2.0D.

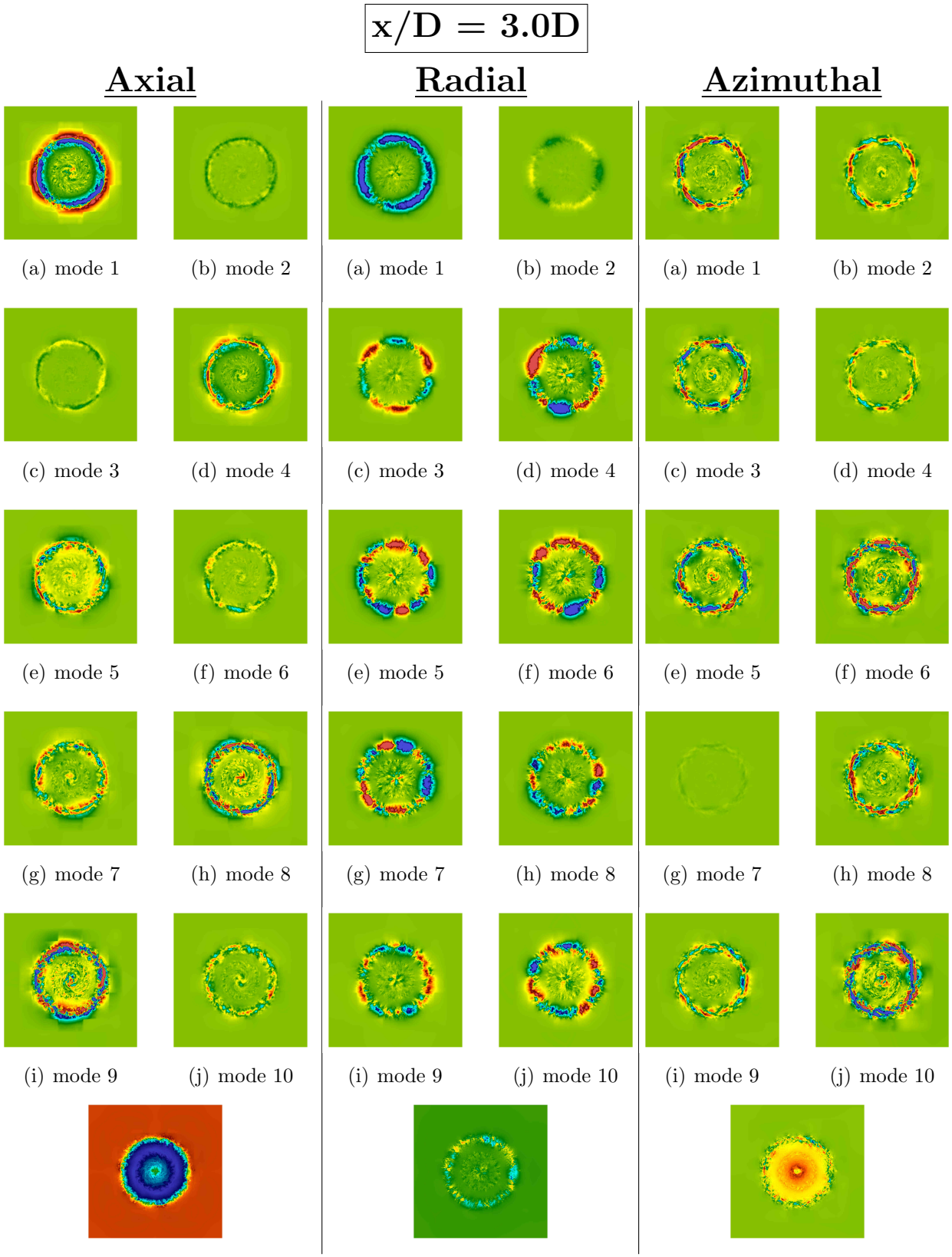


Figure 6.36: POD modal decomposition and instantaneous flow velocities (bottom) at downstream position x/D = 3.0D.

6.5 Atmospheric Inflow Wake Comparison Results

Accurate representation of the atmospheric inflow conditions are required for proper wind plant CFD analysis. To perform these tasks, the W^2A^2KE3D framework has incorporated an atmospheric micro-scale flow solver interface. This interface serves as a one-way coupling from the atmospheric solver to the CFD through the overset methodology. A micro-scale atmospheric solution is registered as a pseudo-CFD solver with an unstructured grid to the overset assembler.. A precursor atmospheric simulation is performed with specific atmospheric conditions, e.g. turbulence intensity, stable, neutral, unstable boundary layer, and complex terrain environments such as a specific geographical location. This precursor simulation is run until statistically converged flow statistics are achieved. When this is complete, a time history of flow solutions are written to disk for a specified duration of physical simulation time. When the CFD simulation is initialized, all initial flow variables in the near-body and off-body mesh system are filled from the atmospheric data. During a wind plant simulation, the boundary elements of the off-body mesh system are updated via linear-interpolated-time atmospheric data.

6.5.1 SOWFA Precursor Results for Neutral ABL

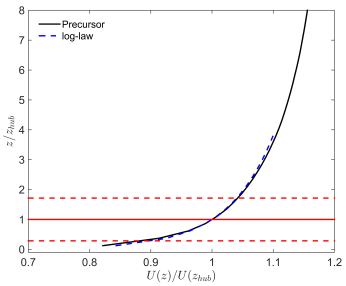
The atmospheric inflow for the Lillgrund wind farm is based on the meteorological conditions described in Bergström et al. [185] and in the large eddy simulation performed by Churchfield *et al.* [186]. A neutral atmospheric boundary layer (ABL) is assumed with a mean hub-height velocity of 9 m/s from a direction of 221.6° and a surface aerodynamic roughness value of $Z_0 = 10^{-4}$ m is chosen to reproduce the hub-height turbulence intensity of about 6% [185].

The precursor LES domain size is 10,240 m × 4,096 m × 1,024 m with a uniform 16 m resolution in all directions resulting in a mesh consisting of 640 × 256 × 64 hexahedral cells. Periodic boundary conditions are applied in the wind-wise and cross-stream directions, and a slip-wall is used at the top boundary of the domain. A capping inversion of 100 m at 700-800 m is applied to limit the boundary layer growth. The initial potential temperature field is kept uniform at 300 K from the surface to 700 m and within the capping inversion

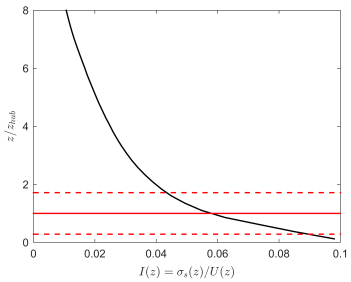
the potential temperature rises by 8 K. Above 800 m, the potential temperature gradually increases at a rate of 0.003 K/m. This potential temperature profile is similar to that used by Churchfield et al. [186] and Moeng and Sullivan [187]. The initial velocity profile is approximated using a log-law of the wall and small perturbations are added near the surface to promote transition to a turbulent flow. The atmospheric boundary layer is simulated for 12,000 seconds to allow the initial transients to pass and achieve a quasi-equilibrium state and then run for an additional 3,000 seconds to record the velocity field at each time step which are to be coupled with W^2A^2KE3D .

Fig. (6.37) depicts vertical profiles of the temporally and horizontally averaged velocity, turbulence intensity and turbulence kinetic energy profile for the lower part of the ABL. The mean velocity closely follows the log-law (based on the specified rotor hub-height velocity and surface roughness) near the surface. Although this simulation contains no rotor, the target wind turbine rotor for this configuration would experience a significant mean wind shear of 1.3 m/s across the rotor diameter. The turbulence intensity is largest near the surface at approximately 10% and decreases to the desired 6% at the hub-height. In the precursor LES, the turbulence kinetic energy is mostly resolved and only a small portion is modeled by the SGS model with the exception of the first two cell levels where the modeled contribution is significant. This is a common problem of all neutral ABL LES irrespective of the resolution since the turbulence integral length scale is proportional to the distance from the surface in the log-law region and thus in the first few cells above the surface the turbulence length scale and the filter scale are comparable.

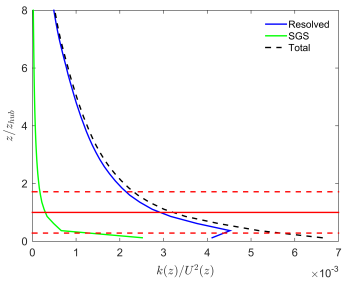
Fig. (6.38) illustrates contours of instantaneous normalized velocity fluctuation in a horizontal plane at the rotor hub-height. These contour plots reveal the presence of a wide range of scales in the turbulent boundary layer. The plots indicate the presence of turbulence structures that are very large (order of several kilometers) in the wind-wise direction. The existence of these large structures motivate the need for large domain sizes considered in LES of the atmospheric boundary layer. If the simulation domain is size is too small, the adopted periodic boundary conditions would artificially lock the elongated structures in place and thereby produce a spatially biased inflow condition for the wind plant CFD simulation.



(a) Mean wind profile.

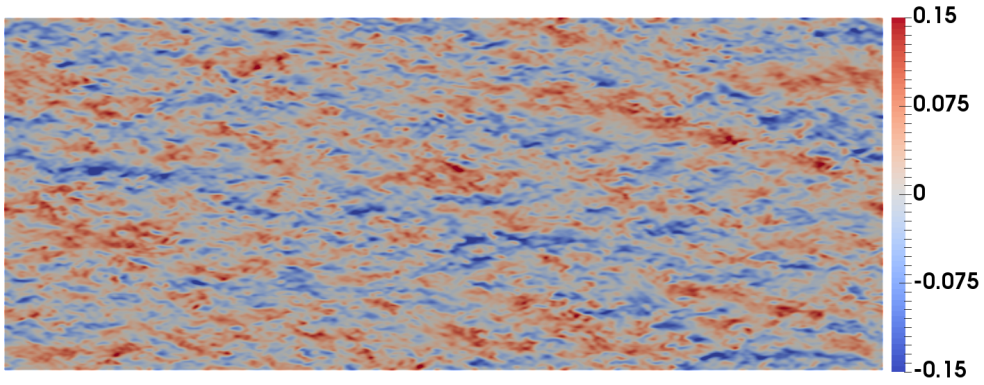


(b) Turbulence intensity profile.

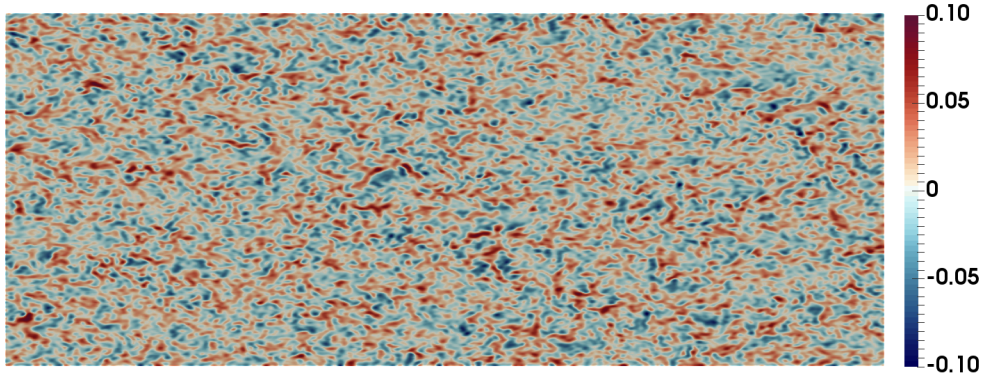


(c) Turbulence kinetic energy profile.

Figure 6.37: Vertical profiles of temporally and horizontally averaged velocity, turbulence intensity and turbulence kinetic energy from the precursor LES. The solid red horizontal line represents the hub height and the two horizontal dashed lines represent the vertical extent of the wind turbine rotor.

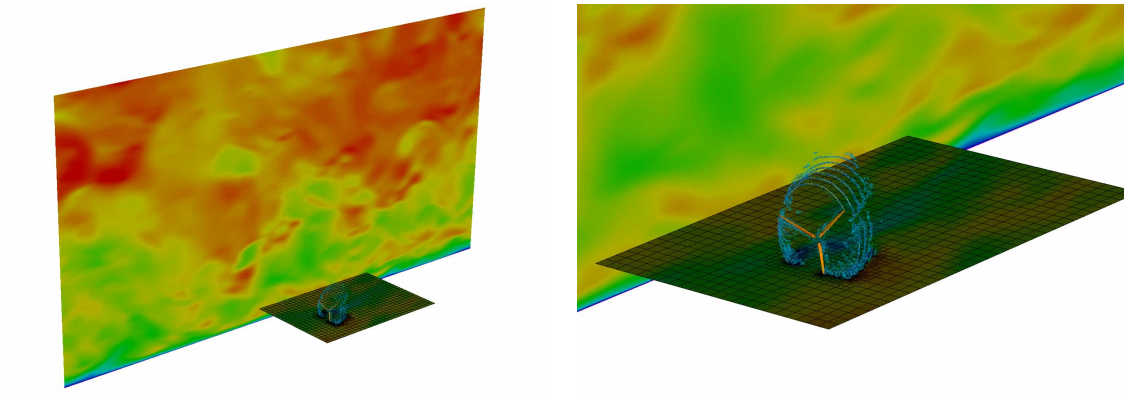


(a) Wind-wise velocity fluctuations.



(b) Vertical velocity fluctuations.

Figure 6.38: Contours of instantaneous velocity fluctuation at rotor hub height horizontal plane of precursor LES velocity normalized by mean wind speed.



(a) NCAR WRF atmospheric inflow coupling to the off-body solver.

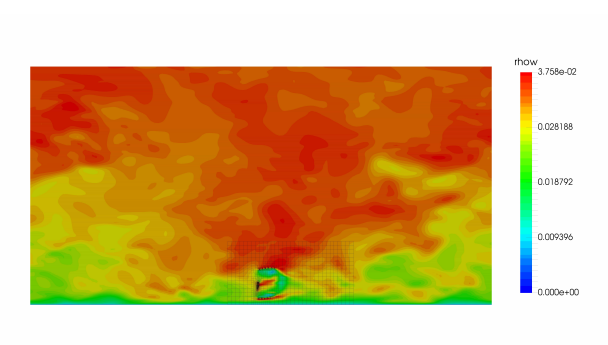
(b) Zoomed view of the NCAR WRF inflow.

Figure 6.39: Micro-scale atmospheric and CFD coupling with NCAR's WRF solver to the off-body CFD solver `dg4est` for a single NREL 5MW wind turbine.

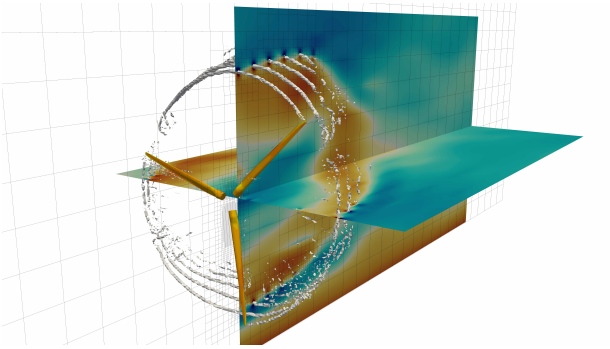
6.5.2 Coupled Micro-Scale Atmospheric and CFD Results

SOWFA is the primary ABL LES solver used in this work to produce the inflow for the wind plant CFD simulations. SOWFA enables simulations of arbitrarily complex terrain through the use of unstructured grids and a terrain aligned implementation of the Schumann-Grötzbach [188,189] wall model. First results for real complex terrain at the Sierra Madre site in south-central Wyoming and the Bolund hill [159–161] are indeed promising. To demonstrate the versatility of the multiple solver paradigm in W^2A^2KE3D , we have incorporated SOWFA inflow as well as NCAR's WRF-LES [158] inflow.

Preliminary work of coupling WRF and SOWFA with the blade resolved wind plant simulation code is shown in Fig. (6.39) and Fig. (6.40), respectively. In particular, Fig. (6.40) depicts the simulation of a single Siemens turbine coupled to the precursor SOWFA calculation described in the previous section. As seen in the figure, atmospheric inflow conditions break down wake structures much faster than uniform inflow conditions. Simulations using uniform inflow significantly under predicts the energy produced by turbines that are in the wake of other turbines [39]. This is due to the inability to capture momentum through the lack of turbulent mixing.



(a) NREL SOWFA atmospheric inflow coupling to the off-body solver.



(b) Zoomed view of the NREL SOWFA inflow.

Figure 6.40: Micro-scale atmospheric and CFD coupling with NREL's SOWFA solver to the off-body CFD solver `dg4est` for a single NREL 5MW wind turbine.

Chapter 7

Wind Farm Simulation Results

This chapter presents results of a parallel weak scaling test of the computational framework by introducing more wind turbines on comparable counts of CPU cores per turbine. Examination of a longer run-time simulation of the 48 wind turbine Lillgrund wind farm is performed, and a discussion of a 144 turbine wind plant simulation is presented.

7.1 Weak Scalability Improvements

Parallel scalability of the computational framework is essential for enabling simulation of wind plants using full rotor models. This section is concerned with weak scalability which is defined as how the solution time varies with the number of processors for a fixed problem size *per* processor. To perform a weak scaling test in the context of a wind plant simulation, we assign a fixed number of processors per wind turbine. Then for each weak scaling sample, we increase the number of wind turbines simulated in a wind plant configuration along with the total number of cores used for the simulations. The ability to weak scale the wind plant simulation software is essential for simulating hundreds of wind turbines in a wind plant configuration.

In this study we perform the weak scaling test starting with six wind turbines using 348 CPU cores per turbine for the near-body solver and 120 CPU cores per turbine for the off-body solver. The weak scaling test evaluates the parallel weak scalability at 6, 12, 24, 48,

and 96 wind turbines over a 9.5 hour wall-clock time simulation window. The total number of CPU cores used ranges from 2,808 to 44,928.

The wind turbine chosen for the weak scaling study is the Siemens SWT-2.3-93. As demonstrated from the single wind turbine performance study, the required mesh resolution to accurately capture the aerodynamic forces uses just over 2.2 million nodes per blade. Allocating 108 CPU cores per blade and 24 CPU cores per tower equates to 20,555 and 21,040 nodes per core for each blade and tower, respectively, for a total of 348 CPU cores per wind turbine. The off-body solution accuracy chosen for this study is $p = 1$ near the unstructured mesh transitioning to $p = 2$ away from the wind turbine.

The weak scaling study is performed on the NSF NWSC-2 Cheyenne supercomputer [190]. Cheyenne contains 145,152 Intel Xeon E5-2697V4 processor cores rated at 2.6 GHz. The Intel Xeon E5-2697V4 processor uses the AVX-2 instruction set allowing for four double precision operations to be performed in single-instruction-multiple-data (SIMD) parallelism. Cheyenne contains 4,032 compute nodes and with two processors per node totaling 36 CPU cores per compute node. The total theoretical peak performance of Cheyenne is 5.34 petaflops. The network is a Partial 9D Enhanced Hypercube single-plane interconnect topology with Mellanox EDR InfiniBand high-speed interconnect.

Two initial challenges that limited weak parallel scalability have been addressed. The first scaling issue arose for mesh domain intersection checking. In the overset framework, all mesh partitions (one per MPI rank) are assigned an alternating digit tree (ADT) for efficient domain searching. The oriented bounding box of the ADTs are sent in an all-to-all communication to check for intersections. However, `p4est` partitions cells into non-contiguous groups due to *z-order* partitioning [145]. That is, groups of mesh cells on a mesh partition in a single MPI rank may not actually touch each other and have a large space between them. This can cause elements on opposite sides of a computational domain to be placed within the same bounding box, resulting in large oriented bounding boxes. When the overset assembler performs an intersection check with this non-contiguous mesh partition, large amounts of intersections may be found even though no cells may actually intersect each other. To address this issue, all near-body bounding boxes are communicated to all

off-body processors and a local bounding-box-to-cell intersection check is performed by the off-body processors. This process removes these false-positive intersections, thus allowing for true intersections to be registered with overset assembler. This local bounding-box-to-cell intersection check uses the efficient octree search built into the `p4est` AMR framework. Using these intersection results, a processor communication map is constructed.

The second scaling issue arose from inter-grid boundary points (IGBPs). IGBPs are points that used to connect the off-body mesh system to the near-body meshes. The off-body mesh resolution must approximately match the resolution of the coarsest mesh elements of the trimmed near-body mesh. These IGBPs are the outer mesh points located on or near the surface of the trimmed near-body meshes. IGBPs locations and corresponding element sizes are communicated from the near-body meshes to the off-body solver so that the off-body AMR mesh can be adaptively refined to these locations to match the mesh resolution for the meshes to accurately exchange solution data. Since the near-body mesh is moving through the off-body mesh and the off-body mesh is adapting or repartitioning after every global time step, the original algorithm sent a list containing all IGBPs globally to all processors. This caused scaling issues at large core counts particularly when many meshes were used. The global list of IGPBs became substantially large, therefore increasing the communication cost. Additionally, since each list contained all IGBPs, searching of the list became costly, therefore creating a bottleneck in the regridding process of the off-body mesh. To address this issue, the same processor map used for the bounding box intersection check is used to reduce the number of IGPBs communicated and to make the IGBP list unique to each off-body processor.

7.2 Weak Scaling

Table (7.1) shows the present performance statistics for the weak scaling study. The results assume that the efficiency of the six wind turbine simulation is perfect as a reference value. When doubling the number of turbines successively, the parallel scalability efficiency is 98.7% for 12 turbines, 96.8% for 24 turbines, and 93.3% for 48 turbines. The weak scalability decreases slightly in performance when simulating 96 wind turbines giving an efficiency of 86.9%.

Table (7.2) displays solver specific timings for each of the wind plant configurations. The blade, tower, and off-body times correspond to the CFD solver times. The various component meshes of the CFD solvers are all run in parallel. However, the overset connectivity determination is performed on all processors at the end of each time step executed by the CFD solvers. Thus the total wall-clock time for each complete time step corresponds to the sum of the maximum CFD solver time for a time step and the overset connectivity time. The blade time corresponds to the near-body blade mesh and solver that is replicated three times for each wind turbine. The run times for the blade and tower meshes are on average constant for all wind plant configurations. This is expected since each of the near-body meshes are independent of each other and the computational work remains constant for the duration of simulations. Each near-body mesh uses a new instance of the near-body solver, thus decoupling the near-body flow meshes. The off-body solve times slightly increase from 6 to 12 turbines then to 24, 48, and 96 wind turbines. The larger wind plant configuration run times become approximately constant. The off-body solver uses only one instance of the off-body flow solver. Thus the weak scalability of the AMR framework `p4est` is demonstrated. Notice the average solve time at 48 wind turbines is approximately the same at 96 wind turbines for the off-body solver.

For deeper analysis of the 96 wind turbine case, Fig. (7.1) shows the distribution of solver times per time step. Row 1 shows the frequency of the CFD solver execution times over the entire run. As seen in Fig. (7.1)(a), the near-body solve time for the blade mesh has a fairly wide distribution ranging from 9 seconds to 10 seconds. The wide distribution can be attributed to I/O of log files. The near-body solver, which was composed of 288

Weak Scaling Wind Farm Study: Overall Performance Statistics

Turbine Count	Efficiency	Revs	Near-Body Cores	Off-Body Cores	Total Cores
6	1.0000	1.374	2,088	720	2,808
12	0.9874	1.360	4,176	1,440	5,616
24	0.9682	1.331	8,352	2,880	11,232
48	0.9333	1.283	16,704	5,760	22,464
96	0.8686	1.194	33,408	11,520	44,928

Table 7.1: Weak scaling wind plant study performed on NWSC-2 Cheyenne up to 96 wind turbines for wall-clock time of 9.5 hours. Six turbines are used as the perfect scaling reference.

Weak Scaling Wind Farm Study: Solver Performance Statistics

Turbine	Blade Time (s)			Tower Time (s)			Off-body Time (s)			Overset Time (s)		
Count	min	max	avg	min	max	avg	min	max	avg	min	max	avg
6	8.944	9.588	9.067	7.111	7.905	7.135	4.299	9.455	7.027	7.623	8.558	8.056
12	8.949	9.347	9.075	7.126	7.632	7.148	4.310	10.11	7.152	7.684	8.632	8.141
24	8.980	9.931	9.178	7.124	7.721	7.208	4.180	11.29	7.261	7.842	9.295	8.314
48	8.996	10.00	9.224	7.147	7.974	7.243	4.203	11.29	7.428	8.056	10.89	8.613
96	9.069	9.903	9.225	7.119	7.774	7.143	4.511	11.16	7.406	9.332	14.08	10.32

Table 7.2: Weak scaling wind plant study solver times up to 96 wind turbines.

solver instances for the blades, logged large amounts of solver data to a single output log file. This causes a bottleneck in the I/O therefore slowing down the execution time even though the blade meshes are independent of each other with fixed numbers of degrees-of-freedom. As shown in Fig. (7.1)(b), the near-body mesh solve time frequencies better demonstrate the independence of the near-body meshes where the cpu-time distribution is very tight. Fig. (7.1)(c) shows the distribution of the off-body solve times. The frequency distribution is expected to have a wide base since the computation work changes throughout the duration of the simulation caused by dynamic mesh adaption. As the solution evolves, the flow features increase therefore requiring more mesh resolution which induces longer solve times. Lastly, the overset data update and grid connectivity times are shown in Fig. (7.1)(d). The time frequencies portray a skewed distribution ranging from 9.2 seconds to 11.5 seconds. However, there are a number of solve times that grow to 14 seconds.

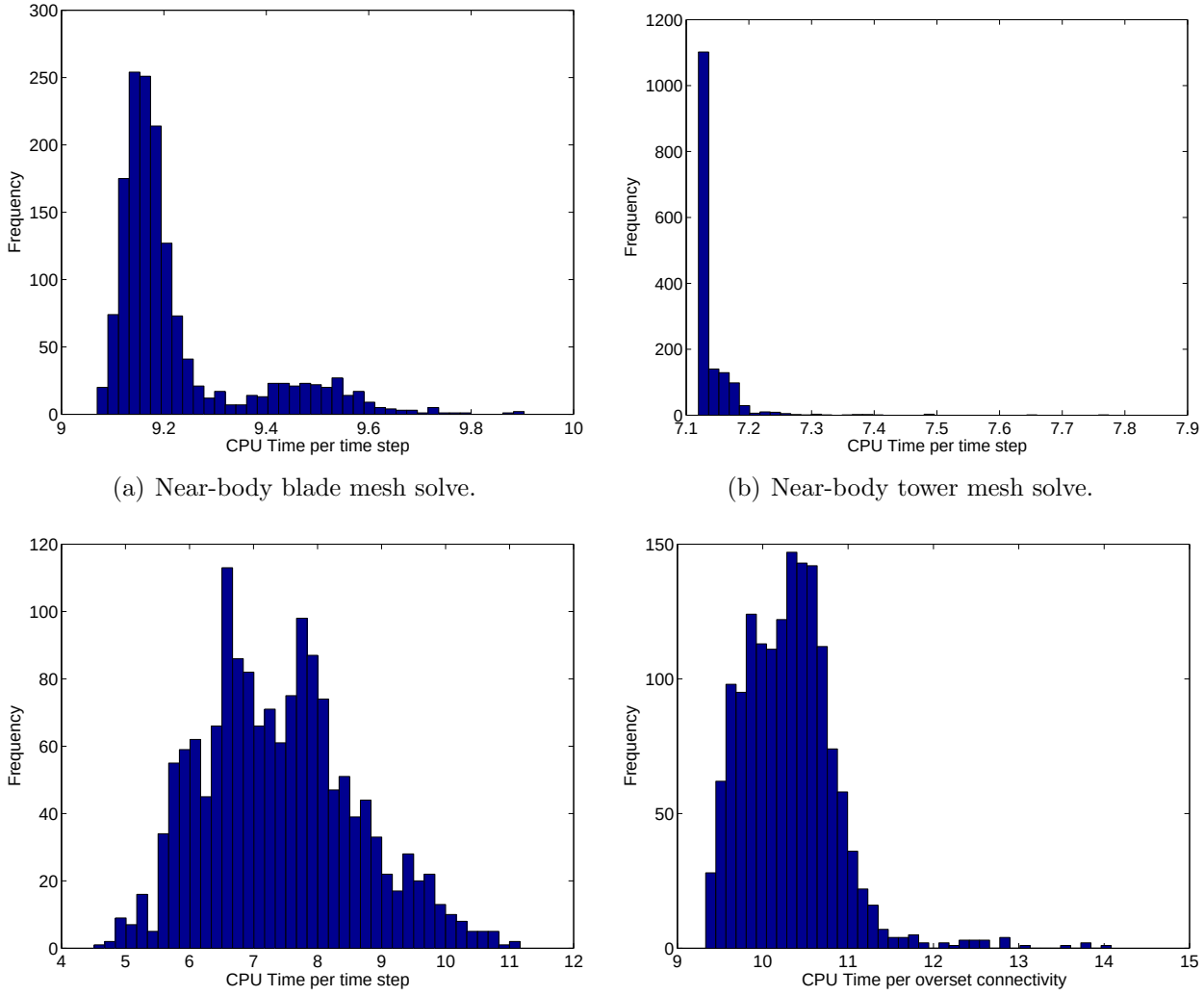
The smaller wind plant configurations demonstrate good performance for all components of the software but at larger wind turbine counts such as 48 and, particularly, 96, the solver time distribution width of the overset module widens. This indicates a small number of ranks in the large-scale simulations may be throttling the overall performance. In general, overset

methods incur larger scalability issues than flow solvers because of the inherent imbalance in the number of searches that need to be performed. To alleviate this issue, active load balancing techniques similar to those implemented in reference [156] need to be included and are planned as part of future work. Furthermore, overall efficiency improvement of the overset grid module is also desired since the execution time for dynamic overset grid assembly are on par currently with flow solution time, while more efficient approaches [156] have demonstrated overset grid assembly to take only one order of magnitude less time compared to flow solution time.

7.3 Longer Run-Time Simulation

A longer physical time simulation using the 48 wind turbine Lillgrund wind farm is simulated to 12 revolutions. The Lillgrund wind plant uses the Siemens SWT-2.3-93 wind turbine. The Lillgrund wind farm contains 48 wind turbines in an arrangement with downstream spacing of 4.3 diameters of the rotor and 3.3 diameters of side spacing. Uniform inflow conditions are used with a velocity of 10.9 m/s. The rotation rate of the rotor is taken as 16 revolutions per minute. Fig. (7.2) shows the wind plant configuration with iso-surfaces of velocity magnitude at approximately eight revolutions of rotation. Fig. (7.3) portrays a velocity magnitude slice for a row of the Lillgrund wind farm and a profile of the adaptive mesh refinement pattern in the wake of a wind turbine. The wake structure is tracked well downstream by the use of adaptive meshes.

Fig. (7.4) demonstrates the evolution of the total number of degrees-of-freedom for the off-body adaptive flow solver. Three linear trends are noticed in the DOFs. The initial cost of refining the off-body mesh to the same mesh resolution as the near-body meshes required for connecting the off-body mesh to all 48 near-body wind turbine meshes (four meshes per turbine) is approximately 300 million DOFs. From the start of the simulation to five revolutions, the DOFs sharply increase to approximately 1.2 billions degrees-of-freedom in a linear fashion representing the initial wake transients. The second linear trend represents the sustained wake growth as the simulation evolves over time. For wind turbine wake



(a) Near-body blade mesh solve.

(b) Near-body tower mesh solve.

(c) Adaptive off-body mesh solve.

(d) Overset update and connectivity.

Figure 7.1: Solver time frequency histograms (in seconds) of the 96 wind turbine case for the weak scaling study. Row 1 shows the near-body CFD solver times which run in parallel; row 2 shows the off-body CFD solver time and the overset data update and connectivity times. The CFD solvers must complete the time step before the overset module can interpolate the solutions between meshes therefore placing the execution process into two sequential components.

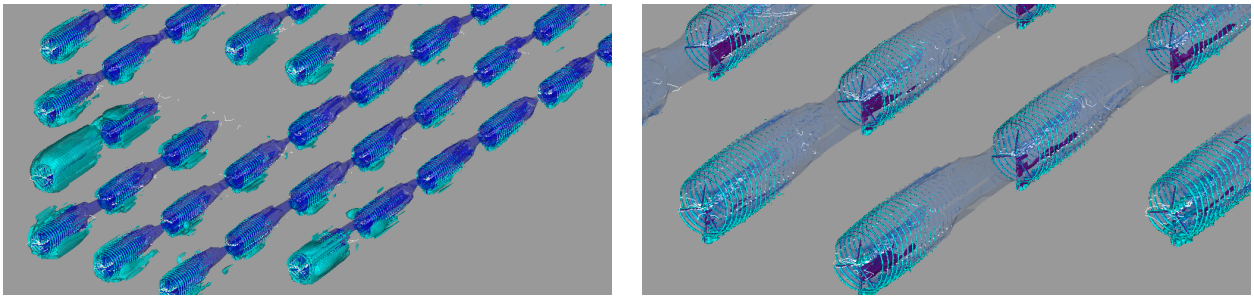
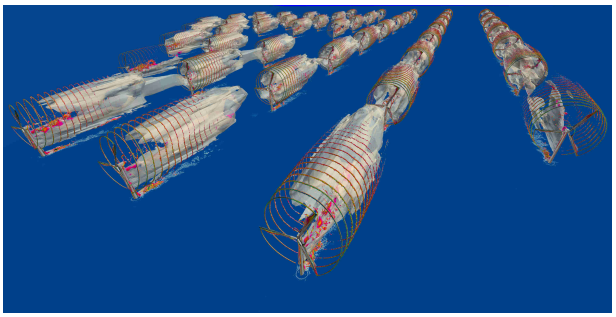
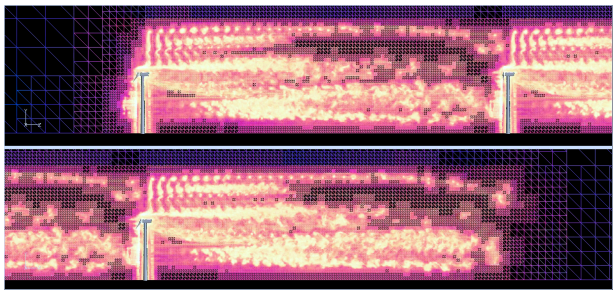


Figure 7.2: Iso-surfaces of velocity magnitude of the Lillgrund wind farm which contains 48 Siemens SWT-2.3-93 wind turbines.



(a) Velocity magnitude of a row of Lillgrund wind turbines.



(b) Adaptive mesh system of the wake of a Siemens turbine.

Figure 7.3: Lillgrund wind farm wake structures and adaptive mesh for the Siemens SWT-2.3-93 wind turbine.

interaction to occur between upstream and downstream wind turbines with a spacing of 4.3 rotor diameters, inflow velocity of 10.9 m/s at 16 revolutions per minute, approximately 36 seconds of physical time simulation are required. This corresponds to 9.78 revolutions of rotation which is exactly the location of the peak of DOFs in Fig. (7.4). The decreasing linear trend represents the time after which the wakes begin to interact. Under uniform inflow conditions, strong blade tip vorticies are formed invoking mesh refinement as demonstrated in Fig. (7.3)(b). Flow features reaching a user-specified threshold of Q-criterion magnitude, which is a measure of vorticity and mean-shear rate, are tagged for mesh refinement. Wake velocity deficits generated by uniform inflow conditions are much larger in comparison to turbulent inflow conditions because there is less entrainment of momentum by turbulent mixing. The reduced inflow velocities for downstream turbines generate weaker blade tip vorticies compared to wind turbines that do not have impinging wake inflow conditions resulting is less elements containing flow features that reach the refinement criterion threshold value. After the wakes impinge on the downstream wind turbines, less elements are tagged for refinement resulting in the observed decrease of DOFs.

7.4 Large-Scale Wind Plant Simulation of 144 Wind Turbines

A short run-time simulation using 144 wind turbines is also performed to stress test the parallel scalability. This simulation used 2.22 million nodes per blade with the 505,000 nodes per tower totaling approximately one billion degrees-of-freedom for the near-body mesh system. The off-body mesh contained 1.6 billion degrees-of-freedom using $p = 4$, fifth-order spacial accuracy elements giving a cumulative total of 2.6 billion DOFs across 62,208 processors. The simulation was executed for a small number of time steps to demonstrate the ability to run 577 independent meshes in an overset environment. Future work involves improving weak and strong parallel scalability of the framework. At such large scale with hundreds to thousands of solver instances in a single simulation, detailed concerns such as I/O of individual log files are required.

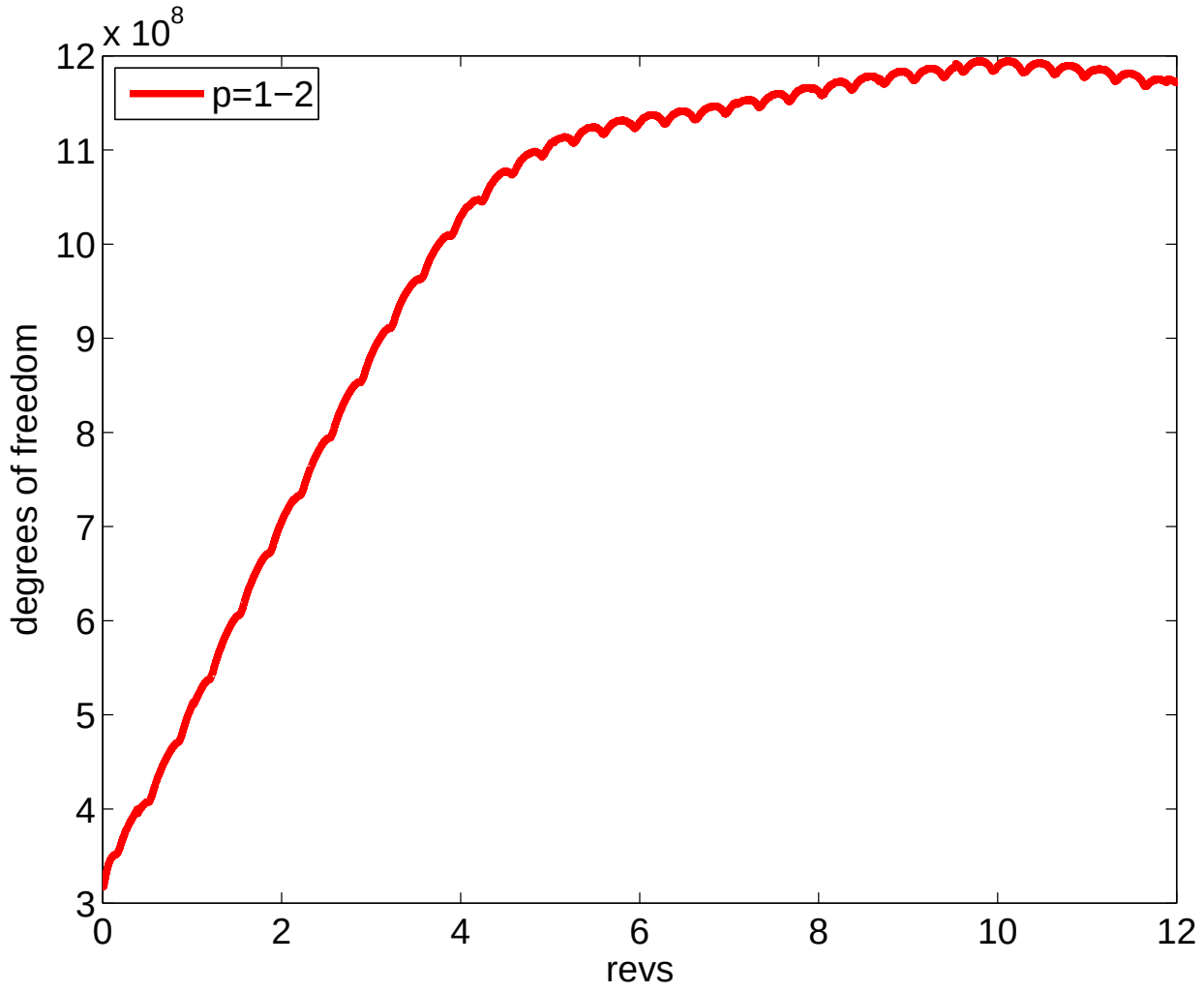


Figure 7.4: Degree of freedom counts for Lillgrund wind farm simulation. The initial linear trend corresponds to the start-up wake transients. The second linear trend corresponds to the sustained wake growth over the duration of the simulation. The last linear trend represents the interaction of the wakes between wind turbines. The peak represents the moment when the upstream wind turbine wake interacts with the downstream wind turbine.

Chapter 8

Conclusions

8.1 Summary

The discontinuous Galerkin (DG) method provides several computational advantages over traditional methods such as finite-difference or finite-volume methods. As demonstrated in this thesis, DG methods have exceptional computational efficiency and parallel scalability, by virtue of compact computational kernels with a nearest-neighbor communication stencil. By increasing the floating-point operations per degree-of-freedom in the computational kernel, the arithmetic intensity increases enabling higher computational performance.

Modern computer architectures can perform floating-point operations significantly faster than the rate at which memory can be read, thus favoring higher arithmetic intensity computations. DG methods at higher-orders of solution approximation are *compute-bound*, and, thus, faster than finite-difference and finite-volume methods per residual evaluation for problems with the same number of degrees-of-freedom. Additionally, DG methods offer multiple levels of parallelism embedded into the algorithm. Future supercomputing systems will require algorithms to contain multiple levels of granularity in parallelism to be able to fully utilize the computer architecture performance. Generally, the granularity of parallelism can be classified into *coarse-grain* and *fine-grain* parallelism. Coarse-grain parallelism is parallelism obtained through domain decomposition of the computational mesh. The main driver of high performance computing algorithm development has stemmed from this level of par-

allelism since the late 1980s. As shown in this work, DG methods are highly effective at coarse-grain parallelism. Leadership-class supercomputers nowadays are trending towards massive parallelism through use of lots of 'small' cores or threads, meaning there are hundreds or thousands of relatively slow computing units within a computing node working in tandem to solve a problem. This approach has become prominent as greater computational efficiency is achieved via higher floating point operation rates per unit of energy. This track of computing using large numbers of threads requires fine-grain parallelism, which is parallelism attained through loop level parallelism and/or vectorization, which is the process of operating on multiple data with a single instruction simultaneously. Since DG methods have higher-order computations within the mesh element, fine-grain parallelism is readily available, thus making DG discretizations excellent candidates for new and future computing architectures.

In addition to effective use of computing hardware, the development of split form DG methods with the summation-by-parts (SBP) property has offered increased robustness properties that previously plagued the method from wide-spread employment. Small alterations to the standard DG Spectral Element Method provides these high-order methods the ability to be applied to more difficult problems, while providing favorable properties that can be paired with the development of turbulence models, such as Large Eddy Simulation models.

Discontinuous Galerkin methods also offer a natural setting for the use of adaptive mesh refinement methods (AMR). By way of construction in the finite-element framework, DG methods enable the development for conservative and efficient discretizations that allow for hanging-node mesh interfaces required for AMR. Additionally, the DG method offers flexibility of polynomial degree inside of the mesh element, locally and independently, to achieve higher-order solution accuracy, which enables the development of a *hp*-adaptive solution techniques. To enable extreme-scale simulations that are inherently multiscale, AMR is essential for simulation tractability.

8.2 Contributions

This work has demonstrated the discontinuous Galerkin method can be successfully utilized for extreme-scale simulations in the field of computational fluid dynamics concerned with problems founded in aerospace and wind energy industries. Development of a robust, highly computationally efficient, and highly parallel scalable discretization has been executed for the compressible Navier-Stokes equations with a constant Smagorinsky Large Eddy Simulation turbulence model. The stand-alone solver was applied to standard problems such as the Ringleb flow problem, which was used for validation of the numerical discretization, a diagonally-driven cavity flow problem, and the Taylor-Green Vortex problem. The stand-alone solver was additionally parallel strong-scaled to over one million MPI ranks using over 500,000 cores. Implementation of the split form DG method with the summation-by-parts property was completed, and demonstrated to show superior robustness compared to the standard DG method.

Development of the DG method into a non-conforming adaptive mesh refinement framework has been successfully demonstrated with the optimal error convergence rate of $p + 1$ for a discretization order of p. This work was first to integrate the split form discontinuous Galerkin method with the summation-by-parts property into a dynamic adaptive mesh framework utilizing an *hp*-adaption strategy.

This work was successfully integrated into a larger software framework, W^2A^2KE3D, which enabled the simulation of aerospace and wind energy applications. Cases verified in the larger framework within the aerospace domain included a three-dimensional NACA0015 wing and a sphere using a low Reynolds number. The primary results of this work were derived from wind energy applications. Four single wind turbines were simulated: NREL 5MW, NREL Phase VI, Siemens SWT-2.3-93, and WindPACT-1.5MW. Additionally, this work contributed to the largest high-fidelity wind farm simulation, using full rotor models for wind turbines, to date.

8.3 Future Work

This work provides multiple avenues of future research and continued development. This includes stand-alone solver development through research of the discretizations and adaptive mesh refinement techniques. Further, future research related to the betterment of the W^2A^2KE3D framework directly pertaining to this work is illuminated. A non-exhaustive list for advancing this work is provided hereafter.

Fine-grain parallelism

As highlighted at the beginning of this chapter, discontinuous Galerkin methods multiple levels of parallelism. Coarse-grain parallelism through domain decomposition has been exploited through the Message-Passing-Interface (MPI) programming model. Fine-grain parallelism of the flow solver needs to be utilized for future computing architectures to fully scale the entire supercomputer system. The programming model for exa-scale era systems is predicted to be MPI + X, where X is any fine-grain programming model, e.g. CUDA, OpenMP. MPI will be used across the nodes in the supercomputing environment, and X will be employed inside the compute node. The prominent platforms during the time of this work are graphical processing units (GPU) and manycore architectures. Multiple choices for the fine-grain parallelism programming model are available, depending on the architecture.

Loop-level parallelism needs to explicitly instrumented using programming models specifically designed for the target architecture. Some programming models have abstracted the architecture type for performance portability. Sandia National Laboratory has developed the Kokkos [191], which is a C++ programming model using custom data templates. Another abstraction programming model is OCCA. OCCA [192] is an open-source library that facilitates programming in an environment containing different types of devices. Devices are abstracted which allows the user to pick at run-time, e.g. CPUs, GPUs, Xeon Phi, FPGAs.

In addition to explicit loop-level parallelism, memory-management and memory-movement require keen attention as it is key to obtaining high performance. Memory movement is extremely costly in comparison to arithmetic operations. Additionally, vectorization must be fully utilized through single-instruction-multiple-data (SIMD) procedures.

Split form method development

Split form discontinuous Galerkin methods with the summation-by-parts property have shown superior robustness compared to the standard discontinuous Galerkin Spectral Element method. Two split forms were implemented in this work: Kennedy & Gruber [131], and Pirozzoli [137].

Numerical investigation needs to be conducted for discovering which method is best suited for wind turbine simulations. Further, there are several other split forms in the literature such as Ducros *et al.* [130], Morinishi [193], Ismail and Roe [194], and Chandrashekar [195]. Gassner *et al* [135] demonstrated unique split forms can be arbitrarily constructed not only from the conservative variables of the Navier-Stokes equations but also from the entropy variables. Construction and investigation of new forms can provide a fruitful path for unique disretizations with favorable characteristics.

Second, split form methods for DG discretizations are relatively new [138], thus full comprehension of why these methods provide superior robustness is still undetermined. Winters *et al.* [196] has initiated research into the fundamental reasons for robustness of this method but still have not fully explored the topic.

Third, preliminary work by Flad *et al.* [197] has shown the ability to develop customized turbulence models tuned with the split form DG method to produce discretizations that demonstrate favorable turbulence spectra characteristics using a Large Eddy Simulation (LES) model. Research in this area is ongoing as this is one of the first approaches that provides correct LES turbulence modeling characteristics with the DG method.

Turbulence model development

The discontinuous Galerkin method provides numerical solutions of the Navier-Stokes equations with high-order accuracy with high computational efficiency. These attributes make the DG method highly attractive for LES. This work makes use of the simplest LES eddy viscosity model which assumes the Smagorinsky constant, C_s, is constant [89]. However, this approach is obviously inappropriate for most turbulent flow problems as eddy viscosity is always applied even when small-scale turbulence is not present. Thus, more suitable turbu-

lence models need to be investigated for vortical flows. Some examples of other LES eddy viscosity models are the Dynamic subgrid-scale model [198], the Dynamic Heinz model [199], and the wall-adapting local eddy-viscosity (WALE) model [200].

Error-based adaptive mesh refinement criterion

The work herein uses a feature-based strategy to drive the adaptive mesh refinement process. Adaptive mesh refinement using an error-based strategy for unsteady mesh adaption provides superior mechanisms for producing highly accurate solutions with efficient use of resources [201]. Not only do error-based strategies guide the adaption process but they also improve robustness of the solver. Further, sources of error may be categorized which may provide information in areas of spatial and temporal domains that are most responsible for error. Error-based strategies provide a rich field of research, and are essential for obtaining the most accurate solutions while most effectively placing resources into regions sourcing the most numerical error.

Temporal discretizations

Exploration of implicit or implicit-explicit (IMEX) temporal discretizations should be conducted to further exploit possible performance enhances through bypassing the explicit CFL condition that limits the time step size for the stability of the numerical discretization.

IMEX may be particularly effective when using spatial discretizations of higher-order accuracy on the finest level of the AMR mesh. Currently, for wind turbine simulation in the overset framework, the finest AMR level, which is used to connect to the near-body mesh, uses $p = 1$, second-order, elements to allow for a reasonable time step. If higher p-orders are used on the finest mesh level, the CFL condition will pose a serious time step restriction. By allowing the finest level of the mesh to be solved implicitly in time, the CFL condition is no longer required for stability of the method, allowing a larger time step. This would enable higher p-orders to be used on all levels of the AMR mesh system. Additionally, IMEX time discretization has been shown to save an order of magnitude in speedup when utilized in a dynamic adaptive mesh refinement setting with a DG method [49].

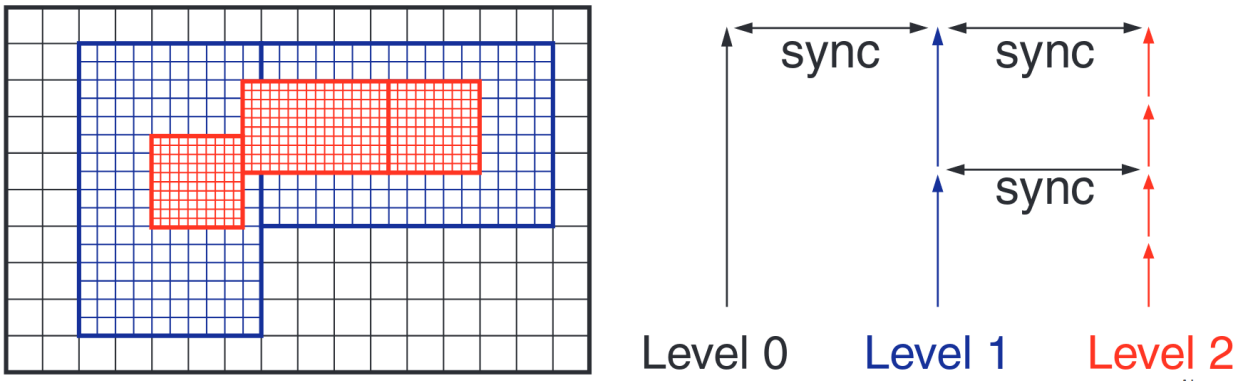


Figure 8.1: Time step sub-cycling between AMR levels. Image courtesy of AMReX.

Time step sub-cycling

As highlighted in previous section, explicit time stepping in the adaptive mesh refinement grid structure is currently limited by the CFL condition on the finest level of refinement in the mesh system. Alternatively, sub-cycling in time is the procedure of allowing each level to use their respective maximum allowable time step, where the coarsest level would take one time step, Δt_c, and the finer resolution mesh levels take multiple steps to get to Δt_c. After each sub-cycle, the fluxes between each level are synchronized. This procedure is shown in Fig. (8.1). The synchronization procedure requires correction of the fluxes as the finer levels of the mesh produce different fluxes than those calculated on the coarse levels. Time sub-cycling has been shown by Calhoun [202] to produce a 15% speedup in wall-clock time for two-dimensional problems by reducing the number of parallel communications and the number of solution point iterations.

Atmospheric boundary layer physics

Within the W^{2}A^{2}KE3D framework, atmospheric boundary layer (ABL) inflow is provided through a flow solver one-way coupling interface using precomputed solutions from either WRF or SOWFA. The atmospheric solution is registered with the overset assembler as an additional CFD solver with its own unstructured mesh. At every global time step in the CFD simulation, the overset assembler must reconnect all meshes, including this additional unstructured mesh, over the full computational domain. This additional mesh can add

complexity and computational expense to the overset assembly procedure. Thus, this procedure may not be fully parallel scalable to hundreds or thousands of turbines, or millions of computing cores.

To circumvent this issue, appropriate physics related to ABL simulation, such as thermal buoyancy, gravity, Coriolis effect, surface roughness, may be implemented into the off-body DG flow solver. Further, complex terrain may be incorporated through multiple pathways depending on the terrain resolution required. Unstructured mesh technologies could be applied to achieve high accuracy and resolution. Simpler techniques such as cut-cell or immersed boundary methods also serve as possible solutions. By using the off-body solver as the ABL solver, all benefits are retained including excellent computational efficiency and parallel scalability. Further, less solution data storage is required as the CFD solution and the ABL solution become one. Lastly, no additional grid is needed for the ABL solution to transfer the solution to the CFD solver through the overset assembler.

Appendix A

Wind Energy Aerodynamics

The primary results of this work are concerned with simulations of wind energy applications. Power and thrust for a wind turbine are provided herein. Table A.1 provides a reference of the relevant variables used to describe the aerodynamics.

Variables	[units]	Description
ω	$[rad/s]$	rotor rotational speed
r	$[m]$	rotor radius
V	$[m/s]$	velocity
ϕ	$[degrees]$	flow angle
P_t	$[N]$	tangential force
P_n	$[N]$	normal force
F_L	$[N]$	lift force
F_D	$[N]$	drag force
τ	$[N \cdot m]$	magnitude of torque
ρ	$[kg/m^3]$	air density
A	$[m^2]$	reference area
L	$[m]$	reference length
T	$[N]$	thrust
P	$[W]$	power
C_D		drag coefficient
C_D^M		drag moment coefficient

Table A.1: Wind turbine aerodynamics variables with descriptions and units.

Figure A.1: Horizontal axis wind turbine. The rotor plane diameter of the wind turbine is the diameter of the disk that the blades form when rotating. The height of of the wind turbine is the given by the height of tower which is the structure that holds the three turbine blades.

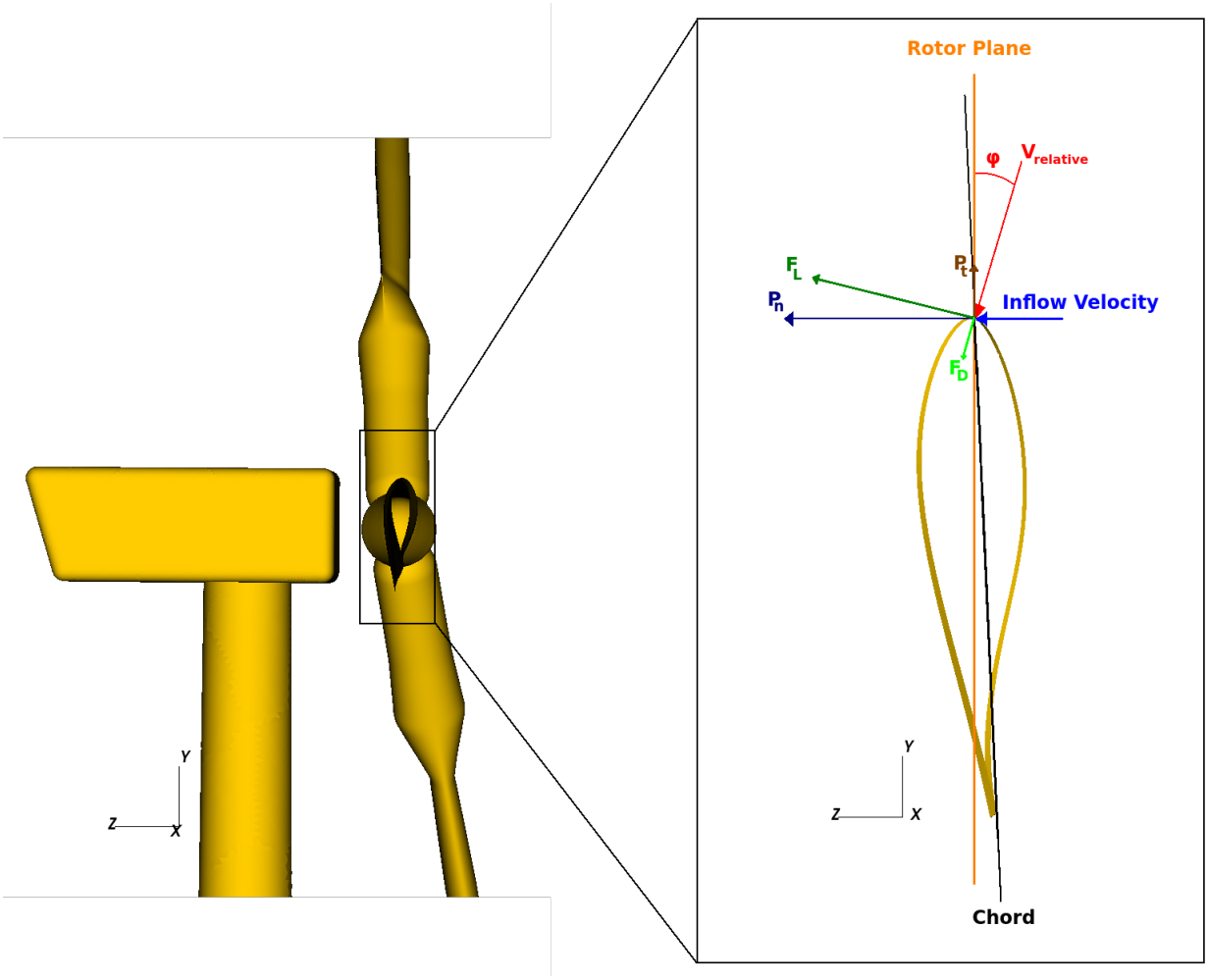


Figure A.2: Airfoil view of wind turbine blade with inflow in the $+z$-direction. Variables: ϕ-flow angle, $\vec{P}_n$-normal force (perpendicular to rotor plane), $\vec{P}_t$-tangent force (in the rotor plane), $\vec{F}_D$-drag force (in the relative velocity plane), $\vec{F}_L$-lift force (perpendicular to relative velocity plane).

A.1 Thrust

The thrust, T, of a horizontal axis wind turbine, using the variables defined in Table (A.1), can be calculated as follows:

$$T = \frac{1}{2}\rho V_{\text{rel}}^2 C_D \cdot A \cdot L \tag{A.1}$$

The drag coefficient, C_D, is calculated by the following:

$$C_D = \frac{F_D}{\frac{1}{2}\rho V_{\text{rel}}^2 A} \tag{A.2}$$

where F_D is the drag force on the wind turbine blades, shown in Fig. (A.2), and, V_{rel} is the tip speed of the wind turbine blade,

$$V_{\text{rel}} = u_{\text{tip}} = \omega \cdot r \tag{A.3}$$

where r is the blade radius. Thus, the thrust is calculated as:

$$\boxed{T = \frac{1}{2}\rho \left(\omega \cdot r\right)^2 C_D \cdot A \cdot L} \tag{A.4}$$

A.2 Power

The power, P, of a horizontal axis wind turbine can be calculated as follows:

$$P = \tau \cdot \omega \tag{A.5}$$

where τ is torque and ω is the rotor rotational speed. The rotor rotation rate is prescribed by the problem, but torque is an aerodynamic force that is calculated as follows:

$$\tau = \frac{1}{2}\rho V_{\text{rel}}^2 \cdot A \cdot C_D^M \tag{A.6}$$

where C_D^M is the drag moment coefficient. Thus, the power is calculated as follows:

$$\boxed{P = \frac{1}{2}\rho\omega \left(\omega \cdot r\right)^2 C_D^M \cdot A} \tag{A.7}$$

References

[1] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[2] S. Wolfram, *A new kind of science*. Wolfram media Champaign, 2002, vol. 5.

[3] J. E. Thornton, "The cdc 6600 project," *Annals of the History of Computing*, vol. 2, no. 4, pp. 338–348, 1980.

[4] F. Hossfeld, "Vector-supercomputers," *Parallel Computing*, vol. 7, no. 3, pp. 373–385, 1988.

[5] R. M. Russell, "The cray-1 computer system," *Communications of the ACM*, vol. 21, no. 1, pp. 63–72, 1978.

[6] G. H. Barnes, R. M. Brown, M. Kato, D. J. Kuck, D. L. Slotnick, and R. A. Stokes, "The illiac iv computer," *IEEE Transactions on computers*, vol. 100, no. 8, pp. 746–757, 1968.

[7] P. J. Denning, "The science of computing: Speeding up parallel processing," *American Scientist*, vol. 76, no. 4, pp. 347–349, 1988.

[8] G. Bell, D. H. Bailey, J. Dongarra, A. H. Karp, and K. Walsh, "A look back on 30 years of the gordon bell prize," *The International Journal of High Performance Computing Applications*, vol. 31, no. 6, pp. 469–484, 2017.

[9] "Top 500, the list: November 2017," 2017, https://www.top500.org/lists/2017/11/.

[10] A. R. Mitchell and D. F. Griffiths, *The finite difference method in partial differential equations*. John Wiley, 1980.

[11] C. Hirsch, *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Butterworth-Heinemann, 2007.

[12] H. Lomax, T. H. Pulliam, and D. W. Zingg, *Fundamentals of computational fluid dynamics*. Springer Science & Business Media, 2013.

[13] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[14] A. Jameson and D. Mavriplis, "Finite volume solution of the two-dimensional euler equations on a regular triangular mesh," *AIAA journal*, vol. 24, no. 4, pp. 611–618, 1986.

[15] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, *The development of discontinuous Galerkin methods*. Springer, 2000.

[16] H. Luo, J. D. Baum, and R. Löhner, "A discontinuous galerkin method based on a taylor basis for the compressible flows on arbitrary grids," *Journal of Computational Physics*, vol. 227, no. 20, pp. 8875–8893, 2008.

[17] M. Ceze and K. J. Fidkowski, "Drag prediction using adaptive discontinuous finite elements," *Journal of Aircraft*, vol. 51, no. 4, pp. 1284–1294, 2014.

[18] D. L. Darmofal, S. R. Allmaras, M. Yano, and J. Kudo, "An adaptive, higher-order discontinuous galerkin finite element method for aerodynamics," aIAA Paper 2013-2871, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 2013.

[19] T. Haga, H. Gao, and Z. Wang, "A high-order unifying discontinuous formulation for the navier-stokes equations on 3d mixed grids," *Mathematical Modelling of Natural Phenomena*, vol. 6, no. 03, pp. 28–56, 2011.

[20] R. Hartmann, "Higher-order and adaptive discontinuous galerkin methods with shock-capturing applied to transonic turbulent delta wing flow," *International Journal for Numerical Methods in Fluids*, vol. 72, no. 8, pp. 883–894, 2013.

[21] M. J. Brazell and D. J. Mavriplis, "3d mixed element discontinuous galerkin with shock capturing," aIAA Paper 2013-3064, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA., June 2013.

[22] L. Wang, W. K. Anderson, J. T. Erwin, and S. Kapadia, "Discontinuous galerkin and petrov galerkin methods for compressible viscous flows," *Computers & Fluids*, vol. 100, pp. 13–29, 2014.

[23] R. S. Glasby, N. Burgess, K. Anderson, L. Wang, S. Allmaras, and D. Mavriplis, "Comparison of su/pg and dg finite-element techniques for the compressible navier-stokes equations on anisotropic unstructured meshes," aIAA Paper 2013-691, 51st AIAA Aerospace Sciences Meeting, Grapevine, TX, January 2013.

[24] H. Huynh and N. Kroll, "Third international workshop on high-order cfd methods," https://www.grc.nasa.gov/hiocfd/.

[25] M. D. Lam, E. E. Rothberg, and M. E. Wolf, "The cache performance and optimizations of blocked algorithms," in *ACM SIGARCH Computer Architecture News*, vol. 19, no. 2. ACM, 1991, pp. 63–74.

[26] P. Vincent, F. D. Witherden, A. M. Farrington, G. Ntemos, B. C. Vermeire, J. S. Park, and A. S. Iyer, "Pyfr: Next-generation high-order computational fluid dynamics on many-core hardware," in *22nd AIAA Computational Fluid Dynamics Conference*, 2015, p. 3050.

[27] F. D. Witherden, B. C. Vermeire, and P. E. Vincent, "Heterogeneous computing on mixed unstructured grids with pyfr," *Computers & Fluids*, vol. 120, pp. 173–186, 2015.

[28] R. D. Hornung, A. M. Wissink, and S. R. Kohn, "Managing complex data and geometry in parallel structured amr applications," *Engineering with Computers*, vol. 22, no. 3-4, pp. 181–195, 2006.

[29] M. Adams, P. Colella, D. T. Graves, J. Johnson, N. Keen, T. J. Ligocki, D. F. Martin, P. McCorquodale, D. M. P. Schwartz, T. Sternberg, and B. V. Straalen, "Chombo software package for amr applications design document," 2014, lawrence Berkeley National Laboratory Technical Report LBNL-6616E.

[30] M. Adams, P. O. Schwartz, H. Johansen, P. Colella, T. J. Ligocki, D. Martin, N. Keen, D. Graves, D. Modiano, B. Van Straalen *et al.*, "Chombo software package for amr applications-design document," Tech. Rep., 2015.

[31] A. Wissink, S. Kamkar, T. Pulliam, J. Sitaraman, and V. Sankaran, "Cartesian adaptive mesh refinement for rotorcraft wake resolution," aIAA Paper 2010-4554, 28th AIAA Applied Aerodynamics Conference, Chicago, IL, June 2010. [Online]. Available: http://people.nas.nasa.gov/~pulliam/mypapers/AIAA_2010-4554.pdf

[32] A. Wissink, B. Jayaraman, A. Datta, J. Sitaraman, M. Potsdam, S. Kamkar, D. Mavriplis, Z. Yang, R. Jain, J. Lim *et al.*, "Capability enhancements in version 3 of the helios high-fidelity rotorcraft simulation code," aIAA Paper 2012-713, 50th AIAA Aerospace Sciences Meeting, Nashville, TN, January 2012.

[33] E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, "Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations," *Journal of computational physics*, vol. 161, no. 1, pp. 35–60, 2000.

[34] C. S. Peskin, "The immersed boundary method," *Acta numerica*, vol. 11, pp. 479–517, 2002.

[35] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, vol. 37, pp. 239–261, 2005.

[36] M. J. Aftosmis, "Solution adaptive cartesian grid methods for aerodynamic flows with complex geometries," *VKI Lecture Series*, vol. 2, 1997.

[37] D. M. Ingram, D. M. Causon, and C. G. Mingham, "Developments in cartesian cut cell methods," *Mathematics and Computers in Simulation*, vol. 61, no. 3, pp. 561–572, 2003.

[38] D. D. Marshall and S. M. Ruffin, "A new inviscid wall boundary condition treatment for embedded boundary cartesian grid schemes," aIAA Paper 2004-583, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2004.

[39] J. Sitaraman, D. J. Mavriplis, and E. P. Duque, "Wind farm simulations using a full rotor model for wind turbines," in *32nd ASME Wind Energy Symposium*, 2014, p. 1086.

[40] K. J. Fidkowski, "A high-order discontinuous galerkin multigrid solver for aerodynamic applications," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[41] L. T. Diosady and S. M. Murman, "Design of a variational multiscale method for turbulent compressible flows," AIAA Paper 2013-2870, 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, June 2014.

[42] N. Burgess and D. Mavriplis, "An hp-adaptive discontinuous galerkin solver for aerodynamic flows on mixed-element meshes," aIAA Paper 2011-490, 49th AIAA Aerospace Sciences Meeting and Exhibit, Orlando FL, January 4-7, 2011.

[43] L. Wang and D. J. Mavriplis, "Adjoint-based h–p adaptive discontinuous galerkin methods for the 2d compressible euler equations," *Journal of Computational Physics*, vol. 228, no. 20, pp. 7643–7661, 2009.

[44] B. Reza Ahrabi, W. K. Anderson, and J. C. Newman, "High-order finite-element method and dynamic adaptation for two-dimensional laminar and turbulent navier-stokes," aIAA Paper 2014-2983, 32nd AIAA Applied Aerodynamics Conference, Atlanta, GA., June 2014.

[45] F. Hindenlang, G. J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, and C.-D. Munz, "Explicit discontinuous galerkin methods for unsteady problems," *Computers & Fluids*, vol. 61, pp. 86–93, 2012.

[46] G. K. El Khoury, P. Schlatter, A. Noorani, P. F. Fischer, G. Brethouwer, and A. V. Johansson, "Direct numerical simulation of turbulent pipe flow at moderately high reynolds numbers," *Flow, turbulence and combustion*, vol. 91, no. 3, pp. 475–495, 2013.

[47] M. Berger, "Adaptive mesh refinement for time-dependent partial differential equations," Ph.D. dissertation, Ph. d. dissertation, Stanford University, 1982. Computer Science Report No. STAN-CS-82-924, 1982.

[48] K. Schaal, A. Bauer, P. Chandrashekar, R. Pakmor, C. Klingenberg, and V. Springel, "Astrophysical hydrodynamics with a high-order discontinuous galerkin scheme and adaptive mesh refinement," *Monthly Notices of the Royal Astronomical Society*, vol. 453, no. 4, pp. 4278–4300, 2015.

[49] M. A. Kopera and F. X. Giraldo, "Analysis of adaptive mesh refinement for imex discontinuous galerkin solutions of the compressible euler equations with application to atmospheric simulations," *Journal of Computational Physics*, vol. 275, pp. 92–117, 2014.

[50] S. Blaise and A. St-Cyr, "A dynamic hp-adaptive discontinuous galerkin method for shallow-water flows on the sphere with application to a global tsunami simulation," *Monthly Weather Review*, vol. 140, no. 3, pp. 978–996, 2012.

[51] S. Blaise, A. St-Cyr, D. Mavriplis, and B. Lockwood, "Discontinuous galerkin unsteady discrete adjoint method for real-time efficient tsunami simulations," *Journal of Computational Physics*, vol. 232, no. 1, pp. 416–430, 2013.

[52] P. Solin, K. Segeth, and I. Dolezel, *Higher-Order Finite Element Methods*. CRC Press, 2004.

[53] A. M. Wissink, M. Potsdam, V. Sankaran, J. Sitaraman, and D. Mavriplis, "A dual-mesh unstructured adaptive cartesian computational fluid dynamics approach for hover prediction," *Journal of the American Helicopter Society*, vol. 61, no. 1, pp. 1–19, 2016.

[54] A. M. Wissink, B. Jayaraman, and J. Sitaraman, "An assessment of the dual mesh paradigm using different near-body solvers in helios," in *55th AIAA Aerospace Sciences Meeting*, 2017, p. 0287.

[55] "Wind vision: A new era for wind power in the united states," Technical report, US Department of Energy, Washington, DC, Tech. Rep., 2015.

[56] P. Messina, "Exascale computing project," 2016, exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. [Online]. Available: https://exascaleproject.org

[57] "Turbine wind plant efficiency," 2016. [Online]. Available: http://www.nrel.gov/news/press/2016/37739

[58] "Nrel to lead one exascale computing project," 2016. [Online]. Available: https://phys.org/news/2016-10-nrel-exascale.html

[59] P. Fleming, P. Gebraad, J.-W. van Wingerden, S. Lee, M. Churchfield, A. Scholbrock, J. Michalakes, K. Johnson, and P. Moriarty, "Sowfa super-controller: A high-fidelity tool for evaluating wind plant control approaches," National Renewable Energy Laboratory (NREL), Golden, CO., Tech. Rep., 2013.

[60] P. A. Fleming, P. M. Gebraad, S. Lee, J.-W. van Wingerden, K. Johnson, M. Churchfield, J. Michalakes, P. Spalart, and P. Moriarty, "Evaluating techniques for redirecting turbine wakes using sowfa," *Renewable Energy*, vol. 70, pp. 211–218, 2014.

[61] P. Gebraad, F. Teeuwisse, J. Wingerden, P. A. Fleming, S. Ruben, J. Marden, and L. Pao, "Wind plant power optimization through yaw control using a parametric model for wake effectsa cfd simulation study," *Wind Energy*, vol. 19, no. 1, pp. 95–114, 2016.

[62] P. Gebraad, J. J. Thomas, A. Ning, P. Fleming, and K. Dykes, "Maximization of the annual energy production of wind power plants by optimization of layout and yaw-based wake control," *Wind Energy*, vol. 20, no. 1, pp. 97–107, 2017.

[63] N. Troldborg, J. N. Srensen, and R. Mikkelsen, "Actuator line simulation of wake of wind turbine operating in turbulent inflow," *Journal of Physics: Conference Series*, vol. 75, no. 1, p. 012063, 2007. [Online]. Available: http://stacks.iop.org/1742-6596/75/i=1/a=012063

[64] M. Churchfield, S. Lee, and P. Moriarty, "Overview of the simulator for offshore wind farm application sowfa," 2012.

[65] M. Churchfield, Q. Wang, A. Scholbrock, T. Herges, T. Mikkelsen, and M. Sjöholm, "Using high-fidelity computational fluid dynamics to help design a wind turbine wake measurement experiment," in *Journal of Physics: Conference Series*, vol. 753. IOP Publishing, 2016, p. 032009.

[66] R. Mikkelsen, "Actuator disc methods applied to wind turbines," Ph.D. dissertation, Technical University of Denmark, 2003.

[67] J. N. Sørensen and A. Myken, "Unsteady actuator disc model for horizontal axis wind turbines," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 39, no. 1-3, pp. 139–149, 1992.

[68] K. Takizawa, B. Henicke, T. E. Tezduyar, M.-C. Hsu, and Y. Bazilevs, "Stabilized space–time computation of wind-turbine rotor aerodynamics," *Computational Mechanics*, vol. 48, no. 3, pp. 333–344, 2011.

[69] Y. Bazilevs, M.-C. Hsu, I. Akkerman, S. Wright, K. Takizawa, B. Henicke, T. Spielman, and T. Tezduyar, "3d simulation of wind turbine rotors at full scale. part i: Geometry modeling and aerodynamics," *International Journal for Numerical Methods in Fluids*, vol. 65, no. 1-3, pp. 207–235, 2011.

[70] M. A. Potsdam and D. J. Mavriplis, "Unstructured mesh cfd aerodynamic analysis of the nrel phase vi rotor," aIAA Paper 2009-1221, 47th AIAA Aerospace Sciences Meeting, Orlando, FL, January 2009.

[71] N. N. Sørensen, J. Michelsen, and S. Schreck, "Navier-stokes predictions of the nrel phase vi rotor in the nasa ames 80 ft× 120 ft wind tunnel," *Wind Energy*, vol. 5, no. 2-3, pp. 151–169, 2002.

[72] E. P. Duque, M. D. Burklund, and W. Johnson, "Navier-stokes and comprehensive analysis performance predictions of the nrel phase vi experiment," *Journal of Solar Energy Engineering*, vol. 125, no. 4, pp. 457–467, 2003.

[73] A. L. Pape and J. Lecanu, "3d navier–stokes computations of a stall-regulated wind turbine," *Wind Energy*, vol. 7, no. 4, pp. 309–324, 2004.

[74] S. Gomez-Iradi, R. Steijl, and G. Barakos, "Development and validation of a cfd technique for the aerodynamic analysis of hawt," *Journal of Solar Energy Engineering*, vol. 131, no. 3, p. 031009, 2009.

[75] F. Zahle, N. N. Sørensen, and J. Johansen, "Wind turbine rotor-tower interaction using an incompressible overset grid method," *Wind Energy*, vol. 12, no. 6, pp. 594–619, 2009.

[76] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger, "3d simulation of wind turbine rotors at full scale. part ii: Fluid–structure interaction modeling with composite blades," *International Journal for Numerical Methods in Fluids*, vol. 65, no. 1-3, pp. 236–253, 2011.

[77] C. Gundling, B. Roget, and J. Sitaraman, "Prediction of wind turbine performance and wake losses using analysis methods of incremental complexity," aIAA Paper 2011-458, 49th AIAA Aerospace Sciences Meeting, Orlando, FL, January 2011.

[78] C. Gundling, B. Roget, J. Sitaraman, and R. Rai, "Comparison of wind turbine wakes in steady and turbulent inflow," aIAA Paper 2012-899, 50th AIAA Aerospace Sciences Meeting, Nashville, TN, January 2012.

[79] R. K. Rai, H. Gopalan, J. W. Naughton, and S. Heinz, "A study of the sensitivity of wind turbine response to inflow temporal and spatial scales," 2012.

[80] Y. Li, K.-J. Paik, T. Xing, and P. M. Carrica, "Dynamic overset cfd simulations of wind turbine aerodynamics," *Renewable Energy*, vol. 37, no. 1, pp. 285–298, 2012.

[81] M. M. Yelmule and E. A. Vsj, "Cfd predictions of nrel phase vi rotor experiments in nasa/ames wind tunnel," *International Journal of Renewable Energy Research (IJRER)*, vol. 3, no. 2, pp. 261–269, 2013.

[82] H. Gopalan, C. Gundling, K. Brown, B. Roget, J. Sitaraman, J. D. Mirocha, and W. O. Miller, "A coupled mesoscale–microscale framework for wind resource estimation and farm aerodynamics," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 132, pp. 13–26, 2014.

[83] C. Gundling, J. Sitaraman, B. Roget, and P. Masarati, "Application and validation of incrementally complex models for wind turbine aerodynamics, isolated wind turbine in uniform inflow conditions," *Wind Energy*, vol. 18, no. 11, pp. 1893–1916, 2015.

[84] J. L. Steger, F. C. Dougherty, and J. A. Benek, "A chimera grid scheme.[multiple overset body-conforming mesh system for finite difference adaptation to complex aircraft configurations]," 1983.

[85] R. Noack, "Suggar: a general capability for moving body overset grid assembly," in *17th AIAA Computational Fluid Dynamics Conference*, 2005, p. 5117.

[86] M. J. Brazell, J. Sitaraman, and D. J. Mavriplis, "An overset mesh approach for 3d mixed element high-order discretizations," *Journal of Computational Physics*, vol. 322, pp. 33–51, 2016.

[87] J. A. Crabill, J. Sitaraman, and A. Jameson, "A high-order overset method on moving and deforming grids," in *AIAA Modeling and Simulation Technologies Conference*, 2016, p. 3225.

[88] R. Benzi, S. Ciliberto, R. Tripiccione, C. Baudet, F. Massaioli, and S. Succi, "Extended self-similarity in turbulent flows," *Physical review E*, vol. 48, no. 1, p. R29, 1993.

[89] J. Smagorinsky, "General circulation experiments with the primitive equations: I. the basic experiment," *Monthly weather review*, vol. 91, no. 3, pp. 99–164, 1963.

[90] A. Favre, "The equations of compressible turbulent gases," AIX-MARSEILLE UNIV (FRANCE) INST DE MECANIQUE STATISTIQUE DE LA TURBULENCE, Tech. Rep., 1965.

[91] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, "A dynamic subgrid-scale eddy viscosity model," *Physics of Fluids A: Fluid Dynamics*, vol. 3, no. 7, pp. 1760–1765, 1991.

[92] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," *Mathematics of computation*, vol. 23, no. 106, pp. 221–230, 1969.

[93] S. A. Orszag, "Spectral methods for problems in complex geometries," *Journal of Computational Physics*, vol. 37, no. 1, pp. 70–92, 1980.

[94] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[95] A. Harten, P. D. Lax, and B. v. Leer, "On upstream differencing and godunov-type schemes for hyperbolic conservation laws," *SIAM review*, vol. 25, no. 1, pp. 35–61, 1983.

[96] R. Hartmann and P. Houston, "An optimal order interior penalty discontinuous galerkin discretization of the compressible navier-stokes equations," *J. Comput. Phys.*, vol. 227, no. 22, pp. 9670 – 85, 2008/11/20. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2008.07.015

[97] K. Shahbazi, D. Mavriplis, and N. Burgess, "Multigrid algorithms for high-order discontinuous galerkin discretizations of the compressible navier-stokes equations," *J. Comput. Phys.*, vol. 228, no. 21, pp. 7917 – 40, 2009/11/20. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2009.07.013

[98] D. A. Kopriva and G. Gassner, "On the quadrature and weak form choices in collocation type discontinuous galerkin spectral element methods," *Journal of Scientific Computing*, vol. 44, no. 2, pp. 136–155, 2010.

[99] D. Kahaner, C. Moler, and S. Nash, "Numerical methods and software," *Englewood Cliffs: Prentice Hall, 1989*, vol. 1, 1989.

[100] A. Jameson, W. Schmidt, E. Turkel *et al.*, "Numerical solutions of the euler equations by finite volume methods using runge-kutta time-stepping schemes," aIAA Paper 1981-1259, 14th Fluid and Plasma Dynamics Conference, Palo Alto, CA, June 1981.

[101] J. C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.

[102] S. Ruuth, "Global optimization of explicit strong-stability-preserving runge-kutta methods," *Mathematics of Computation*, vol. 75, no. 253, pp. 183–207, 2006.

[103] C.-W. Shu, "Total-variation-diminishing time discretizations," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 6, pp. 1073–1084, 1988.

[104] S. Gottlieb, C.-W. Shu, and E. Tadmor, "Strong stability-preserving high-order time discretization methods," *SIAM review*, vol. 43, no. 1, pp. 89–112, 2001.

[105] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods: fundamentals in single domains*. Springer, 2006.

[106] G. Strang and G. J. Fix, *An analysis of the finite element method*. Prentice-hall Englewood Cliffs, NJ, 1973, vol. 212.

[107] R. M. Kirby and G. E. Karniadakis, "De-aliasing on non-uniform grids: algorithms and applications," *Journal of Computational Physics*, vol. 191, no. 1, pp. 249–264, 2003.

[108] G. J. Gassner and A. D. Beck, "On the accuracy of high-order discretizations for underresolved turbulence simulations," *Theoretical and Computational Fluid Dynamics*, vol. 27, no. 3-4, pp. 221–237, 2013.

[109] R. C. Moura, S. J. Sherwin, and J. Peiró, "On dg-based iles approaches at very high reynolds numbers," *Report, Research Gate*, 2015.

[110] W. Koepf and D. Schmersau, "Representations of orthogonal polynomials," *Journal of Computational and Applied Mathematics*, vol. 90, no. 1, pp. 57–94, 1998.

[111] P.-O. Persson and J. Peraire, "Sub-cell shock capturing for discontinuous galerkin methods," *AIAA paper*, vol. 112, p. 2006, 2006.

[112] B. Cockburn and C.-W. Shu, "Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws. ii. general framework," *Mathematics of computation*, vol. 52, no. 186, pp. 411–435, 1989.

[113] X. Zhang and C.-W. Shu, "On positivity-preserving high order discontinuous galerkin schemes for compressible euler equations on rectangular meshes," *Journal of Computational Physics*, vol. 229, no. 23, pp. 8918–8934, 2010.

[114] 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*, vol. 131, no. 2, pp. 267–279, 1997.

[115] H. Yoshihara, H. Norstrud, J. Boerstoel, G. Chiocchia, and D. Jones, "Test cases for inviscid flow field methods." ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE), Tech. Rep., 1985.

[116] G. Taylor and A. Green, "Large ones," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 158, no. 895, pp. 499–521, 1937.

[117] M. Brachet, "Direct simulation of three-dimensional turbulence in the taylorgreen vortex," *Fluid Dynamics Research*, vol. 8, no. 1, pp. 1–8, 1991.

[118] A. Povitsky, "High-incidence 3-d lid-driven cavity flow," aIAA Paper 2001-2847, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, CA, June 2001.

[119] Y. Feldman and A. Y. Gelfgat, "From multi-to single-grid cfd on massively parallel computers: Numerical experiments on lid-driven flow in a cube using pressure–velocity coupled formulation," *Computers & Fluids*, vol. 46, no. 1, pp. 218–223, 2011.

[120] D. d'Humières, "Multiple–relaxation–time lattice boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[121] A. Gelfgat and Y. Feldman, "Reply to a letter of a. povitsky regarding benchmark problem of 3d flow in a cubic cavity driven by a diagonally moving lid," *Computers & Fluids*, vol. 92, p. 224, 2014.

[122] F. Ringleb, "Exakte loesungen der differentialgleichungen einer adiabatischen gasstroemung," *A. Angew. Math. Mech.*, vol. 20, no. 4, pp. 185–198, 1940.

[123] S. Albensoeder and H. C. Kuhlmann, "Accurate three-dimensional lid-driven cavity flow," *Journal of Computational Physics*, vol. 206, no. 2, pp. 536–558, 2005.

[124] A. Povitsky, "Three-dimensional flow in cavity at yaw," 2001.

[125] W. van Ress, A. Leonard, D. Pullin, and P. Koumoutsakos, "A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high reynolds number," *J. Comput. Phys.*, vol. 230, pp. 2794–2805, 2011.

[126] *Yellowstone: IBM iDataPlex System (Climate Simulation Laboratory)*, National Center for Atmospheric Research, Boulder, CO, 2012, http://n2t.net/ark:/85065/d7wd3xhc.

[127] A. Jameson, "Formulation of kinetic energy preserving conservative schemes for gas dynamics and direct numerical simulation of one-dimensional viscous compressible flow in a shock tube using entropy and kinetic energy preserving schemes," *Journal of Scientific Computing*, vol. 34, no. 2, pp. 188–208, 2008.

[128] B. Strand, "Summation by parts for finite difference approximations for d/dx," *Journal of Computational Physics*, vol. 110, no. 1, pp. 47–67, 1994.

[129] M. H. Carpenter, T. C. Fisher, E. J. Nielsen, and S. H. Frankel, "Entropy stable spectral collocation schemes for the navier–stokes equations: Discontinuous interfaces," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. B835–B867, 2014.

[130] F. Ducros, F. Laporte, T. Souleres, V. Guinot, P. Moinat, and B. Caruelle, "High-order fluxes for conservative skew-symmetric-like schemes in structured meshes: application to compressible flows," *Journal of Computational Physics*, vol. 161, no. 1, pp. 114–139, 2000.

[131] C. A. Kennedy and A. Gruber, "Reduced aliasing formulations of the convective terms within the navier–stokes equations for a compressible fluid," *Journal of Computational Physics*, vol. 227, no. 3, pp. 1676–1700, 2008.

[132] A. Kravchenko and P. Moin, "On the effect of numerical errors in large eddy simulations of turbulent flows," *Journal of Computational Physics*, vol. 131, no. 2, pp. 310–322, 1997.

[133] K. Black, "A conservative spectral element method for the approximation of compressible fluid flow," *Kybernetika*, vol. 35, no. 1, pp. 133–146, 1999.

[134] T. C. Fisher and M. H. Carpenter, "High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains," *Journal of Computational Physics*, vol. 252, pp. 518–557, 2013.

[135] G. J. Gassner, A. R. Winters, and D. A. Kopriva, "Split form nodal discontinuous galerkin schemes with summation-by-parts property for the compressible euler equations," *Journal of Computational Physics*, vol. 327, pp. 39–66, 2016.

[136] D. A. Kopriva and G. J. Gassner, "An energy stable discontinuous galerkin spectral element discretization for variable coefficient advection problems," *SIAM Journal on Scientific Computing*, vol. 36, no. 4, pp. A2076–A2099, 2014.

[137] S. Pirozzoli, "Numerical methods for high-speed flows," *Annual review of fluid mechanics*, vol. 43, pp. 163–194, 2011.

[138] G. J. Gassner, "A kinetic energy preserving nodal discontinuous galerkin spectral element method," *International Journal for Numerical Methods in Fluids*, vol. 76, no. 1, pp. 28–50, 2014.

[139] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[140] W. Zhang, A. Almgren, M. Day, T. Nguyen, J. Shalf, and D. Unat, "Boxlib with tiling: an amr software framework," *arXiv preprint arXiv:1604.03570*, 2016.

[141] P. MacNeice, K. M. Olson, C. Mobarry, R. De Fainchtein, and C. Packer, "Paramesh: A parallel adaptive mesh refinement community toolkit," *Computer physics communications*, vol. 126, no. 3, pp. 330–354, 2000.

[142] A. M. Wissink, R. D. Hornung, S. R. Kohn, S. S. Smith, and N. Elliott, "Large scale parallel structured amr calculations using the samrai framework," in *Supercomputing, ACM/IEEE 2001 Conference*. IEEE, 2001, pp. 22–22.

[143] *AMReX An adaptive mesh refinement software framework, Users Guide*, https://amrex-codes.github.io/AMReXUsersGuide.pdf.

[144] C. Burstedde, O. Ghattas, M. Gurnis, T. Isaac, G. Stadler, T. Warburton, and L. Wilcox, "Extreme-scale amr," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010, pp. 1–12.

[145] C. Burstedde, L. C. Wilcox, and O. Ghattas, "p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[146] G. Fubini, "Sugli integrali multipli," *Rend. Acc. Naz. Lincei*, vol. 16, pp. 608–614, 1907.

[147] D. J. Mavriplis, "Grid resolution study of a drag prediction workshop configuration using the nsu3d unstructured mesh solver," aIAA Paper 2005-729, 23rd AIAA Applied Aerodynamics Conference, Toronto, Ontario Canada, June 2005.

[148] D. Mavriplis and M. Long, "Nsu3d results for the fourth aiaa drag prediction workshop," *Journal of Aircraft*, vol. 51, no. 4, pp. 1161–1171, 2014.

[149] D. J. Mavriplis and K. Mani, "Unstructured mesh solution techniques using the nsu3d solver," aIAA Paper 2014-0081, Presented at the 52nd AIAA Aerospace Sciences Conference, National Harbor, MD, January 2014.

[150] P. Spalart and S. Allmaras, "A one-equation turbulence model for aerodynamic flows," in *30th aerospace sciences meeting and exhibit*, 1992, p. 439.

[151] D. C. Wilcox, "Reassessment of the scale-determining equation for advanced turbulence models," *AIAA journal*, vol. 26, no. 11, pp. 1299–1310, 1988.

[152] P. R. Spalart, S. Deck, M. Shur, K. Squires, M. K. Strelets, and A. Travin, "A new version of detached-eddy simulation, resistant to ambiguous grid densities," *Theoretical and computational fluid dynamics*, vol. 20, no. 3, pp. 181–195, 2006.

[153] M. L. Shur, M. K. Strelets, A. K. Travin, and P. R. Spalart, "Turbulence modeling in rotating and curved channels: assessing the spalart-shur correction," *AIAA journal*, vol. 38, no. 5, pp. 784–792, 2000.

[154] D. Mavriplis, M. Long, T. Lake, and M. Langlois, "Nsu3d results for the second aiaa high-lift prediction workshop," *Journal of Aircraft*, vol. 52, no. 4, pp. 1063–1081, 2015.

[155] M. A. Park, K. R. Laflin, M. S. Chaffin, N. Powell, and D. W. Levy, "Cfl3d, fun3d, and nsu3d contributions to the fifth drag prediction workshop," *Journal of Aircraft*, vol. 51, no. 4, pp. 1268–1283, 2014.

[156] B. Roget and J. Sitaraman, "Robust and efficient overset grid assembly for partitioned unstructured meshes," *Journal of Computational Physics*, vol. 260, pp. 1–24, 2014.

[157] Y. S. Jung, B. Govindarajan, and J. D. Baeder, "Unstructured/structured overset methods for flow solver using hamiltonian paths and strand grids," aIAA Paper 2016-1056, 54th AIAA Aerospace Sciences Meeting, San Diego, CA., June 2016.

[158] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers, "A description of the advanced research wrf version 2," DTIC Document, Tech. Rep., 2005.

[159] Y. Han, M. Stoellinger, and J. Naughton, "Large eddy simulation for atmospheric boundary layer flow over flat and complex terrains," in *Journal of Physics: Conference Series*, vol. 753, no. 3. IOP Publishing, 2016, p. 032044.

[160] R. Roy and M. K. Stoellinger, "Large eddy simulation of wind flow over complex terrain: The bolund hill case," in *35th Wind Energy Symposium*, 2017, p. 1160.

[161] Y. Han and M. K. Stoellinger, "Large eddy simulation of atmospheric boundary layer flows over complex terrain with varying stability conditions," in *35th Wind Energy Symposium*, 2017, p. 1161.

[162] B. Whitlock, J. Favre, and J. Meredith, "Parallel in situ coupling of simulation with a fully featured visualization system," eurographics Symposium on Parallel Graphics and Visualization 2011.

[163] Tech. Rep., silo Documentation Version 4.0, Lawrence Livermore National Laboratory, Livermore, CA.

[164] B. Fornberg, "Steady viscous flow past a sphere at high reynolds numbers," *Journal of Fluid Mechanics*, vol. 190, pp. 471–489, 1988.

[165] K. W. McAlister and R. Takahashi, "Naca 0015 wing pressure and trailing vortex measurements," National Aeronautics and Space Administration Moffett Field Ca Ames Research Center, Tech. Rep., 1991.

[166] A. Wissink, "An overset dual-mesh solver for computational fluid dynamics," in *7th International Conference on Computational Fluid Dynamics, Paper ICCFD7-1206, Hawaii*, 2012.

[167] J. Sitaraman and J. D. Baeder, "Evaluation of the wake prediction methodologies used in cfd based rotor airload computations," in *AIAA 24th Conference on Applied Aerodynamics, No. AIAA-2006-3472, AIAA, Washington, DC*, 2006.

[168] N. Hariharan, "Rotary-wing wake capturing: High-order schemes toward minimizing numerical vortex dissipation," *Journal of aircraft*, vol. 39, no. 5, pp. 822–829, 2002.

[169] A. Wissink, J. Sitaraman, V. Sankaran, D. Mavriplis, and T. Pulliam, "A multi-code python-based infrastructure for overset cfd with adaptive cartesian grids," *AIAA Paper 2008-927*, 2015/05/27 2008. [Online]. Available: http://dx.doi.org/10.2514/6.2008-927

[170] J. Sitaraman, M. Floros, A. Wissink, and M. Potsdam, "Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids," *Journal of Computational Physics*, vol. 229, no. 12, pp. 4703 – 4723, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002199911000118X

[171] S. J. Kamkar, A. M. Wissink, V. Sankaran, and A. Jameson, "Feature-driven cartesian adaptive mesh refinement for vortex-dominated flows," *Journal of Computational Physics*, vol. 230, no. 16, pp. 6271–6298, 7 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999111002725

[172] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, "Definition of a 5-mw reference wind turbine for offshore system development," *National Renewable Energy Laboratory, Golden, CO, Technical Report No. NREL/TP-500-38060*, 2009.

[173] M. M. Hand, D. Simms, L. Fingersh, D. Jager, J. Cotrell, S. Schreck, and S. Larwood, *Unsteady aerodynamics experiment phase VI: wind tunnel test configurations and available data campaigns*. National Renewable Energy Laboratory, Golden, Colorado, USA, 2001.

[174] D. A. Simms, S. Schreck, M. Hand, and L. Fingersh, *NREL unsteady aerodynamics experiment in the NASA-Ames wind tunnel: a comparison of predictions to measurements*. National Renewable Energy Laboratory Golden, CO, USA, 2001.

[175] S. Schreck, "The nrel full-scale wind tunnel experiment introduction to the special issue," *Wind Energy*, vol. 5, no. 2-3, pp. 77–84, 2002.

[176] L. J. Fingersh, D. Simms, M. Hand, D. Jager, J. Cotrell, M. Robinson, S. Schreck, and S. Larwood, "Wind tunnel testing of nrels unsteady aerodynamics experiment," aIAA Paper 2001-35, 20th ASME Wind Energy Symposium and the 39th Aerospace Sciences Meeting, Reno, NV, 2001.

[177] P. Moriarty, J. S. Rodrigo, P. Gancarski, M. Chuchfield, J. W. Naughton, K. S. Hansen, E. Machefaux, E. Maguire, F. Castellani, L. Terzi *et al.*, "Iea-task 31 wakebench: Towards a protocol for wind farm flow model evaluation. part 2: Wind farm wake models," in *Journal of Physics: Conference Series*, vol. 524. IOP Publishing, 2014, p. 012185.

[178] "Wind partnerships for advanced component technology: Windpact advanced wind turbine drivetrain designs," *U.S. Dept. of Energy, Energy Efficiency and Renewable Energy*, 2006.

[179] D. Malcolm and A. Hansen, "Windpact turbine rotor design study," *National Renewable Energy Laboratory, Golden, CO*, vol. 5, 2002.

[180] A. Hassanzadeh, J. W. Naughton, C. L. Kelley, and D. C. Maniaci, "Wind turbine blade design for subscale testing," in *Journal of Physics: Conference Series*, vol. 753, no. 2. IOP Publishing, 2016, p. 022048.

[181] M. Khan, Y. Odemark, and J. H. Fransson, "Effects of inflow conditions on wind turbine performance and near wake structure," *Open Journal of Fluid Dynamics*, vol. 7, pp. 105–129, 2017.

[182] D. Medici, "Experimental studies of wind turbine wakes: power optimisation and meandering," Ph.D. dissertation, KTH, 2005.

[183] J. M. Jonkman and M. L. Buhl Jr, "Fast user's guide-updated august 2005," National Renewable Energy Laboratory (NREL), Golden, CO., Tech. Rep., 2005.

[184] L. Sirovich, "Turbulence and the dynamics of coherent structures. i. coherent structures," *Quarterly of applied mathematics*, vol. 45, no. 3, pp. 561–571, 1987.

[185] H. Bergstrom, "Meteorological conditions at lillgrund," Uppsala University Document, Tech. Rep., 2009.

[186] M. Churchfield, S. Lee, P. Moriarty, L. Martinez, S. Leonardi, G. Vijayakumar, and J. Brasseur, "A Large-Eddy Simulations of Wind-Plant Aerodynamics," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. Reston, Virigina: American Institute of Aeronautics and Astronautics, January 2012. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2012-537

[187] C.-H. Moeng and P. P. Sullivan, "A comparison of shear-and buoyancy-driven planetary boundary layer flows," *Journal of the Atmospheric Sciences*, vol. 51, no. 7, pp. 999–1022, 1994.

[188] U. Schumann, "Subgrid Scale Model for Finite Difference Simulations of Turbulent Flows in Plane Channels and Annuli," *Journal of Computational Physics*, vol. 18, pp. 376–404, 1975.

[189] G. Groetzbach and U. Schumann, "Direct numerical simulation of turbulent velocity-, pressure-, and temperature-fields in channel flows," *Symposium on Turbulent Shear Flows*, pp. 14.11–14.19, april 18-20, 1977, Proceedings. Volume 1. (A77-33806 15-34) University Park, Pa., Pennsylvania State University, 1977.

[190] "Nsf nwsc-2 cheyenne," 2016, computational and Information Systems Laboratory. 2017. Cheyenne: SGI ICE XA System (Climate Simulation Laboratory). Boulder, CO: National Center for Atmospheric Research. doi:10.5065/D6RX99HX.

[191] H. C. Edwards and D. Sunderland, "Kokkos array performance-portable manycore programming model," in *Proceedings of the 2012 International Workshop on Programming Models and Applications for Multicores and Manycores*. ACM, 2012, pp. 1–10.

[192] D. S. Medina, A. St-Cyr, and T. Warburton, "Occa: A unified approach to multi-threading languages," *arXiv preprint arXiv:1403.0968*, 2014.

[193] Y. Morinishi, "Skew-symmetric form of convective terms and fully conservative finite difference schemes for variable density low-mach number flows," *Journal of Computational Physics*, vol. 229, no. 2, pp. 276–300, 2010.

[194] F. Ismail and P. L. Roe, "Affordable, entropy-consistent euler flux functions ii: Entropy production at shocks," *Journal of Computational Physics*, vol. 228, no. 15, pp. 5410–5436, 2009.

[195] P. Chandrashekar, "Kinetic energy preserving and entropy stable finite volume schemes for compressible euler and navier-stokes equations," *Communications in Computational Physics*, vol. 14, no. 5, pp. 1252–1286, 2013.

[196] A. R. Winters, R. C. Moura, G. Mengaldo, G. J. Gassner, S. Walch, J. Peiro, and S. J. Sherwin, "A comparative study on polynomial dealiasing and split form discontinuous galerkin schemes for under-resolved turbulence computations," 2017, arXiv:1711.10180 [math.NA].

[197] D. Flad and G. Gassner, "On the use of kinetic energy preserving dg-schemes for large eddy simulation," *Journal of Computational Physics*, vol. 350, pp. 782–795, 2017.

[198] E. Garnier, N. Adams, and P. Sagaut, *Large eddy simulation for compressible flows*. Springer Science & Business Media, 2009.

[199] S. Heinz and H. Gopalan, "Realizable versus non-realizable dynamic subgrid-scale stress models," *Physics of Fluids*, vol. 24, no. 11, p. 115105, 2012.

[200] F. Nicoud and F. Ducros, "Subgrid-scale stress modelling based on the square of the velocity gradient tensor," *Flow, turbulence and Combustion*, vol. 62, no. 3, pp. 183–200, 1999.

[201] Y. Luo and K. Fidkowski, "Output-based space-time mesh adaptation for unsteady aerodynamics," 2011.

[202] D. Calhoun and C. Burstedde, "Algorithmic components of forestclas adaptive mesh library: Multi-rate time stepping," 2016, http://math.boisestate.edu/~calhoun/ForestClaw/files/siam_pp_2016_two.pdf.