To the University of Wyoming:

The members of the Committee approve the dissertation of Nicholas K. Burgess presented on November 10 2011.

Dr. Dimitri J. Mavriplis, Chairperson

Dr. Luis Felipe Pereira, External Department Member

Dr. Jonathan Naughton

Dr. Jay Sitaraman

Dr. Andrew Hansen

Dr. Zhi J. Wang, External Examiner, Iowa State University

APPROVED:

Professor Paul Dellenback, Head, Department of Mechanical Engineering

Robert Ettema, Dean, College of Engineering and Applied Science

Burgess, Nicholas K., An Adaptive Discontinuous Galerkin Solver for Aerodynamic Flows , Ph.D., Department of Mechanical Engineering, November, 2011.

This work considers the accuracy, efficiency, and robustness of an unstructured high-order accurate discontinuous Galerkin (DG) solver for computational fluid dynamics (CFD). Recently, there has been a drive to reduce the discretization error of CFD simulations using high-order methods on unstructured grids. However, high-order methods are often criticized for lacking robustness and having high computational cost. The goal of this work is to investigate methods that enhance the robustness of high-order discontinuous Galerkin (DG) methods on unstructured meshes, while maintaining low computational cost and high accuracy of the numerical solutions. This work investigates robustness enhancement of high-order methods by examining effective non-linear solvers, shock capturing methods, turbulence model discretizations and adaptive refinement techniques. The goal is to develop an all encompassing solver that can simulate a large range of physical phenomena, where all aspects of the solver work together to achieve a robust, efficient and accurate solution strategy.

The components and framework for a robust high-order accurate solver that is capable of solving viscous, Reynolds Averaged Navier-Stokes (RANS) and shocked flows is presented. In particular, this work discusses robust discretizations of the turbulence model equation used to close the RANS equations, as well as stable shock capturing strategies that are applicable across a wide range of discretization orders and applicable to very strong shock waves. Furthermore, refinement techniques are considered as both efficiency and robustness enhancement strategies. Additionally, efficient non-linear solvers based on multigrid and Krylov subspace methods are presented. The accuracy, efficiency, and robustness of the solver is demonstrated using a variety of challenging aerodynamic test problems, which include turbulent high-lift and viscous hypersonic flows.

Adaptive mesh refinement was found to play a critical role in obtaining a robust and efficient high-order accurate flow solver. A goal-oriented error estimation technique has been developed to estimate the discretization error of simulation outputs. For high-order discretizations, it is shown that functional output error super-convergence can be obtained,

provided the discretization satisfies a property known as dual consistency. The dual consistency of the DG methods developed in this work is shown via mathematical analysis and numerical experimentation. Goal-oriented error estimation is also used to drive an hp-adaptive mesh refinement strategy, where a combination of mesh or h-refinement, and order or p-enrichment, is employed based on the smoothness of the solution. The results demonstrate that the combination of goal-oriented error estimation and hp-adaptation yield superior accuracy, as well as enhanced robustness and efficiency for a variety of aerodynamic flows including flows with strong shock waves.

This work demonstrates that DG discretizations can be the basis of an accurate, efficient, and robust CFD solver. Furthermore, enhancing the robustness of DG methods does not adversely impact the accuracy or efficiency of the solver for challenging and complex flow problems. In particular, when considering the computation of shocked flows, this work demonstrates that the available shock capturing techniques are sufficiently accurate and robust, particularly when used in conjunction with adaptive mesh refinement . This work also demonstrates that robust solutions of the Reynolds Averaged Navier-Stokes (RANS) and turbulence model equations can be obtained for complex and challenging aerodynamic flows. In this context, the most robust strategy was determined to be a low-order turbulence model discretization coupled to a high-order discretization of the RANS equations.

Although RANS solutions using high-order accurate discretizations of the turbulence model were obtained, the behavior of current-day RANS turbulence models discretized to high-order was found to be problematic, leading to solver robustness issues. This suggests that future work is warranted in the area of turbulence model formulation for use with high-order discretizations. Alternately, the use of Large-Eddy Simulation (LES) subgrid scale models with high-order DG methods offers the potential to leverage the high accuracy of these methods for very high fidelity turbulent simulations. This thesis has developed the algorithmic improvements that will lay the foundation for the development of a three-dimensional high-order flow solution strategy that can be used as the basis for future LES simulations.

AN ADAPTIVE DISCONTINUOUS GALERKIN SOLVER FOR AERODYNAMIC FLOWS

by

Nicholas K. Burgess,
M.S. Aerospace Engineering, Georgia Institute of Technology (2007)
B.S. Mechanical Engineering, Lehigh University (2006)

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
November 2011

To my father Fred, mother Joan, brother Matthew and dear friends Scott and Kate.

Without your support this would never have been possible.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to acknowledge the support of my colleagues, friends and family.

First and foremost I would like to acknowledge the contributions of my advisor Professor Dimitri Mavriplis. His contributions as an advisor and as a mentor cannot be overstated. He provided exceptional guidance and showed a great deal of patience during the course of this work. He has provided me with a truly unique opportunity by allowing me to do all of my work from scratch and for providing an excellent education in the area of CFD. I have gained more knowledge than I thought possible in the field of CFD. Also thank you to Dr. Jay Sitaraman who was always available for helpful discussions about aerodynamics and validation. Also I would like to acknowledge the contributions of Dr. Jonathan Naughton who helped me to understand some of the difficulties I encountered with turbulent flows.

Second, I am very grateful to the members of my doctoral committee: Dr. Dimitri J. Mavriplis, Dr. Luis F.Pereira, Dr. Jay Sitaraman, Dr. Jonathan Naughton and Dr. Andrew C. Hansen who put time and effort into reading and revising the manuscript of this dissertation. Their comments and criticisms have helped to clarify the work presented within. Also a special thank you is given to the external examiner, Dr. Zhi J. Wang of Iowa State university for his review of the manuscript.

I would also like to acknowledge the contributions of two of my colleges in the CFD lab: Brian Lockwood and Karthik Mani. They were always available to bounce ideas around with and offer their opinion on this work as it progressed. Their comments and criticisms helped improve this work more than I can acknowledge. Furthermore, thank you to Dr. Andrew Shelton of Auburn University for the many helpful and insightful conversations and for his advice during the course of this work. Also thank you to Dr. Markus Rumpfkeil for his many

helpful and insightful conversations regarding turbulence modeling and for providing some supplementary data. Finally, I would like to acknowledge my former research advisors Dr. Terry Delph (Lehigh University) and Dr. Marilyn Smith (Georgia Tech) for their support of my decision to come to the University of Wyoming and for their advice during my time under their mentorship. This work was made possible by the following funding agencies, NASA under grant number NNX07AC31A, AFOSR under a Phase 2 STTR project: contract F9550-09-C-0051 and NSF under grant 0904936.

On a personal note, I am grateful to my family and friends for all their support during this process. It has been a long and difficult road and without their support I would not have been able to get to the end of this road. In particular, I would like to acknowledge the support of my family: Fred, Joan, Matthew as well as two of my oldest and closest friends Scott and Kate, all of whom have supported me throughout my academic career. This work is dedicated to them, for everything they have done to keep me going.

Nicholas K. Burgess

University of Wyoming

November 2011

Chapter 1

Introduction

Computational fluid dynamics (CFD) is now a standard tool for conducting flow field analysis for a variety of applications spanning engineering and science. Current state-of-the-art techniques in CFD are nominally second-order accurate flow solvers that are usually based on finite-volume [1–6] or finite-difference methods [2, 3, 7–10]. However, solvers based on second-order continuous finite-element methods [11, 12] have also been developed for CFD applications. Due to the second-order accuracy of these CFD solvers, computing low error simulations requires very large meshes that take a great deal of time to generate, process, and partition for parallel processing. As CFD matures and computational resources grow, the complexity and scope of the problems being solved grows in tandem. However, as the complexity of the problems increases, the resolution requirements of the computational simulations also increase. For example, it is now commonplace to compute entire helicopter fuselages and rotor blades in the same simulation, which requires very high resolution to obtain adequate results for non-trivial flight conditions. Due to the high resolution requirements of such problems, interest in the use of high-order discretizations (higher than second-order) for industrial computational fluid dynamic problems, including aerodynamics and aerothermodynamics, has become more widespread over the last several years. This is partly due to the difficulties encountered with traditional second-order accurate methods at delivering consistently grid converged results and in quantifying the spatial discretization errors [13, 14]. Furthermore, complex multi-scale problems such as the Large Eddy Simula-

tion (LES) of turbulent flows require very high resolution to obtain accurate results. The asymptotic error properties of high-order methods makes them suitable for problems where high spatial accuracy is required, since for smooth solutions, spatial error is reduced ever more rapidly with increasing grid resolution at higher "p" orders of accuracy. However, there is still a great amount of research that must be conducted before high-order methods are suitable for industrial scale problems. In particular, the robustness of high-order methods is considered specifically in this work. Robustness is an area that has received limited attention in the literature and is critical to industrial applications.

Complex aerodynamic flow fields exhibit a wide range of phenomena including thin boundary layers, high streamline curvature regions, shock waves and turbulence model artifacts. The resolution of the latter two types of phenomena represents a significant challenge for high-order methods. Direct application of high-order methods results in Gibbs phenomena that may cause solver failure [15], which is a result of using high-order polynomials to approximate non-smooth solution behavior. The standard treatments of Gibbs phenomena that are employed in the low-order methods context are not suitable for high-order discretizations. As a result, much of this work focuses on finding optimal strategies for dealing with these challenges robustly and efficiently, while maintaining high-order accuracy as often as possible. In order to address these challenges, a discontinuous Galerkin(DG) method is employed as the basis for the high-order unstructured CFD solver in this work. DG discretization methods are capable of generating arbitrarily high-order accurate results on unstructured grids made of triangles and/or quadrilaterals, in two spatial dimensions. In addition, DG discretizations have a rich mathematical foundation with excellent stability theory, conservation properties, and consistency.

An efficient and robust high-order accurate unstructured CFD solver for aerodynamic applications based on a DG discretizations, requires the consideration of many facets of CFD. In particular, the discretization must be able to ensure that high-order accuracy is obtained. The stabilization methods employed must be robust and suitable for DG discretizations. Furthermore, the non-linear solver and the corresponding linear solver must ensure adequate and rapid convergence of the discrete equations. Finally, refinement must be conducted

carefully so that degrees of freedom (DoFs) are placed efficiently and robustly. This work focuses on each of these areas as well as the coupling between them to develop a high-order accurate DG CFD solver on mixed-element unstructured meshes. The dual consistency of the discretization is derived by analysis and verified through numerical experiments. Preconditioned Krylov subspace methods are developed for the efficient solution of the discrete equations. Additionally, adaptive methods are also investigated for both efficiency and robustness enhancement. The adaptation is driven via a goal-oriented adjoint-based error estimation technique similar to those used in references [16–20] such that the adaptation strategy targets the error in a specific simulation output (objective) as efficiently as possible.

1.1 High-order Discontinuous Galerkin Methods

High-order methods have been successfully applied to linear partial differential equation(PDE) based problems [21–24]. However, this does not preclude the application of high-order methods to non-linear equations e.g. such as those associated with CFD [25–32]. High-order methods represent an excellent strategy for removing the discretization error from CFD simulation results because they can deliver asymptotic solution error convergence rates of $\mathcal{O}\left(h^{p+1}\right)$, where h is the average element size and p is the discretization order. Furthermore, under a specific condition known as dual consistency, simulation output functional error will convergence at the rate of $\mathcal{O}\left(h^{2p}\right)$. Therefore, for increasingly high accuracy tolerances, the use of high-order methods such as DG becomes more appealing because simulation error is reduced evermore rapidly as the discretization order is increased. However, as the discretization order is increased the number of degrees of freedom (DoF) rises rapidly. For example, second-order accurate ($p = 1$) DG discretizations for the two-dimensional Euler or Navier-Stokes equations have 12 DoFs per element while a fourth-order accurate ($p = 3$) element has 40 DoFs per element. Thus high-order methods have a significant computational cost associated with them, and this cost scales with the discretization order of the element.

DG methods are, in their most basic form, a finite-element method. However, typical finite-element methods are continuous finite-element methods where the basis functions,

which approximate the discrete solution, are continuous at the element interfaces. Continuous finite-element methods traditionally have been applied to linear structural and thermal analysis problems that constitute purely elliptic operators, and hence continuous basis functions are appropriate. DG methods employ basis functions that are discontinuous at the element interfaces, which makes DG methods naturally suitable for computing convection dominated problems. DG discretizations are an ideal choice for convection dominated problems because the discontinuous basis functions allow for upwind flux calculations using approximate Riemann solvers. Employing approximate Riemann solvers at the element interfaces is a strategy that is borrowed from finite-volume methods. Thus DG can be thought of as a combination of traditional finite-element and finite-volume methods. The blending of these methods is the result of simultaneously viewing the element as a control volume and as a domain over which interpolation functions (which are also known as basis functions) may be defined. However, since the DG method is a finite-element method, the order of accuracy and number of unknowns are coupled. DG methods attain high-order accuracy by adding additional basis functions within the elements, which results in additional degrees of freedom for increased orders of accuracy. Alternatively, finite-volume and finite-difference methods reconstruct high-order data from neighboring elements, which does not increase the total number of degrees of freedom. Therefore, finite-volume and finite-difference methods do not couple the order of accuracy with the number of degrees of freedom. The coupling of the order of accuracy and number of unknowns within an element is a non-trivial property of DG methods, which affects many aspects of solver robustness and hence is a recurring theme throughout this work. However, locating extra unknowns within the elements can be advantageous, provided that great care is taken in constructing and implementing these methods.

As problem size increases, the efficient use of parallel computers becomes more important. DG methods add resolution to a given problem via two approaches. DG methods can add resolution by increasing the number of degrees of freedom within the element, which results in increased parallel efficiency over low-order methods for unstructured grids [33]. By locating the DoFs within the element, higher computational density is achieved and pro-

portionally less inter-element data communication is required. This makes high-order DG methods an ideal candidate for large scale parallel computing. Contrarily, while high-order finite-difference methods have been developed, these methods require the construction of extended interpolation stencils. Extending the interpolation stencil can cause parallel scaling to degrade as the order of accuracy is increased. This degradation of parallel efficiency is a result of the stencils of the grid points on partition boundaries relying on information from multiple data points on neighboring processors. Reference [33] has shown that high-order DG methods have the opposite trend, as the order of accuracy increases the parallel scalability increases as well.

1.1.1 Shock Waves and High-order Methods

High-order methods rely on the solution being sufficiently smooth to attain high-order convergence rates and maintain non-oscillatory solutions. However, if the solution is not sufficiently smooth high-order methods often fail due to Gibbs phenomena. Gibbs phenomena are manifested as oscillations due to the use of high-order interpolation for non-smooth or discontinuous solutions. These oscillations can cause negative pressure and density values, which are non-physical states for the equations of fluid motion and hence result in solver failure. While there has been extensive work conducted on stabilization methods [15, 28, 34–38] additional work is still warranted to determine the most effective way to robustly compute flows with discontinuities. Despite the availability of several shock capturing schemes, there is still a great deal of debate [39] on the most effective way to compute flows containing shock waves. In order to discuss resolving shock waves with compact high-order methods, the coupling between order of accuracy and number of degrees of freedom must be addressed.

In this work, stabilization methods for shocked flows are discussed. In particular, stabilization methods are evaluated based on their ability to account for the coupling of order of accuracy and the number of degrees of freedom. In CFD, stabilization methods take two main forms, slope/flux limiters and artificial diffusion methods. Slope and flux limiters are commonly used in finite-volume and finite-difference methods, and reference [1] gives a concise review of these techniques. However, artificial diffusion methods have also been investigated

for finite-volume and finite-difference methods. VonNeumann and Ritchmeyer were the first to consider this idea in reference [40]. Reference [4] also considered artificial diffusion shock capturing methods in a structured two-dimensional finite-volume setting. Recently, there has been a resurgence of interest in artificial diffusion methods for shock capturing within a compact high-order discretization setting [28, 34, 36, 37]. Artificial diffusion is an attractive method for application to DG discretizations because it can take the coupling between order of accuracy and number of degrees of freedom into account. The governing parameter of an artificial diffusion method is the artificial viscosity. This work discusses limiters and artificial viscosity and makes some comparisons between two of the most successful artificial viscosity techniques applied to high-order methods. While stabilization methods are an important subject pertaining to the computation of shocked flows, the choice of refinement method is equally important. Therefore, refinement methods for shocked flows are also discussed in detail in this work. The relationship between refinement method and robustness for the resolution of shock waves is of particular importance for DG discretizations because of the coupling between the order of accuracy and the number of degrees of freedom.

1.1.2 Turbulence Modeling

Turbulent flows modeled using the Reynolds Average Navier-Stokes (RANS) equations represent another challenging application area for high-order DG discretizations. Turbulent modeled using the Reynolds Averaged Navier-Stokes (RANS) equations employ a closure model for the turbulent eddy viscosity. This work employs the turbulence model of Spalart and Allmaras (SA) [41] as the closure model for the RANS system. The challenges in computing RANS flows stem from a discontinuity in the SA turbulence model working variable. This discontinuity takes the form of artificial sharp interfaces that occur at the edges of boundary layers and wakes and has proven extremely difficult to eliminate. The presence of this discontinuity has gone largely ignored in the low-order methods and turbulence modeling literature. For example, reference [41] does not mention this discontinuity in the turbulence model working variable. This discontinuity can lead to negative values of the turbulence model working variable which impact the stability of the SA turbulence model discretiza-

tion. References [20, 42, 43] have also computed RANS solutions using a DG method but have cited significant difficulty in doing so. Herein, an attempt is made to explain why the discontinuity is so pronounced in the high-order setting. While obtaining high-order solutions to the turbulence model is sometimes possible, obtaining these solutions is not robust enough for general applications. Reference [20] presented modifications to the SA turbulence model source terms that are intended to stabilize the model for negative values of the turbulence model working variable. These modifications are implemented within the presented DG solver in order to help alleviate the difficulties encountered when the turbulence model working variable becomes negative. Additionally, the behavior and effectiveness of these modifications are analyzed in detail. The discretization and solution method of the SA turbulence model equation are discussed extensively. In particular, the choice of convective numerical flux function as well as implicit solver treatments are important aspects of the discretization and solution methodology of the SA turbulence model equation.

The lineage of the work on turbulent RANS flows can be traced directly to reference [20]. However, there are significant differences between reference [20] and the work on RANS flows presented in this dissertation. In particular reference [20] does not consider the robustness of high-order DG discretizations of the RANS equations. Furthermore, this work expands on reference [20] by considering the grid convergence of lift and drag rather than attached viscous drag alone. This work also considers the solution of turbulent high-lift flows on mixed-element meshes and is the first to demonstrate a robust strategy for these flows, which are particularly challenging to solve.

1.2 Error Estimation and Adaptation

Functional or output error estimation has been the subject of intense research over the last decade [16–18, 43–48]. Error estimation can be used to drive adaptive refinement procedures that target the error in a particular output of interest. For aerodynamic flows, the output is often an aerodynamic loading such as lift or drag. Output-based error estimation often results in adaptive mesh refinements that are non-intuitive, which is the reason one appeals

to these methods to guide adaptive refinement. In order to estimate the error in a functional, one must solve the discrete adjoint problem to obtain the sensitivity of the functional with respect to the solution residual. In this work, the discrete adjoint is obtained from the discretization of the physical or primal problem. Functional error super-convergence and accurate error estimates are only obtained if the discrete adjoint problem is consistent with the corresponding continuous adjoint problem. The consistency is defined such that the discrete adjoint derived from the primal discretization represents a discretization of the continuous adjoint equation. This property is known as dual consistency [49].

Dual consistency is a particularly important property for high-order discretizations, because if a discretization is dual consistent one can show that functional error behaves as $O(h^{2p})$, where h is the mesh size and p is the discretization order. This functional error convergence behavior is known as super-convergence. In this work, dual consistency is analyzed for some model problems to demonstrate that the strategies adopted are indeed dual consistent. In particular, the dual consistency of the stabilization methods for shock waves is discussed in Chapter 3. When applying stabilization methods for shock waves one must be very careful not to introduce terms that may be dual inconsistent. If this dual consistency is lost, then output error estimates will be affected and one cannot use them as reliable adaptation, correction or simulation termination criteria. Additionally, the treatment of boundary conditions affects the dual consistency of a discretization, and hence this work examines the dual consistency of an example boundary condition. The analysis demonstrates the origin of the mathematical form of the boundary conditions used in this work. Additionally, the functional error super-convergence bound (i.e. $O(h^{2p})$) is also derived.

A large part of this work focuses on the computation of shock waves and other discontinuous solutions. Therefore, the subject of error estimation for discontinuous solutions is discussed. Solution irregularities are oscillations in the solution such as Gibbs phenomena that result from projecting the solution from a coarse to a fine mesh. Usually the irregularity is the result of projecting a solution that is discontinuous from a discretization order p to a discretization order $p+1$. If the solution contains irregularities, then these irregularities will likely corrupt the functional error estimate. Corrupt error estimates often manifest them-

selves as error estimates that do not converge under conditions when the actual functional error is obviously converging. This work shows how to obtain accurate error estimates for shocked flows even in the presence of potential irregularities. The effect of turbulence model discontinuity on functional error estimates will also be considered.

In this work, functional error estimates are used to drive mesh adaptation. In particular *hp*-adaptation [17, 47] is used throughout this work. Recall that DG methods can increase resolution via two approaches: mesh-refinement which decreases the size h of the elements or p-enrichment which increases the discretization order p within the elements. *hp*-adaptation is an adaptation strategy where both forms of resolution enhancement are conducted simultaneously and was first presented in reference [17]. While the *hp*-adaptation strategy in this work is based on reference [17], this work has made several improvements to enhance the robustness and flexibility of this adaptation strategy. In particular this work considers non-conforming *hp*-adaptation on mixed-element meshes as well as viscous flows. Furthermore, this work considers combining *hp*-adaptation with artificial diffusion for shock capturing, which has not been considered by previous work. While *hp*-adaptation certainly shows significant efficiency improvements compared to low-order mesh adaptation and uniform refinement [17, 47], *hp*-adaptation can also be used as a robustness enhancement method. By resolving non-smooth features using low-order accurate discretizations, the solver becomes significantly more robust and still maintains high-order accuracy of functional outputs, as will be shown. Slope limiters examine the smoothness of a cell to decide whether to reduce the discretization order locally. Additionally, *hp*-adaptation can be viewed as slope limitation applied in the reverse direction, and is similar in that it examines the smoothness of an element to decide whether or not to increase the discretization order locally. Thus these two procedures are similar but operate in reverse directions. The benefits of applying *hp*-adaptation to DG discretizations will be discussed and demonstrated.

1.3 Dissertation Overview

The main theme of this work is robust, efficient, and accurate high-order discretization and solution strategies that can be applied to real world fluid dynamics problems. Much of the work focuses on shocked and turbulent flows, although simpler laminar and inviscid flows are considered as well. The applications range from simple inviscid flows to viscous supersonic and viscous hypersonic flows, as well as turbulent flows over complex geometries, all in two spatial dimensions. The main contributions of the dissertation are:

- Chapter 3 considers the dual consistency analysis of artificial diffusion methods for shock capturing.
- Chapter 4 develops efficient h and p-independent solvers for viscous flows on high aspect ratio mixed-element meshes.
- Chapter 5 considers an investigation of hp-adaptation for robustness and development of this adaptation strategy for mixed element meshes.
- Investigation of the effectiveness of combining hp-adaptation and artificial diffusion for computing shocked flows is considered in Chapters 5 and 7.
- Chapter 6 considers the development of robust Spalart Allmaras turbulence model discretizations for the RANS equations.
- Chapter 7 considers an investigation of the most robust and effective combination of refinement and shock capturing methods for hypersonic flows.
- Chapter 8 compares high-order unstructured DG methods to current state-of-the-art second-order accurate unstructured finite-volume methods.

The overall goal of this work is to examine the robustness of high-order DG methods and improve the robustness where necessary. In order to demonstrate the robustness and efficiency of the presented DG solver, several test problems that span a range of complexity and difficulty from simple steady-state laminar viscous flows to hypersonic and turbulent

flows are examined. The goal of the dissertation is to demonstrate that, with the described strategy, high-order solutions to practical problems can be obtained. Furthermore, a wide range of problems is considered in order to demonstrate the robustness of the solver. The final chapter discusses quantitative comparisons between the DG solver and a finite-volume solver also written by the author. The subjects of computational expense, robustness and shock capturing are compared between the two solvers in a quantitative fashion. A graphical outline of the dissertation is shown in Figure 1.1, which demonstrates how various subjects in the dissertation are related to one another.

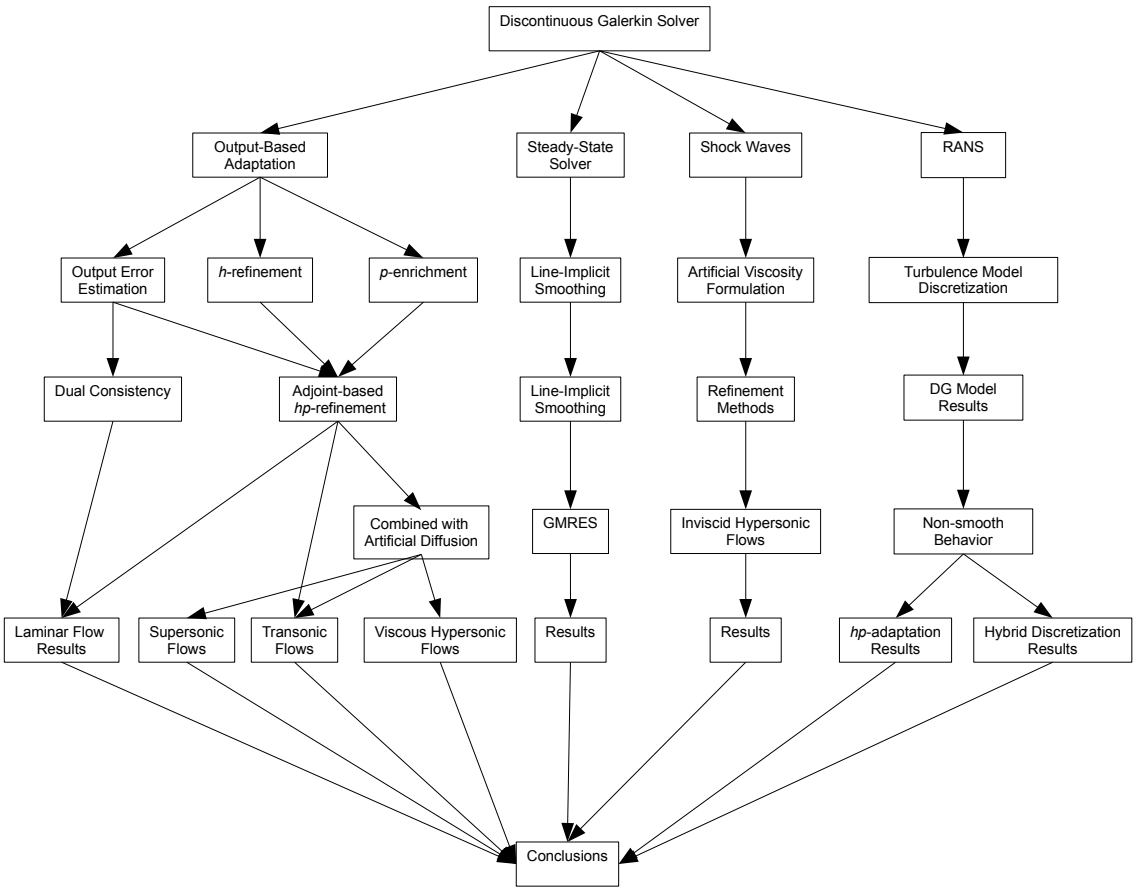


Figure 1.1: Diagram of how various portions of the dissertation are related.

Chapter 2

Discontinuous Galerkin Methods

Discontinuous Galerkin(DG) methods can be regarded as a combination of finite-volume and finite-element methods. DG methods are capable of obtaining arbitrarily high-order accuracy by expanding the solution as a set of basis functions and coefficients within each element. This is similar to traditional finite-element methods with the exception that the basis functions are allowed to be discontinuous at the element interfaces. Allowing discontinuous basis functions establishes upwinding inter-element communication similar to finite-volume methods. This property of DG discretizations allows for the natural computation of hyperbolic operators but complicates the treatment of diffusion operators. In this work, both the Euler and Navier-Stokes equations are considered, which requires stable and accurate convection and diffusion discretizations. This section discusses the governing equations and discontinuous Galerkin discretization of both convection and diffusion operators as well as shock capturing via artificial diffusion. A detailed derivation of the discretization of diffusion operators is provided in Appendix A.

In addition to the discretization, the boundary conditions used throughout this work are derived and discussed in detail. In particular the derivation of the slip wall, no-slip wall, and far-field boundary condition is considered. Boundary conditions play a critical role in obtaining optimal functional error convergence and are not derived in the typical "ghost" cell fashion, which is common in many unstructured finite-volume methods. The "ghost" cell method is avoided due to the lack of dual consistency of this approach (Chapter 3).

2.1 Governing Equations

The conservative form of the compressible Reynolds Averaged Navier-Stokes (RANS) equations describing the conservation of mass, momentum and total energy in two dimensions is given as:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \left(\vec{\mathbf{F}}_c(\mathbf{u}) - \vec{\mathbf{F}}_v(\mathbf{u}, \nabla \mathbf{u}) \right) = \mathbf{S}\left(\mathbf{u}, \nabla \mathbf{u}\right) \tag{2.1.1}$$

within a domain Ω, subject to the appropriate boundary conditions on the domain boundary Γ and a suitable initial condition at $t = 0$. In equation (2.1.1), $\mathbf{u}$ is the vector of conserved variables, $\vec{\mathbf{F}}_c$ is the convective flux, $\vec{\mathbf{F}}_v$ is the viscous flux and $\mathbf{S}$ is the source term. In this work, the RANS equations are coupled to the one equation turbulence model of Spalart and Allmaras (SA model) [41] with the modifications given in reference [20]. The equation for this turbulence model is given by:

$$\frac{\partial \rho \tilde{\nu}}{\partial t} + \nabla \cdot (\rho \tilde{\nu} \vec{u}) = \mathcal{P}\left(\mathbf{u}, \nabla \mathbf{u}\right) + \frac{1}{\sigma}\left[\nabla \cdot (\eta \nabla \tilde{\nu}) + c_{b_2} \rho \nabla \tilde{\nu} \cdot \nabla \tilde{\nu}\right] - \mathcal{D}\left(\mathbf{u}, \nabla \mathbf{u}\right) \tag{2.1.2}$$

where $\tilde{\nu}$ is the turbulence model working variable, ρ is the density, $\vec{u}$ is the velocity field, and σ, c_{b_2} are constants. In equation (2.1.2), $\nabla \cdot (\rho \tilde{\nu} \vec{u})$ is turbulence model convection term, the term multiplied by $\frac{1}{\sigma}$ is the diffusion term, η is the diffusion coefficient, $\mathcal{P}\left(\mathbf{u}, \nabla \mathbf{u}\right)$ is the production source term, and $\mathcal{D}\left(\mathbf{u}, \nabla \mathbf{u}\right)$ is the destruction source term. Appendix B gives a full description of the model terms and analyzes the modifications to the production and destruction terms given in reference [20].

The state vector and flux vectors including those of the SA model equation for two

dimensional flow are explicitly given as:

$$\mathbf{u} = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ E_t \\ \rho\tilde{\nu} \end{array} \right\}, \quad \mathbf{F_c}^x = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + P \\ \rho uv \\ u(E_t + P) \\ \rho u\tilde{\nu} \end{array} \right\}, \quad \mathbf{F_c}^y = \left\{ \begin{array}{c} \rho v \\ \rho uv \\ \rho v^2 + P \\ v(E_t + P) \\ \rho v\tilde{\nu} \end{array} \right\},$$

$$\mathbf{F_v}^x = \left\{ \begin{array}{c} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + c_p \left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T} \right) \frac{\partial T}{\partial x} \\ \frac{1}{\sigma} (\eta) \frac{\partial \tilde{\nu}}{\partial x} \end{array} \right\},$$

$$\mathbf{F_v}^y = \left\{ \begin{array}{c} 0 \\ \tau_{yx} \\ \tau_{yy} \\ u\tau_{yx} + v\tau_{yy} + c_p \left(\frac{\mu}{P_r} + \frac{\mu_T}{P_{r_T}} \right) \frac{\partial T}{\partial y} \\ \frac{1}{\sigma} (\eta) \frac{\partial \tilde{\nu}}{\partial y} \end{array} \right\}, \tag{2.1.3}$$

$$\mathbf{S} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ \mathcal{P}(\mathbf{u}, \nabla\mathbf{u}) + \frac{1}{\sigma} [c_{b_2} \rho \nabla\tilde{\nu} \cdot \nabla\tilde{\nu}] - \mathcal{D}(\mathbf{u}, \nabla\mathbf{u}) \end{array} \right\}$$

where ρ is fluid density, ($\vec{u} = (u, v)$) are the Cartesian velocity components, P is the fluid pressure, E_t is the total energy, c_p is the specific heat at constant pressure, T is the fluid temperature, P_r and P_{r_T} are the Prandtl and turbulent Prandtl numbers respectively and τ_{ij} is the total viscous stress tensor including the Boussinesq approximated Reynolds stresses. Assuming a Newtonian fluid and using the Boussinesq approximation for the Reynolds stresses,

the viscous stress tensor takes the form (with $x_i = x, y; \quad i = 1, 2$):

$$\begin{aligned} &\tau_{ij} = 2\left(\mu + \mu_T\right) S_{ij} \\ &S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \\ &\text{for } i = 1, 2, j = 1, 2 \end{aligned} \tag{2.1.4}$$

where μ is the fluid viscosity obtained via Sutherland's law and μ_T is a turbulent eddy viscosity, which is given by:

$$\begin{aligned} \mu_T &= \begin{cases} \rho\tilde{\nu} f_{v_1} & \tilde{\nu} \geq 0 \\ 0 & \tilde{\nu} < 0 \end{cases} \\ f_{v_1} &= \frac{\left(\frac{\rho\tilde{\nu}}{\mu}\right)^3}{\left(\frac{\rho\tilde{\nu}}{\mu}\right)^3 + c_{v_1}^3} \\ c_{v_1} &= 7.1 \end{aligned} \tag{2.1.5}$$

The components of the viscous stress tensor for two dimensional flow are given explicitly as:

$$\begin{aligned} \tau_{xx} &= \left(\mu + \mu_T\right)\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right), \quad \tau_{xy} = \left(\mu + \mu_T\right)\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \tau_{yx} &= \left(\mu + \mu_T\right)\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \quad \tau_{yy} = \left(\mu + \mu_T\right)\left(\frac{4}{3}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{\partial u}{\partial x}\right) \end{aligned} \tag{2.1.6}$$

It should be understood that all quantities in the above equations are the Reynolds Averaged quantities (the usual $\bar{()}$ notation is omitted for simplicity). The pressure is obtained from the ideal gas equation of state given as:

$$P = \left(\gamma - 1\right)\left[E_t - \frac{1}{2}\rho\left(u^2 + v^2\right)\right] \tag{2.1.7}$$

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

The RANS equations are subject to a non-reflecting far-field boundary condition and a wall boundary condition. The wall boundary condition is a no-slip wall, hence the velocity is zero at the wall. Additionally, the eddy viscosity is zero at the wall because the Reynolds stresses are zero at the wall.

$$\begin{aligned} \vec{u} &= (0, 0) \quad \vec{x} \in \Gamma_{wall} \\ \rho\tilde{\nu} &= 0 \quad \vec{x} \in \Gamma_{wall} \end{aligned} \tag{2.1.8}$$

Laminar flow solutions are obtained by simply setting $\mu_T = 0$ and eliminating the turbulence model from the system of equations.

When shock waves are present, artificial diffusion fluxes are added to the governing equations in order to stabilize the solution in the vicinity shock waves. In this case the governing equations take the following form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \left(\vec{\mathbf{F}}_c(\mathbf{u}) - \vec{\mathbf{F}}_v(\mathbf{u}, \nabla \mathbf{u}) - \vec{\mathbf{F}}_{ad}\left(\hat{\epsilon}, \mathbf{u}, \nabla \mathbf{u}\right) \right) = 0 \tag{2.1.9}$$

where the convective ($\mathbf{F}_c$) and viscous ($\mathbf{F}_v$) fluxes are the same as equation (2.1.3) and the artificial diffusion fluxes are given as:

$$\mathbf{F}_{\mathbf{ad}x} = \begin{Bmatrix} \hat{\epsilon}\frac{h_x}{\bar{h}}\frac{\partial \rho}{\partial x} \\ \hat{\epsilon}\frac{h_x}{\bar{h}}\frac{\partial \rho u}{\partial x} \\ \hat{\epsilon}\frac{h_x}{\bar{h}}\frac{\partial \rho v}{\partial x} \\ \hat{\epsilon}\frac{h_x}{\bar{h}}\frac{\partial \rho H}{\partial x} \end{Bmatrix}, \quad \mathbf{F}_{\mathbf{ad}y} = \begin{Bmatrix} \hat{\epsilon}\frac{h_y}{\bar{h}}\frac{\partial \rho}{\partial y} \\ \hat{\epsilon}\frac{h_y}{\bar{h}}\frac{\partial \rho u}{\partial y} \\ \hat{\epsilon}\frac{h_y}{\bar{h}}\frac{\partial \rho v}{\partial y} \\ \hat{\epsilon}\frac{h_y}{\bar{h}}\frac{\partial \rho H}{\partial y} \end{Bmatrix} \tag{2.1.10}$$

where $\hat{\epsilon}$ is the artificial viscosity and H is the total enthalpy, which is given as:

$$H = E_t + \frac{P}{\rho} \tag{2.1.11}$$

The terms, h_x, h_y and $\bar{h}$ are mesh size metrics, which are discussed in Section 2.7.3. For an infinitely fine mesh, which is obtained by taking the he mesh size h taken to zero, the artificial diffusion fluxes vanish. Therefore the artificial diffusion operator is consistent with the governing partial differential equations (PDEs).

2.1.1 Simplification to the Euler Equations

This work also considers inviscid shocked flows that are governed by the compressible Euler Equations of gas dynamics. For an inviscid flow both the molecular viscosity μ and the turbulent eddy viscosity μ_T are set to zero and thus no viscous stresses exist. Additionally, since the viscosities are zero there is no heat conduction, as seen in equation (2.1.3). This gives rise to the following governing equations

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \left(\vec{\mathbf{F}}_c(\mathbf{u}) - \vec{\mathbf{F}}_{ad}\left(\hat{\epsilon}, \mathbf{u}, \nabla \mathbf{u}\right) \right) = 0 \tag{2.1.12}$$

where the artificial diffusion fluxes are the same as in equation (2.1.10) and the convective flux is the same as in equation (2.1.3). The appropriate boundary condition for the Euler equations is a zero normal flow boundary condition.

$$\vec{u} \cdot \vec{n} = 0 \quad \vec{x} \in \Gamma_{wall} \tag{2.1.13}$$

The far-field boundary condition is the same non-reflective boundary condition that is applied to the Navier-Stokes equations.

2.2 Discontinuous Galerkin Discretizations

To discretize the governing equations, a mesh is defined consisting of elements such that the union of these elements makes up the domain on which the PDEs are to be solved. Within each element, a finite dimensional function space consisting of a finite set of functions of order p is defined. DG discretizations are carried out by first taking the inner product of the governing equations with each function on each element. These weighting functions are known as test functions, which in this work take the form of polynomials. The solution $\mathbf{u}$ is then discretized into a polynomial representation $\mathbf{u}_h$ that takes the form of known polynomials and unknown coefficients or modes. The solution expansion polynomials, also known as basis functions, are the same as the test functions and are defined in a standard element that must be mapped to the physical element. Taking the solution basis functions to be the same as the test functions is the key property that defines a Galerkin method.

The final step of the discretization is to integrate the governing equations by parts once to yield the weak form discretization. The integrals are evaluated using numerical quadrature formulas in the standard element. The integrals are transfered back to the physical space using a mapping from the standard element to the physical element. In order to account for geometry curvature, super-parameter mappings are used, i.e the mappings of boundary elements are generated to $p+1$ order where p is the solution order. For each boundary element the mapping is generated by interpolating additional surface points on the boundary onto a set of mapping basis functions for each element on the boundary. The additional surface geometry points are taken from original analytic definition of the configuration geometry. In

general, the mapping basis functions are defined differently from the solution basis functions used to define the approximate solution $\mathbf{u}_h$.

Before the discretization is described in detail it is important to explain the notation used in deriving the discretization. Firstly, a **bold face** symbol denotes a vector with size equation to the number of fields (i.e. the number of partial differential equations denoted N_f) while $\vec{(\cdot)}$ denotes a vector in d spatial dimensions (herein $d = 2$). A matrix will be denoted by $[\cdot]$. With regard to the discretization, given an exact solution $\mathbf{u}$, the corresponding discrete solution is denoted $\mathbf{u}_h$. Likewise for a continuous test function $\mathbf{v}$ the discrete test function is denoted by $\mathbf{v}_h$. For the derivation of the discretization, the test functions are written as vectors with the same dimension as the solution. The inner products result in scalar discrete equations from the continuous system, the test function space has enough functions to obtain the required number of discrete equations (i.e. number of PDEs N_f times number of modes M on the element).

The DG discretization is carried out by considering $\mathbf{u} \in \mathcal{V}$, where $\mathcal{V}$ is the space which contains the exact solution. Let the computational domain Ω be partitioned into a set of non-overlapping elements such that

$$\Omega = \bigcup_{k=1}^{N} \Omega_k \quad k \in \mathcal{T}_{h,p} \tag{2.2.1}$$

where $\mathcal{T}_{h,p} = \{k\}$ is the set of all elements k in the mesh of size h and discretization order p. Subsequently the subscript p in the notation will be omitted for brevity and simply let $\mathcal{T}_h$ represent the discretized domain. Let k denote an element $k \in \mathcal{T}_h$ on which a discrete function space $\mathcal{V}_h^p$ is defined, which is chosen such that $\mathcal{V}_h^p \subset \mathcal{V}$. Additionally, let the collection of interior faces of $\mathcal{T}_h$ be denoted $\mathcal{I}_h = \{i\}$, where i denotes a face $\in \mathcal{I}_h$. Also let the boundary $\partial\Omega$ be discretized into a set of non-overlapping faces $\mathcal{B}_h = \{b\}$ with a single boundary face denoted $b \in \mathcal{B}_h$. Let $\mathbf{u}_h$ represent the discrete solution to $\mathbf{u}$, then the DG discretization is derived by substituting $\mathbf{u}_h$ for $\mathbf{u}$ in equation (2.1.9), multiplying with the test function $\mathbf{v}_h^T$, and integrating over the domain Ω, which has been partitioned in elements as defined by equation (2.2.1). Formally, the discretization is given by finding $\mathbf{u}_h \in \mathcal{V}_h^p$ such

that

$$\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\frac{\partial\mathbf{u}_h}{\partial t}+\mathbf{v}_h^T\nabla\cdot\left(\vec{\mathbf{F}}_c\left(\mathbf{u}_h\right)-\vec{\mathbf{F}}_v\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)-\vec{\mathbf{F}}_{ad}\left(\epsilon,\mathbf{u}_h,\nabla\mathbf{u}_h\right)\right)$$
$$-\mathbf{v_h}^T\mathbf{S}\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)d\Omega_k=0,\quad\forall\mathbf{v}_h\in\mathcal{V}_h^p \tag{2.2.2}$$

This can be written as

$$\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\frac{\partial\mathbf{u}_h}{\partial t}d\Omega_k+\mathbf{R}_h\left(\mathbf{u}_h,\nabla\mathbf{u}_h,\mathbf{v}_h\right)=0,\quad\forall\mathbf{v}_h\in\mathcal{V}_h^p \tag{2.2.3}$$

where the $\mathbf{R}_h\left(\mathbf{u}_h,\nabla\mathbf{u}_h,\mathbf{v}_h\right)$ is the discrete spatial residual. The spatial residual is integrated by parts resulting in the following weak form

$$\begin{aligned}\mathbf{R}_h\left(\mathbf{u}_h,\nabla\mathbf{u}_h,\mathbf{v}_h\right)=&-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\mathbf{v}_h^T\cdot\left(\vec{\mathbf{F}}_c\left(\mathbf{u}_h\right)-\vec{\mathbf{F}}_v\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)-\vec{\mathbf{F}}_{ad}\left(\epsilon,\mathbf{u}_h,\nabla\mathbf{u}_h\right)\right)\\&+\mathbf{v_h}^T\mathbf{S}\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)d\Omega_k+\\&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\mathcal{H}_c\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\vec{n}\right)-\mathcal{H}_v\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\nabla\mathbf{u}_h^+,\nabla\mathbf{u}_h^-,\vec{n}\right)-\\&\mathcal{H}_{ad}\left(\epsilon^+,\epsilon^-,\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\nabla\mathbf{u}_h^+,\nabla\mathbf{u}_h^-,\vec{n}\right)ds+\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathcal{H}_c^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\vec{n}\right)-\\&\mathcal{H}_v^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\mathbf{v}_h^+,\nabla\mathbf{u}_h^+,\vec{n}\right)-\mathcal{H}_{ad}^b\left(\epsilon^+,\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\mathbf{v}_h^+,\nabla\mathbf{u}_h^+,\vec{n}\right)ds\end{aligned} \tag{2.2.4}$$

where $\mathcal{H}_c(\cdot,\cdot,\cdot,\cdot,\vec{n})$ is the convective numerical flux, $\mathcal{H}_v(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\vec{n})$ is the viscous numerical flux and $\mathcal{H}_{ad}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\vec{n})$ is the artificial diffusion numerical flux on the interior faces Γ^i. Numerical fluxes take the discontinuous states on each side on the interior faces and compute a unique flux for the face. The numerical fluxes $\mathcal{H}_c^b(\cdot,\vec{n})$, $\mathcal{H}_v^b(\cdot,\cdot,\cdot,\vec{n})$ and $\mathcal{H}_{ad}^b(\cdot,\cdot,\cdot,\cdot,\vec{n})$ denote boundary numerical fluxes (which are different from the interior numerical fluxes) on a boundary edge Γ^b. The $()^+$ and $()^-$ notation refers to the elements on each side of an edge $i\in\mathcal{I}_h$, where a $()^+$ denotes the element with the normal $\vec{n}$ of i pointing out of the element and $()^-$ denotes the element with the normal of i pointing into the element, depicted pictorially in Figure 2.1. In the case of boundary edges, $()^b$ denotes the state at the boundary interface and $()^+$ denotes the state from the element adjacent to the boundary. The boundary edge normal $\vec{n}$ points out of the $()^+$ element as shown in Figure 2.2

Reference [49] has shown that taking $\mathcal{H}_c^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\vec{n}\right)=\mathcal{H}_c\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\vec{n}\right)$ results in a dual inconsistent discretization. Dual consistency is discussed in Chapter 3 where the

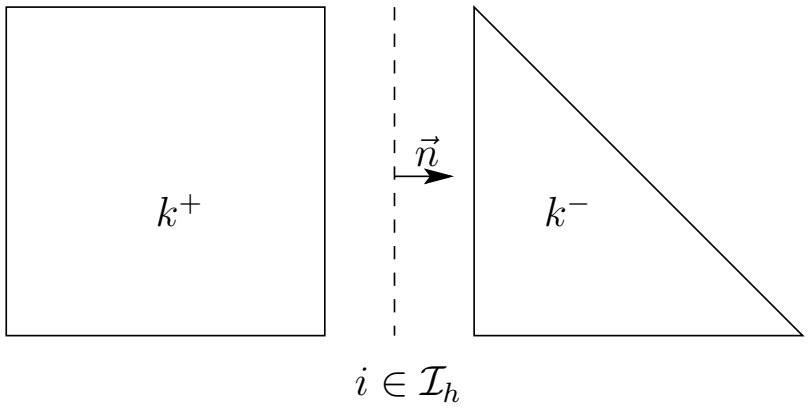


Figure 2.1: Graphical explanation of $\pm$ notation used in edge flux discretization.

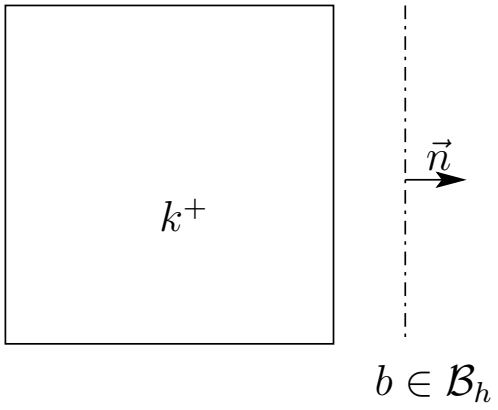


Figure 2.2: Graphical explanation of $()^b$ and $()^+$ notation used in boundary flux discretization.

analysis of this boundary condition will prove the dual inconsistency. For a dual consistent discretization, the boundary numerical flux is taken as $\mathcal{H}_c^b = \vec{\mathbf{F}}_c\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right) \cdot \vec{n}$, which is the convective flux $\vec{\mathbf{F}}_c$ given in equation (2.1.3) normal to the boundary evaluated at the boundary condition state $\mathbf{u}_h^b(\mathbf{u}_h^+)$. The interior convective numerical fluxes are chosen to be approximate Riemann solvers, which are approximations to the exact Riemann problem on the interface. The Riemann problem considers the solution of a system of PDEs at the interface between two discontinuous states e.g. $\mathbf{u}_h^+$ and $\mathbf{u}_h^-$. This solution results in a unique value of the flux at the interface between the two discontinuous states. An approximate Riemann solver considers the same two states $\mathbf{u}_h^+$ and $\mathbf{u}_h^-$ and solves the Riemann problem approximately to generate the single valued interface flux. Reference [50] provides an excellent description of theory and applications of Riemann solvers, as well as descriptions of

several approximate Riemann solvers. Current implementations of approximate Riemann solvers include the flux difference splitting schemes of Rusanov [51] and Roe [52] as well as the flux vector splitting scheme of Hänel and Schwane [53].

The numerical flux for the viscous term is obtained via a modified version of the symmetric interior penalty method (SIP) presented in references [54–57], which seeks to penalize the solution for being discontinuous at the element interfaces. The form of the SIP method used in this work is the same as reference [56], however the penalty parameter value is taken from reference [57]. It is now convenient to introduce the following average and jump operators for both vector and scalar quantities. The average operator is defined for a scalar φ and vector $\vec{\chi}$ by

$$\begin{aligned} \{\phi\} &= \frac{1}{2}\left(\phi^{+} + \phi^{-}\right) \\ \{\vec{\chi}\} &= \frac{1}{2}\left(\vec{\chi}^{+} + \vec{\chi}^{-}\right) \end{aligned} \tag{2.2.5}$$

with the scalar and vector jump operators given by

$$\begin{aligned} [\![\varphi]\!] &= (\varphi^{+} - \varphi^{-})\vec{n} \\ [\![\vec{\chi}]\!] &= (\vec{\chi}^{+} - \vec{\chi}^{-}) \cdot \vec{n} \end{aligned} \tag{2.2.6}$$

respectively. Note that the jump in a scalar quantity is a vector and the jump in a vector quantity is a scalar. Also note that the jump of a vector denoted with a **bold** symbol, which is a vector across the system of equation of size N_f, is obtained by considering each component of the vector as a scalar. Therefore, the jump of $\mathbf{u}_h$ is a matrix with the number of rows equal to the number of equations and the number of columns equal to the number of spatial dimensions in the domain. Using this notation the SIP interior numerical flux is given as

$$\mathcal{H}_v = [\![\mathbf{v}_h^T]\!] \cdot \left\{\vec{\mathbf{F}}_v\left(\mathbf{u}_h, \nabla\mathbf{u}_h\right)\right\} + \left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]^{T_{block}} \nabla\mathbf{v}_h\right\} \cdot [\![\mathbf{u}_h]\!] + \nu [\![\mathbf{v}_h^T]\!] \cdot \left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\right\} [\![\mathbf{u}_h]\!] \tag{2.2.7}$$

and the boundary SIP numerical flux is given by

$$\begin{aligned} \mathcal{H}_v^b = {} & (\mathbf{v}_h^T)^{+} F_v\left(\mathbf{u}_h^b\left(\mathbf{u}_h^{+}\right), \nabla\mathbf{u}_h^{+}\right) \cdot \vec{n} + \left[\mathbf{G}\left(\mathbf{u}_h^b\left(\mathbf{u}_h^{+}\right)\right)\right]^{T_{block}} \nabla\mathbf{v}_h^{+} \cdot \vec{n}\left(\mathbf{u}_h^{+} - \mathbf{u}_h^b\left(\mathbf{u}_h^{+}\right)\right) \\ & + \nu(\mathbf{v}_h^T)^{+}\left[\mathbf{G}\left(\mathbf{u}_h^b\left(\mathbf{u}_h^{+}\right)\right)\right]\left(\mathbf{u}_h^{+} - \mathbf{u}_h^b\left(\mathbf{u}_h^{+}\right)\right)\vec{n} \cdot \vec{n} \end{aligned} \tag{2.2.8}$$

The corresponding SIP numerical fluxes for the artificial diffusion are

$$\begin{aligned}
\mathcal{H}_{ad} =& [\![\mathbf{v}_h^T]\!] \cdot \left\{ \vec{\mathbf{F}}_{ad}\left(\epsilon, \mathbf{u}_h, \nabla \mathbf{u}_h\right) \right\} + \left\{ \left[\mathbf{G}_{ad}\left(\epsilon, \mathbf{u}_h\right)\right]^{T_{block}} \nabla \mathbf{v}_h \right\} \cdot [\![\mathbf{u}_h]\!] + \\
& \nu [\![\mathbf{v}_h^T]\!] \cdot \left\{ \left[\mathbf{G}_{ad}\left(\epsilon, \mathbf{u}_h\right)\right] \right\} [\![\mathbf{u}_h]\!] \\
\mathcal{H}_{ad}^b =& (\mathbf{v}_h^T)^+ \vec{\mathbf{F}}_{av}\left(\epsilon^b, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \nabla \mathbf{u}_h^+\right) \cdot \vec{n} + \\
& \left[\mathbf{G}_{ad}\left(\epsilon^b, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right)\right]^{T_{block}} \nabla \mathbf{v}_h^+ \cdot \vec{n} \left(\mathbf{u}_h^+ - \mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right) + \\
& \nu (\mathbf{v}_h^T)^+ \left[\mathbf{G}_{ad}\left(\epsilon^b, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right)\right] \left(\mathbf{u}_h^+ - \mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right) \vec{n} \cdot \vec{n}
\end{aligned} \tag{2.2.9}$$

For the SIP numerical fluxes the matrix $[\mathbf{G}]$ is actually a block matrix with $d \times d$ blocks and with each block having $N_f \times N_f$ dimensions, where N_f is the number of equations (or fields). The $[\cdot]^{T_{block}}$ indicates a transposing of the blocks of $[\mathbf{G}]$. The blocks of $[\mathbf{G}]$ are defined as derivative of the viscous flux $\vec{\mathbf{F}}_v$ from equation (2.1.3) with respect to the solution gradient $\nabla \mathbf{u}$. In particular for two dimensional flow the blocks of $[\mathbf{G}]$ are given such that

$$\begin{aligned}
\mathbf{F_v}^x &= [G_{11}] \frac{\partial \mathbf{u}_h}{\partial x} + [G_{12}] \frac{\partial \mathbf{u}_h}{\partial y} \\
\mathbf{F_v}^y &= [G_{21}] \frac{\partial \mathbf{u}_h}{\partial x} + [G_{22}] \frac{\partial \mathbf{u}_h}{\partial y}
\end{aligned} \tag{2.2.10}$$

The $[G_{ij}]$ matrices for two dimensional flow are given as:

$$[G_{11}] = \frac{1}{\rho} \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\frac{4}{3}\alpha_v u & \frac{4}{3}\alpha_v & 0 & 0 \\ -\alpha_v v & 0 & \alpha_v & 0 \\ \beta_v \gamma \left(u^2 + v^2 - \frac{E_t}{\rho}\right) - \alpha_v \left(\frac{4}{3}u^2 + v^2\right) & -\beta_v \gamma u + \alpha_v \frac{4}{3} u & -\beta_v \gamma v + \alpha_v v & \beta_v \end{bmatrix}$$

$$[G_{12}] = \frac{1}{\rho} \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{2}{3}\alpha_v v & 0 & -\frac{2}{3}\alpha_v & 0 \\ -\alpha_v u & \alpha_v & 0 & 0 \\ -\alpha_v \frac{1}{3} uv & \alpha_v v & -\frac{2}{3}\alpha_v u & 0 \end{bmatrix}, [G_{21}] = \frac{1}{\rho} \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\alpha_v v & 0 & \alpha_v & 0 \\ \frac{2}{3}\alpha_v u & -\frac{2}{3}\alpha_v & 0 & 0 \\ -\frac{1}{3}\alpha_v uv & -\frac{2}{3}\alpha_v v & \alpha_v u & 0 \end{bmatrix}$$

$$[G_{22}] = \frac{1}{\rho} \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\alpha_v u & \alpha_v & 0 & 0 \\ -\alpha_v v & 0 & \alpha_v & 0 \\ \beta_v \gamma \left(u^2 + v^2 - \frac{E_t}{\rho}\right) - \alpha_v \left(u^2 + \frac{4}{3}v^2\right) & -\beta_v \gamma u + \alpha_v u & -\beta_v \gamma v + \frac{4}{3}\alpha_v v & \beta_v \end{bmatrix}$$

where the α_v and β_v viscous coefficients are defined as:

$$\alpha_v = \mu + \mu_T, \qquad \beta_v = \frac{\mu}{P_r} + \frac{\mu_T}{P_{r_T}}$$

P_r and P_{r_T} represent the laminar and turbulent Prandtl numbers respectively (which are the ratio of momentum diffusivity to thermal diffusivity). The values of the Prandtl numbers are set based on the working fluid of air as $P_r = .72$ and $P_{r_T} = .9$. The $[\mathbf{G}_{ad}]$ is defined as the derivative of the artificial diffusion fluxes in equation (2.1.10) with respect to the solution gradient $\nabla \mathbf{u}$. The $\left[G_{ad_{ij}}\right]$ for the artificial diffusion operator are diagonal matrices since the artificial diffusion operator is a Laplacian type operator:

$$[G_{ad_{11}}] = \begin{bmatrix} \hat{\epsilon}\frac{h_x}{h} & 0 & 0 & 0 \\ 0 & \hat{\epsilon}\frac{h_x}{h} & 0 & 0 \\ 0 & 0 & \hat{\epsilon}\frac{h_x}{h} & 0 \\ 0 & 0 & 0 & \hat{\epsilon}\frac{h_x}{h} \end{bmatrix}, [G_{ad12}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[G_{ad_{21}}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, [G_{ad_{22}}] = \begin{bmatrix} \hat{\epsilon}\frac{h_y}{h} & 0 & 0 & 0 \\ 0 & \hat{\epsilon}\frac{h_y}{h} & 0 & 0 \\ 0 & 0 & \hat{\epsilon}\frac{h_y}{h} & 0 \\ 0 & 0 & 0 & \hat{\epsilon}\frac{h_y}{h} \end{bmatrix}$$

The final piece of the discretization is the approximation of $\mathbf{u} \in \mathcal{V}$ by $\mathbf{u}_h \in \mathcal{V}_h^p$ for each element. In particular $\mathcal{V}_h^p$ is the space spanned by the polynomials $\{\phi_i, i = 1..M\}$ where M is the number of polynomials or modes required to specify a complete basis of order p on an element. The set of functions chosen are Legendre polynomial-based bubble functions ϕ_i from reference [58], which are so-called modal basis functions. The discrete solution $\mathbf{u}_h$ for an element is given as a sum of the unknown coefficients times the basis functions

$$\mathbf{u}_h \equiv \mathbf{u}_h(\vec{x}, t) = \sum_{j=1}^{M} \hat{\mathbf{u}}_j(t)\phi_j(\vec{x}) \tag{2.2.11}$$

where M is the number of modes defining the truncation level. The semi-discrete discontinuous Galerkin formulation (*i.e.* continuous in time) is given by choosing $\mathbf{v}_h$ to be each

$\phi_i, \forall i = 1...M$, resulting in $N_f \times M$ equations for each element $k \in \mathcal{T}_h$ where N_f is the number of equations (fields) in the system of Navier-Stokes equations.

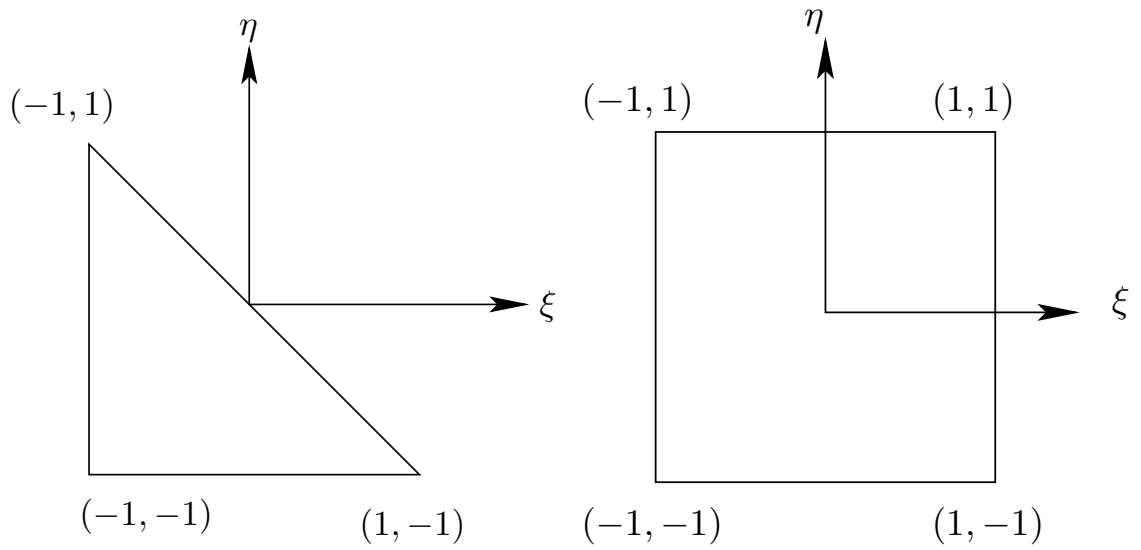
The choice of the penalty parameter(ν) can be rather ad-hoc as the value is only required to be "large enough" to stabilize the scheme. However, Shahbazi in reference [54] derived an explicit expression for the penalty parameter for Poisson's equation. A modified version of this expression given in reference [57] has been successfully implemented in this work. The value of the penalty parameter on an interface is taken as

$$\nu = max\left(M^+, M^-\right) max\left(\frac{|\partial\Omega_k^-|}{|\Omega_k^-|}, \frac{|\partial\Omega_k^+|}{|\Omega_k^+|}\right) \tag{2.2.12}$$

where $|\Omega_k^{\pm}|$,$|\partial\Omega_k^{\pm}|$ and $M^{\pm}$ are the area, perimeter and number of modes of elements $k^{\pm}$ respectively and where the $(\quad)^{\pm}$ denotes the elements on each side of the interface. Note that as the discretization order p is increased the number of modes M in an element increases, hence the number of degrees of freedom(DoFs) within an element is coupled to the order of accuracy.

2.3 Basis Functions

As mentioned previously, the discrete DG solution $\mathbf{u}_h$ is expanded in a series of basis functions $\{\phi_i, i = 1, ...M\}$ and corresponding coefficients $\{\hat{\mathbf{u}}_i(t), i = 1, ...M\}$ where M is the number of modes and is chosen such that a complete basis of order p is obtained. Various basis functions exist and this work uses a set of hierarchical basis functions ϕ_i, which take the form of modal shape functions in a standard element that spans $\{-1 < \xi < 1, -1 < \eta < 1\}$. The standard elements for a triangle and a quadrilateral are shown in Figure 2.3(a) and Figure 2.3(b) respectively. The basis functions are hierarchical, meaning that a basis of order p contains all the functions from a basis of order $p-1$. This work employs meshes that contain both triangles and quadrilaterals and the basis functions are defined for triangular and quadrilateral elements separately. The basis functions are based on combinations of one



(a) Standard Triangle (b) Standard Quadrilateral

Figure 2.3: Standard triangle and standard quadrilateral on which basis functions are defined.

dimensional Legendre polynomials, which are given by

$$\begin{aligned} L_n(x) &= \frac{(2n+1)\,xL_{n-1}(x) - nL_{n-2}(x)}{n+1} \\ L_0(x) &= 1 \\ L_1(x) &= x \end{aligned} \tag{2.3.1}$$

where n is the polynomial order.

2.3.1 Triangular Elements

For triangular elements the basis functions take form of generalized Legendre polynomials defined on the standard triangle [58]. Consider a local polynomial of order p for an element. The complete basis set of order p is made up of the $M = (p+1)(p+2)/2$ Legendre polynomials given by

$$\phi_i = L_{n_1}(\lambda_3 - \lambda_2)\, L_{n_2}(\lambda_2 - \lambda_1), 0 \le n_1, n_2; n_1 + n_2 \le p \tag{2.3.2}$$

where

$$\begin{aligned}\lambda_1 &= \frac{\eta+1}{2} \\ \lambda_2 &= -\frac{\xi+\eta}{2} \\ \lambda_3 &= \frac{\xi+1}{2}\end{aligned} \tag{2.3.3}$$

and L_{n_1} is the 1-D Legendre polynomial of order n_1, which is given by equation (2.3.1). This basis is hierarchical making it ideal for p-multigrid solvers, as discussed in Section 4.4.1, and contains $p = 0$, i.e. the constant function within the hierarchical structure. Figure 2.4 depicts the basis functions of order $p = 3$ for a triangular element.

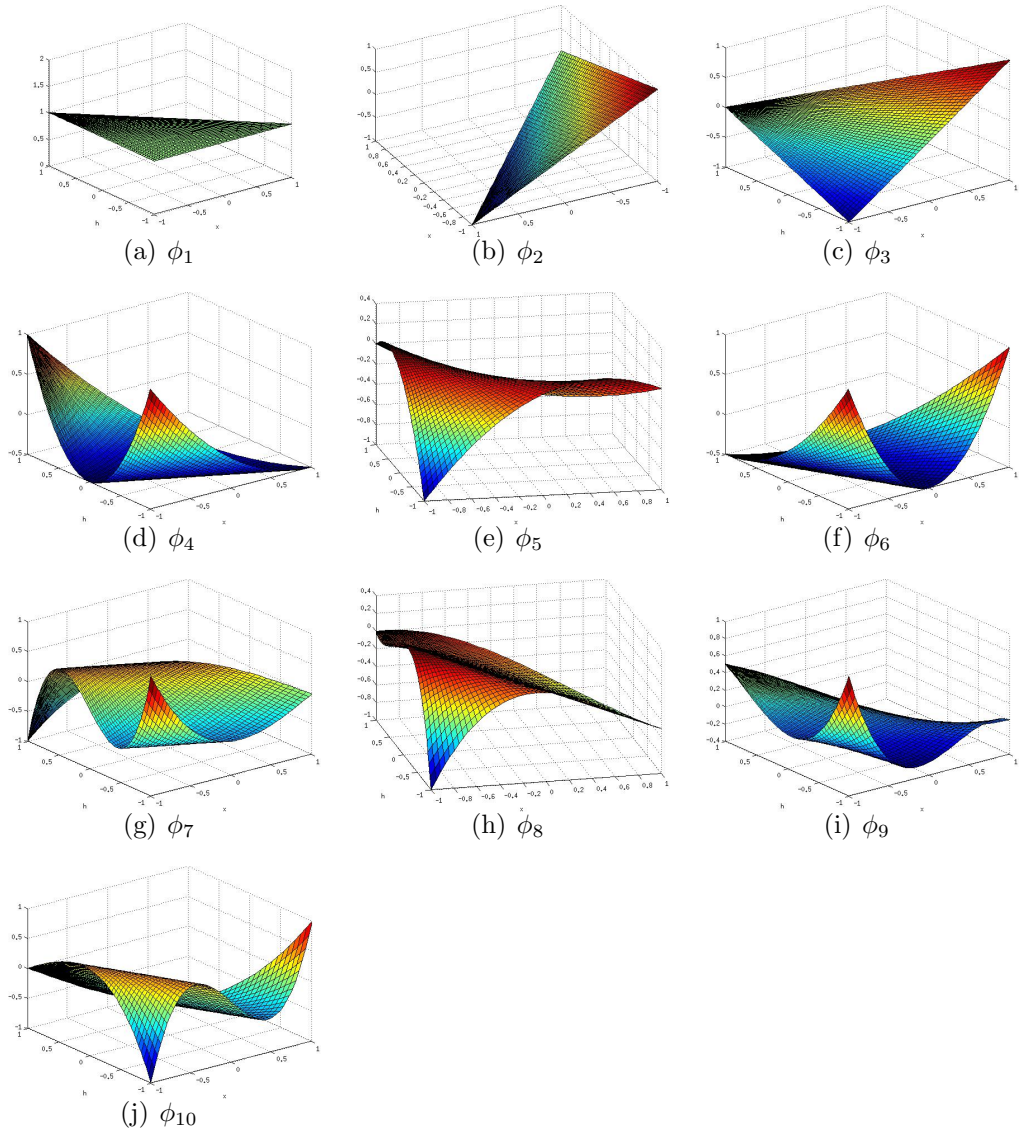
Reference [59] has made use of a set of so-called H^1 hierarchical shape functions, which do not have $p = 0$ as part of the hierarchical structure. These H^1 basis functions are particularly useful for geometry mappings and are used to generate the mapping from the reference element to the physical element. The H^1 designation means that the lowest order polynomial in the set is a linear or $p = 1$ polynomial. Details of these functions and their formulation can be found in Section 2.4 as well as in references [58, 59].

2.3.2 Quadrilaterals

The basis functions employed for quadrilateral elements are similar to those employed for triangular elements. The functions are based on Legendre polynomials and are also hierarchical and have $p = 0$ within the hierarchical structure. Again consider a polynomial basis of order p that is now made up of $M = (p+1)^2$ modes. This is actually more functions than required for a polynomial basis of order p but still constitutes a complete basis set of order p. The basis functions are a tensor product of 1-D Legendre Polynomials(equation (2.3.1)) that span $\{-1 < \xi < 1, -1 < \eta < 1\}$ from reference [58].

$$\phi_i = L_{n_1}(\xi)\, L_{n_2}(\eta), \quad 0 \le n_1, 0 \le n_2 \tag{2.3.4}$$

Figure 2.5 depicts the basis functions of order $p = 3$ for a quadrilateral element. Although these functions are a tensor product, no effort is made in the implementation to take advantage of this property and these function are implemented in the same way as the basis functions on the triangle.



(a) ϕ_1 (b) ϕ_2 (c) ϕ_3

(d) ϕ_4 (e) ϕ_5 (f) ϕ_6

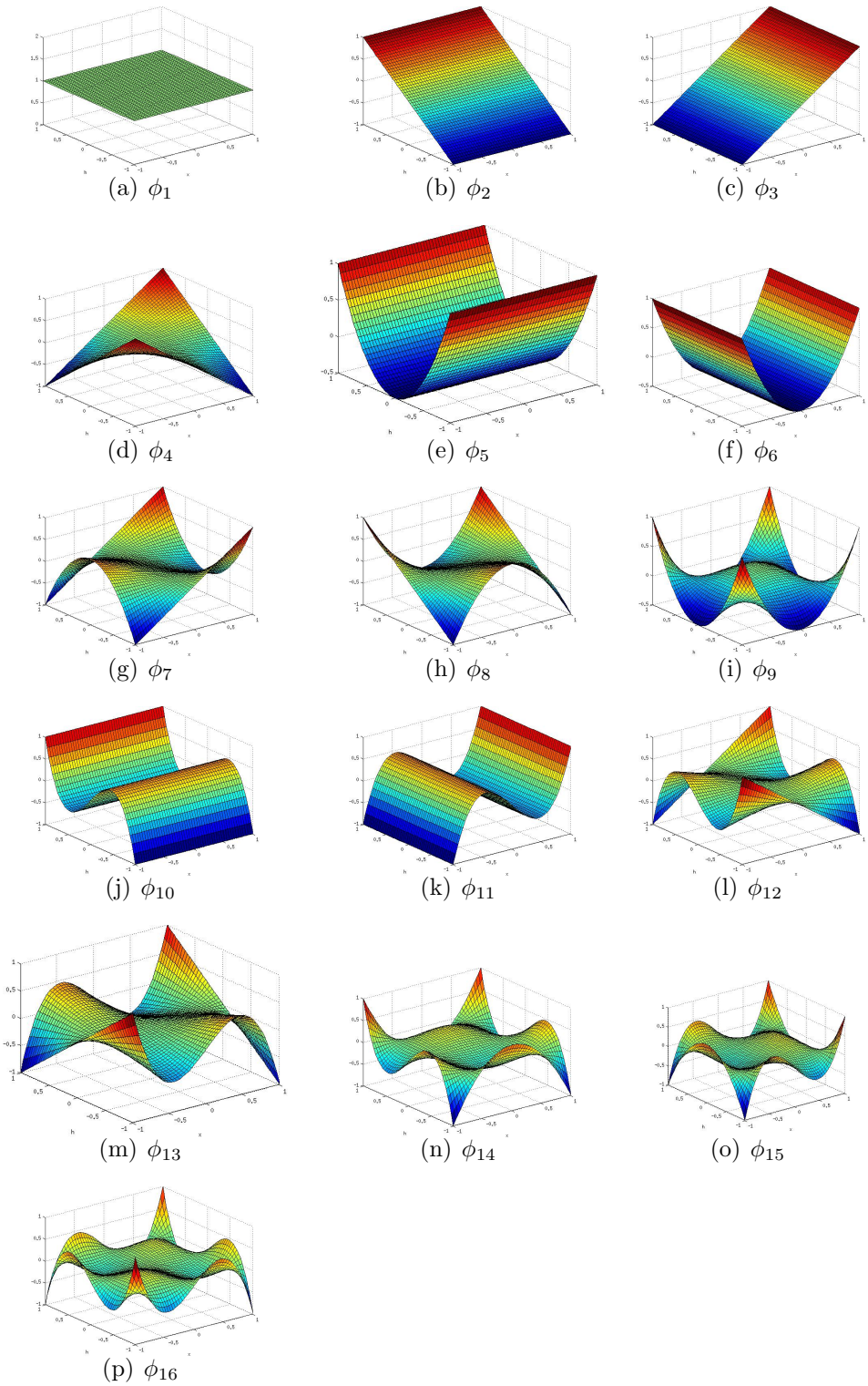
(g) ϕ_7 (h) ϕ_8 (i) ϕ_9

(j) ϕ_{10}

Figure 2.4: Complete basis function set for $p = 3$ discretization on triangular elements.

2.4 Element Mappings

In order to compute integrals and define basis functions on any given physical element in the mesh, a mapping from the standard element in (ξ, η) to the physical element in (x, y), is generated using a set of mapping basis functions, denoted $\{\psi_j\}$. The elements are mapped to $p_{max} + 1$ order where p_{max} is the maximum solution discretization order in the mesh. Let the mapping order be represented as $p_{map} = p_{max} + 1$. Mathematically the mapping for an



(a) ϕ_1 (b) ϕ_2 (c) ϕ_3

(d) ϕ_4 (e) ϕ_5 (f) ϕ_6

(g) ϕ_7 (h) ϕ_8 (i) ϕ_9

(j) ϕ_{10} (k) ϕ_{11} (l) ϕ_{12}

(m) ϕ_{13} (n) ϕ_{14} (o) ϕ_{15}

(p) ϕ_{16}

Figure 2.5: Complete basis function set for $p = 3$ discretization on quadrilateral elements.

element amounts to defining the $\vec{x}$ coordinates by:

$$\begin{aligned} x_k(\xi,\eta) &= \sum_{j=1}^{M_{map}} \hat{x}_{k_j}\psi_j(\xi,\eta) \\ y_k(\xi,\eta) &= \sum_{j=1}^{M_{map}} \hat{y}_{k_j}\psi_j(\xi,\eta) \end{aligned} \tag{2.4.1}$$

where the ψ_j basis function is a set of hierarchical basis functions which belong to the H^1 space, i.e. the lowest degree of any function in the set is a linear or $p = 1$ polynomial. The corresponding mapping Jacobian and mapping Jacobian inverse for an element k are given by:

$$\begin{aligned} [J_k] &= \begin{bmatrix} \frac{\partial x_k}{\partial \xi} & \frac{\partial x_k}{\partial \eta} \\ \frac{\partial y_k}{\partial \xi} & \frac{\partial y_k}{\partial \eta} \end{bmatrix} \\ [J_k]^{-1} &= \frac{1}{|J_k|}\begin{bmatrix} \frac{\partial y_k}{\partial \eta} & -\frac{\partial x_k}{\partial \eta} \\ -\frac{\partial y_k}{\partial \xi} & \frac{\partial x_k}{\partial \xi} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x_k} & \frac{\partial \xi}{\partial y_k} \\ \frac{\partial \eta}{\partial x_k} & \frac{\partial \eta}{\partial y_k} \end{bmatrix} \end{aligned} \tag{2.4.2}$$

The mapping from the standard or reference element to the physical element is depicted in Figure 2.6.

The set of functions used to map the element from the standard element to the physical element are a set of localized *vertex*, *edge* and *bubble* functions as described in [58]. These functions are particularly useful due to the decoupling of the *vertex*, *edge* and *bubble* functions as seen in Figure 2.7 and Figure 2.8 where the functions on a given edge are zero on all other edges. This decoupling allows for the specification of a reference map of order p_{map} using only $p_{map}+1$ surface points. These basis functions are all based on a Jacobi polynomial kernel Φ_n given as

$$\begin{aligned} \Phi_n(z) &= -2\frac{\sigma}{n-1}\mathsf{P}_{n-1}^{1,1}(z) \\ \sigma &= \frac{1}{\sqrt{\frac{2}{2n-1}}} \end{aligned} \tag{2.4.3}$$

where the $\mathsf{P}_n^{\alpha,\beta}$ is the Jacobi polynomial of order n with weights α and β. The Jacobi

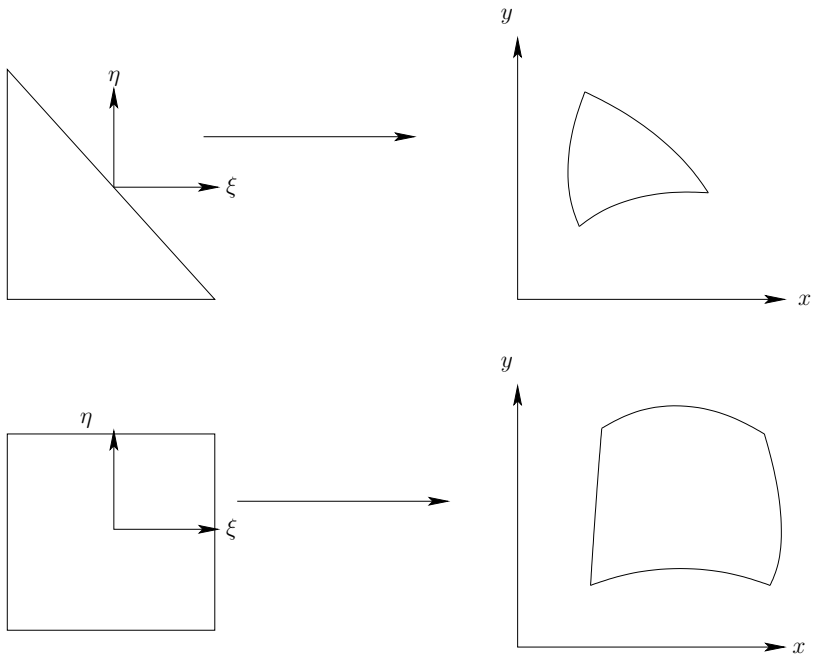


Figure 2.6: Diagram of mapping the standard triangle and quadrilateral to the arbitrarily curved one in physical space.

polynomials can be generated via the recurrence relation

$$\begin{aligned}
\mathsf{P}_0^{\alpha,\beta}(z) &= 1 \\
\mathsf{P}_1^{\alpha,\beta}(z) &= \frac{1}{2}\left[\alpha-\beta+(\alpha+\beta+2)\,z\right] \\
\mathsf{P}_n^{\alpha,\beta}(z) &= (a_{n_2}+a_{n_3}z)\,\mathsf{P}_{n-1}^{\alpha,\beta}(z) - a_{n_4}\mathsf{P}_{n-2}^{\alpha,\beta}(z) \\
a_{n_1} &= 2\,(n+1)\,(n+\alpha+\beta+1)\,(2n+\alpha+\beta) \\
a_{n_2} &= (2n+\alpha+\beta+1)\,\alpha^2-\beta^2 \\
a_{n_3} &= (2n+\alpha+\beta)\,(2n+\alpha+\beta+1)\,(2n+\alpha+\beta+2) \\
a_{n_4} &= 2\,(n+\alpha)\,(n+\beta)\,(2n+\alpha+\beta+2)
\end{aligned} \tag{2.4.4}$$

from reference [60], which also contains additional useful formulas. The complete set of mapping functions for the triangle are defined by *vertex* functions

$$\begin{aligned}
\psi^{v_1} &= -\frac{\xi+\eta}{2} \\
\psi^{v_2} &= \frac{\eta+1}{2} \\
\psi^{v_3} &= \frac{\xi+1}{2}
\end{aligned} \tag{2.4.5}$$

edge functions

$$\begin{aligned}\psi_n^{e_1} &= \psi^{v_1}\psi^{v_2}\Phi_{n-2}\left(\psi^{v_2}-\psi^{v_1}\right)\\ \psi_n^{e_2} &= \psi^{v_2}\psi^{v_3}\Phi_{n-2}\left(\psi^{v_3}-\psi^{v_2}\right)\\ \psi_n^{e_3} &= \psi^{v_3}\psi^{v_1}\Phi_{n-2}\left(\psi^{v_1}-\psi^{v_3}\right)\end{aligned} \tag{2.4.6}$$

and *bubble* functions

$$\begin{aligned}\psi_{n_1,n_2}^{b} = \psi^{v_3}\psi^{v_1}\psi^{v_2}\Phi_{n_1-2}\left(\psi^{v_2}-\psi^{v_1}\right)\Phi_{n_2-2}\left(\psi^{v_1}-\psi^{v_3}\right),\\ \left\{n_1,n_2 : 1\le n_1,n_2; n_1+n_2\le p^b-1\right\}\end{aligned} \tag{2.4.7}$$

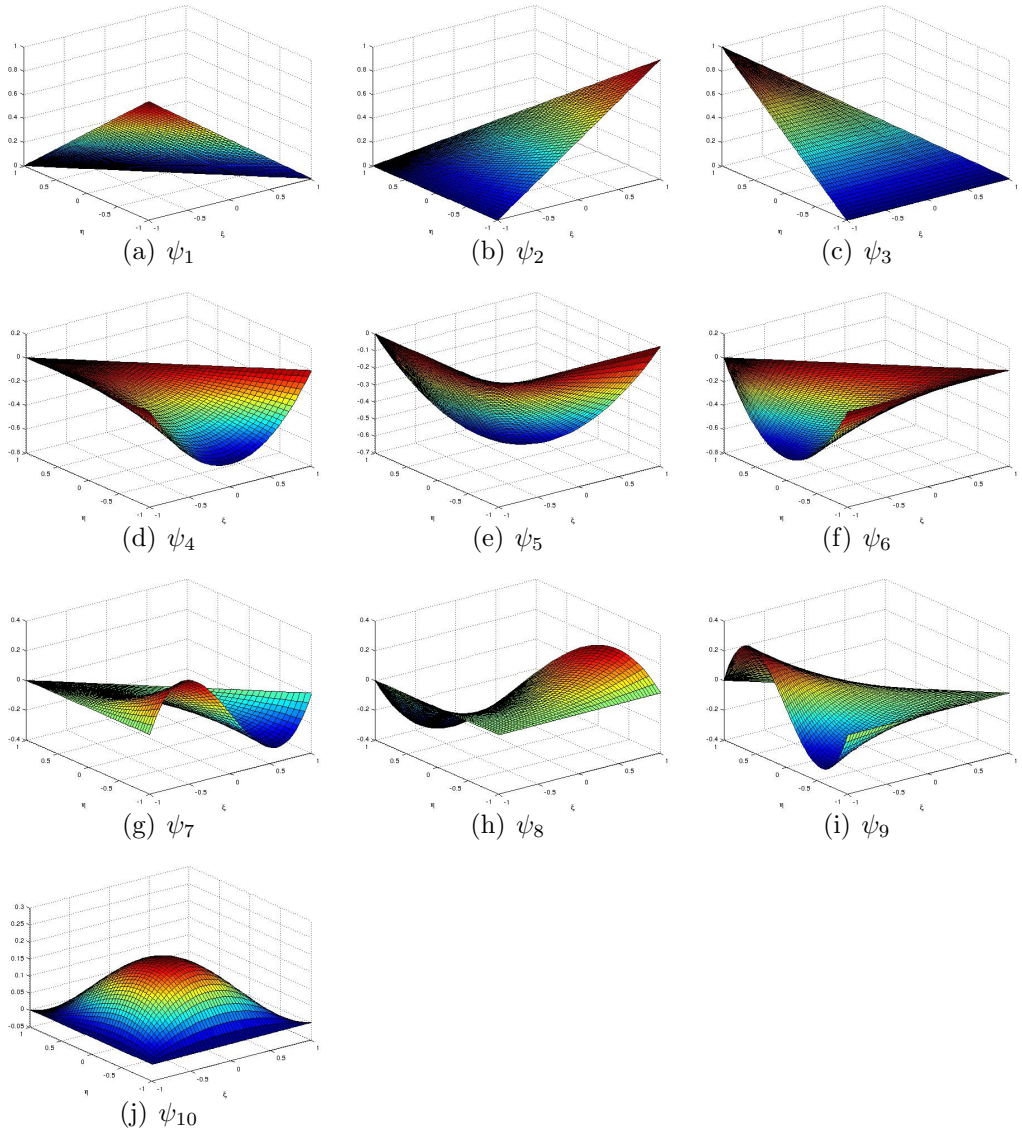
where superscripts v, e, and b denote *vertex*, *edge* and *bubble* respectively. The order of the *edge* and *bubble* functions is the same as p_{map}, i.e. $p^e = p^b = p_{map}$, where p^e is the edge function order and p^b is the bubble function order. The mapping basis functions of order $p_{map} = 3$ are depicted in Figure 2.7. The separation of the *vertex*, *edge* and *bubble* modes is seen in Figure 2.7, which shows that functions defined for the edges (e.g. ψ_4) are zero at the vertices and for all edges other than the edge on which they are defined. Similarly, bubble functions such as ψ_{10} are zero on all edges and vertices of the element.

Quadrilaterals mappings are defined by an analogous set of basis functions which are also given as *vertex* functions

$$\begin{aligned}\psi^{v_1} &= \left(\frac{1-\xi}{2}\right)\left(\frac{1-\eta}{2}\right)\\ \psi^{v_2} &= \left(\frac{1+\xi}{2}\right)\left(\frac{1-\eta}{2}\right)\\ \psi^{v_3} &= \left(\frac{1-\xi}{2}\right)\left(\frac{1+\eta}{2}\right)\\ \psi^{v_4} &= \left(\frac{1+\xi}{2}\right)\left(\frac{1+\eta}{2}\right)\end{aligned} \tag{2.4.8}$$

edge functions

$$\begin{aligned}\psi_n^{e_1} &= \left(\frac{1-\xi}{2}\right)\left(\frac{1-\eta}{2}\right)\left(\frac{1+\eta}{2}\right)\Phi_{n-2}\left(\eta\right)\\ \psi_n^{e_2} &= \left(\frac{1+\xi}{2}\right)\left(\frac{1-\eta}{2}\right)\left(\frac{1+\eta}{2}\right)\Phi_{n-2}\left(\eta\right)\\ \psi_n^{e_3} &= \left(\frac{1-\eta}{2}\right)\left(\frac{1-\xi}{2}\right)\left(\frac{1+\xi}{2}\right)\Phi_{n-2}\left(\xi\right)\\ \psi_n^{e_4} &= \left(\frac{1+\eta}{2}\right)\left(\frac{1-\xi}{2}\right)\left(\frac{1+\xi}{2}\right)\Phi_{n-2}\left(\xi\right)\end{aligned} \tag{2.4.9}$$



(a) ψ_1 (b) ψ_2 (c) ψ_3

(d) ψ_4 (e) ψ_5 (f) ψ_6

(g) ψ_7 (h) ψ_8 (i) ψ_9

(j) ψ_{10}

Figure 2.7: Complete mapping basis function set for $p_{map} = 3$ on triangular elements.

and *bubble* functions

$$\phi^b_{n_1,n_2} = \left(\frac{1-\xi}{2}\right)\left(\frac{1+\xi}{2}\right)\Phi_{n_1-2}(\xi)\left(\frac{1-\eta}{2}\right)\left(\frac{1+\eta}{2}\right)\Phi_{n_2-2}(\eta), \quad \{n_1, n_2 : 2 \le n_1, n_2 \le p^b\} \tag{2.4.10}$$

where the superscripts v, e, and b denote *vertex*, *edge* and *bubble* respectively. The order of the *edge* and *bubble* functions is the same as p_{map}, i.e. $p^e = p^b = p_{map}$, where p^e is the *edge* function order and p^b is the *bubble* function order. The mapping basis functions for

quadrilaterals of order $p_{map} = 3$ are shown in Figure 2.8.

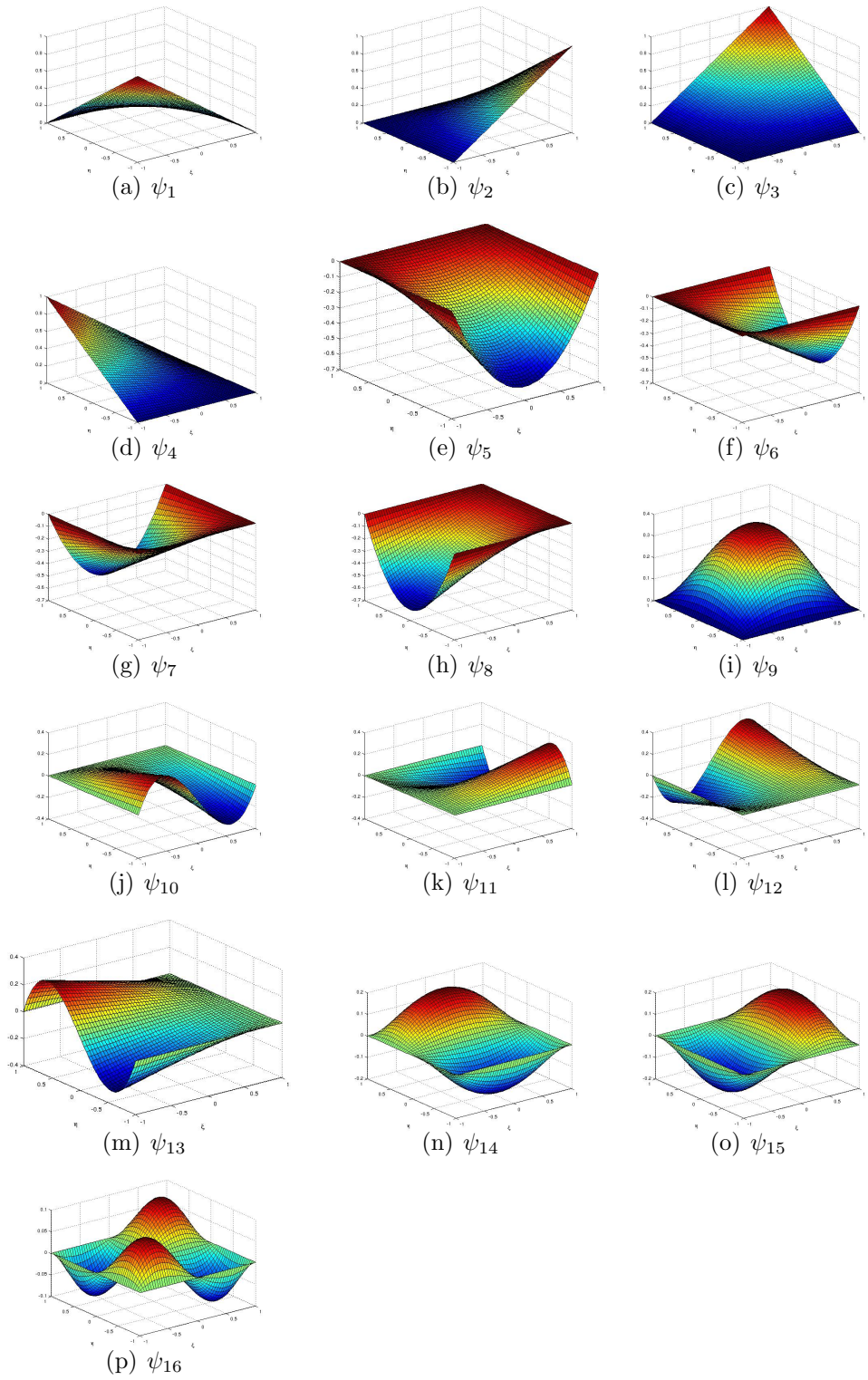
The surface points, which are the physical points that are interpolated to the mapping basis functions to define the element mappings, encompass the two nodes an a boundary edge and additional points interior to the edge, which are all given by the mesh generator as shown in Figure 2.9. The black points in Figure 2.9 are the element nodal points and the red points are the additional geometry points given by the mesh generator. The mapping modal coefficients $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are found by interpolating the x and y coordinates to the mapping basis functions, following:

$$\begin{aligned} \hat{\mathbf{x}} &= [V]^{-1}\,\mathbf{x} \\ \hat{\mathbf{y}} &= [V]^{-1}\,\mathbf{y} \\ V_{ij} &= \psi_i\left(\xi_j, \eta_j\right) \\ x_j &= x\left(\xi_j, \eta_j\right) \\ y_j &= y\left(\xi_j, \eta_j\right) \end{aligned} \tag{2.4.11}$$

where $[V]$ is the Vandermonde matrix found by evaluating the mapping basis functions $\{\psi_j\}$ at the standard element locations(ξ_j, η_j) at both the red and black points in Figure 2.9. The vectors $\mathbf{x}$ and $\mathbf{y}$ are the coordinates given by the mesh generator at the points illustrated in Figure 2.9.

2.5 Numerical quadrature

The set of discrete equations (2.2.3) is solved in modal space and the integrals are evaluated using Gaussian quadrature rules [58, 61, 62]. Modal space is the space $\mathcal{V}_h^p$ that is made up of the basis functions in Section 2.3. These quadrature rules require projection of the solution from the modal space to the quadrature points. To preserve the $p+1$ accuracy of the DG scheme, the volume integrals, which are the $\int_{\Omega_k}$ terms in equation (2.2.4), are computed using a rule that integrates a polynomial of degree $2p$ exactly. To the same end, surface and boundary surface integrations, which are the $\int_{\Gamma^i}$ and $\int_{\Gamma^b}$ terms in equation (2.2.4) respectively, are carried out with a rule that integrates a polynomial of degree $2p+1$



(a) ψ_1 (b) ψ_2 (c) ψ_3

(d) ψ_4 (e) ψ_5 (f) ψ_6

(g) ψ_7 (h) ψ_8 (i) ψ_9

(j) ψ_{10} (k) ψ_{11} (l) ψ_{12}

(m) ψ_{13} (n) ψ_{14} (o) ψ_{15}

(p) ψ_{16}

Figure 2.8: Complete mapping basis function set for $p_{map} = 3$ on quadrilateral elements.

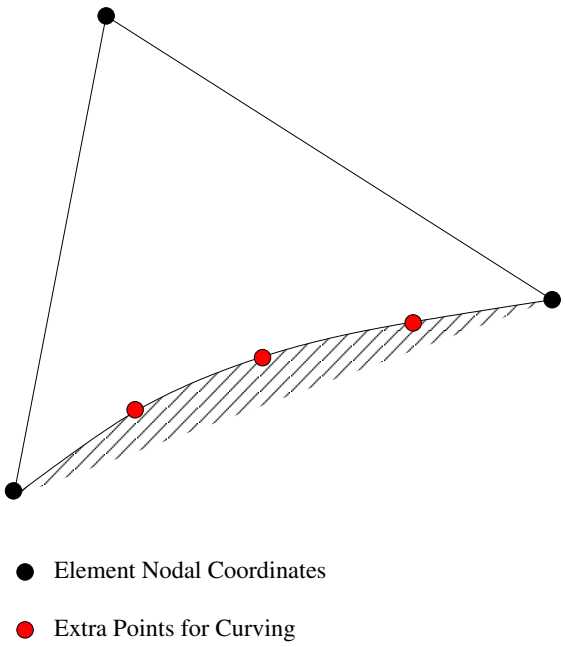


Figure 2.9: Diagram of the points used to construct curved elements on a boundary. Black points are the element nodes and red points are the additional geometry points used to define the curved element mapping

exactly [63]. Figure 2.10 shows an element stencil with the quadrature points employed for integrating a $p = 3$ solution on a mixed element mesh.

In order to apply Gaussian quadrature, the integrals must first be mapped into the standard element. For volume integrals the integration is transformed as

$$\begin{aligned}&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\mathbf{v}_h^T\cdot\left(\vec{\mathbf{F}}_c\left(\mathbf{u}_h\right)-\vec{\mathbf{F}}_v\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)-\vec{\mathbf{F}}_{ad}\left(\epsilon,\mathbf{u}_h,\nabla\mathbf{u}_h\right)\right)d\Omega_k=\\&\sum_{k\in\mathcal{T}_h}\int_{\xi_{min}}^{\xi_{max}}\int_{\eta_{min}}^{\eta_{max}}\nabla\mathbf{v}_h^T\cdot\left(\vec{\mathbf{F}}_c\left(\mathbf{u}_h\right)-\vec{\mathbf{F}}_v\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)-\vec{\mathbf{F}}_{ad}\left(\epsilon,\mathbf{u}_h,\nabla\mathbf{u}_h\right)\right)|J_k|d\eta d\xi\end{aligned}\tag{2.5.1}$$

where the gradient is defined by:

$$\nabla=\left(\frac{\partial}{\partial\xi}\frac{\partial\xi}{\partial x}+\frac{\partial}{\partial\eta}\frac{\partial\eta}{\partial x},\frac{\partial}{\partial\xi}\frac{\partial\xi}{\partial y}+\frac{\partial}{\partial\eta}\frac{\partial\eta}{\partial y}\right)^T\tag{2.5.2}$$

Let the integrand in equation (2.5.1) be denoted as $\mathcal{F}\left(\xi,\eta\right)$, then the integral in equation (2.5.1) is approximated using Gaussian quadrature as

$$\sum_{k\in\mathcal{T}_h}\int_{\xi_{min}}^{\xi_{max}}\int_{\eta_{min}}^{\eta_{max}}\mathcal{F}\left(\xi,\eta\right)d\eta d\xi\approx\sum_{k\in\mathcal{T}_h}\sum_{i_q=1}^{N_q}\mathcal{F}\left(\xi_{i_q},\eta_{i_q}\right)w_{i_q}\tag{2.5.3}$$

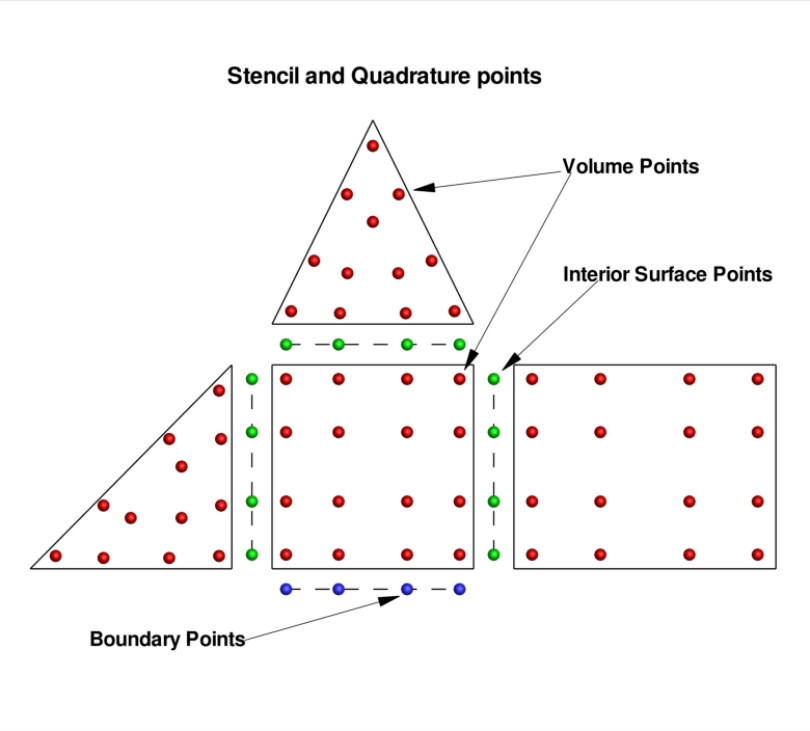


Figure 2.10: Example quadrature points for $p = 3$ discretization. Volume integral: red points, surface integral: green points, boundary surface: blue points.

where ξ_{i_q} and η_{i_q} are the quadrature point locations in the standard element and w_{i_q} are the quadrature weights. For volume integrals, the location, number of points and weights are selected such that a polynomial of degree $2p$ is integrated exactly. Surface integrals are computed in a similar fashion. For interior surfaces the integration in the standard element is given by:

$$\begin{aligned}
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\mathcal{H}_c\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\vec{n}\right)-\mathcal{H}_v\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\nabla\mathbf{u}_h^+,\nabla\mathbf{u}_h^-,\vec{n}\right)-\\
&\mathcal{H}_{ad}\left(\epsilon^+,\epsilon^-,\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\nabla\mathbf{u}_h^+,\nabla\mathbf{u}_h^-,\vec{n}\right)ds=\\
&\sum_{i\in\mathcal{I}_h}\int_{-1}^{1}\left(\mathcal{H}_c\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\vec{n}\right)-\mathcal{H}_v\left(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\nabla\mathbf{u}_h^+,\nabla\mathbf{u}_h^-,\vec{n}\right)-\right.\\
&\left.\mathcal{H}_{ad}\left(\epsilon^+,\epsilon^-,\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{v}_h^+,\mathbf{v}_h^-,\nabla\mathbf{u}_h^+,\nabla\mathbf{u}_h^-,\vec{n}\right)\right)\sqrt{\frac{dx}{ds}^2+\frac{dy}{ds}^2}\,ds
\end{aligned}\tag{2.5.4}$$

which is also approximated by Gaussian quadrature. The quadrature point locations and weights are defined on the edges of the elements. Let the integrand in equation (2.5.4) be

denoted by $\mathcal{F}_i(\xi,\eta)$. The integration of equation (2.5.4) is approximated as

$$\sum_{i\in\mathcal{I}_h}\int_{-1}^{1}\mathcal{F}_i(\xi,\eta)\,ds \approx \sum_{i\in\mathcal{I}_h}\sum_{i_q=1}^{N_q}\mathcal{F}_i\left(\xi_{i_q},\eta_{i_q}\right)w_{i_q} \tag{2.5.5}$$

where quadrature point locations $\left(\xi_{i_q},\eta_{i_q}\right)$ and weights w_{i_q} are obtained from a rule that integrates polynomials of $2p+1$ exactly on the edges. The boundary surface integral is transformed in the same way as the interior surface integral.

$$\begin{aligned}
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathcal{H}_c^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\vec{n}\right)-\\
&\mathcal{H}_v^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\mathbf{v}_h^+,\nabla\mathbf{u}_h^+,\vec{n}\right)-\mathcal{H}_{ad}^b\left(\epsilon^+,\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\mathbf{v}_h^+,\nabla\mathbf{u}_h^+,\vec{n}\right)ds=\\
&\sum_{b\in\mathcal{B}_h}\int_{-1}^{1}\left(\mathcal{H}_c^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\vec{n}\right)-\mathcal{H}_v^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\mathbf{v}_h^+,\nabla\mathbf{u}_h^+,\vec{n}\right)-\right.\\
&\left.\mathcal{H}_{ad}^b\left(\epsilon^+,\mathbf{u}_h^b\left(\mathbf{u}_h^+\right),\mathbf{v}_h^+,\nabla\mathbf{u}_h^+,\vec{n}\right)\right)\sqrt{\frac{dx}{ds}^2+\frac{dy}{ds}^2}\,ds
\end{aligned} \tag{2.5.6}$$

Let the integrand in equation (2.5.6) be denoted by $\mathcal{F}_b(\xi,\eta)$. The integral in equation (2.5.6) is approximated using Gaussian quadrature as

$$\sum_{b\in\mathcal{B}_h}\int_{-1}^{1}\mathcal{F}_b(\xi,\eta)\,ds \approx \sum_{b\in\mathcal{B}_h}\sum_{i_q=1}^{N_q}\mathcal{F}_b\left(\xi_{i_q},\eta_{i_q}\right)w_{i_q} \tag{2.5.7}$$

where the quadrature point locations and weights are the same as for the interior surface integrations. The Gaussian quadrature point locations and weights used in this work are tabulated in reference [58].

2.6 Turbulence Model Discretization

Due to the aforementioned turbulence model discontinuity discussed in Chapter 1, this work employs an alternative discretization which makes use of a first-order finite-volume discretization of the convection terms for the turbulence model in equation (2.1.2). This discretization has been successful in many applications such as those in references [5,41,64–66]. In order to maintain an appropriate testing method, two options are available for discretizing the turbulence model equation. The standard DG discretization can be utilized or the aforementioned

first-order finite-volume discretization can be employed. The standard DG discretization is exactly the same as previously described for the mean flow equations. The first-order finite-volume discretization is carried out in a slightly different manner due to the lack of turbulence model variable gradients (i.e. a first-order discretization cannot contain gradient information).

This section will discuss the first-order convection term finite-volume discretization of the SA model as well as how it is coupled to the DG discretization of the RANS equations. Unfortunately by discretizing the turbulence model to first-order accuracy, there is now a resolution discrepancy between the mean flow equations and the model equation. This is inevitable for a DG method because of the coupling between degrees of freedom (DoFs) and order of accuracy. In order to have a first-order finite-volume representation and not reconstruct model variable gradients, the diffusive source term must be cast in a non-conservative form, as shown in reference [41]. This allows one to place the entire diffusion term on the surface in order to avoid requiring gradients of the SA model variable. The turbulence model is discretized on a mesh of elements $\mathcal{T}_{h,0}$ and the discrete solution is in the space of piecewise constant functions $\rho\tilde{\nu}_h \in \mathcal{V}_h^0$. Substitution of the discrete solution into the model equation given in equation (2.1.2) and integration over the domain yields.

$$\begin{aligned}&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\left[\frac{\partial\rho\tilde{\nu}_h}{\partial t}+\frac{\partial\left(u_h\rho\tilde{\nu}_h\right)}{\partial x}+\frac{\partial\left(v_h\rho\tilde{\nu}_h\right)}{\partial y}-\frac{1}{\sigma}\nabla\cdot\left(\left(\mu_h+\rho\tilde{\nu}_h\right)\left(1+c_{b_2}\right)\nabla\tilde{\nu}_h\right)-\right.\\&\left.c_{b_2}\rho\tilde{\nu}_h\nabla^2\tilde{\nu}_h-S_h\right]d\Omega_k=0\\&\text{with}\\&S_h=\mathcal{P}\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)-\mathcal{D}\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)\end{aligned}\tag{2.6.1}$$

where $\mathcal{P}\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)$ is given in equation (B.1.3) and $\mathcal{D}\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)$ is given in equation (B.1.6). Integration by parts of equation (2.6.1) results in

$$\begin{aligned}&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\frac{\partial\rho\tilde{\nu}_h}{\partial t}-S_h\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)d\Omega_k+\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\mathcal{H}_c\left(\mathbf{u}_h\right)-\left(\mathcal{H}_v\left(\mathbf{u}_h\right)\right)^{\pm}ds+\\&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathcal{H}_c^b\left(\mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right)+\mathcal{H}_v^b\left(\mathbf{u}_h^+,\mathbf{u}_h^b\left(\mathbf{u}_h^+\right)\right)ds=0\end{aligned}\tag{2.6.2}$$

where

$$\begin{aligned}\mathcal{H}_c &= \frac{1}{2}\left[\left(\tilde{u}_h\rho\tilde{\nu}_h\right)^+ + \left(\tilde{u}_h\rho\tilde{\nu}_h\right)^- + max\left(\tilde{u}_h^+, \tilde{u}_h^-\right)\left(\rho\tilde{\nu}_h^+ - \rho\tilde{\nu}_h^-\right)\right] \\ \tilde{u}_h &= \left(u_h n_x + v_h n_y\right)\end{aligned} \tag{2.6.3}$$

$$\mathcal{H}_c^b = \left(\tilde{u}_h\rho\tilde{\nu}_h\right)^b \tag{2.6.4}$$

$\mathcal{H}_c$ is an upwind numerical flux function for a scalar convection equation and $\mathcal{H}_c^b$ is a boundary numerical flux function for a scalar convection equation. The numerical flux for the turbulence model equation diffusion term is taken as:

$$\begin{aligned}\left(\mathcal{H}_v\right)^+ = &\left[\frac{1}{2}\left(\mu^+ + \mu^- + \left(\rho\tilde{\nu}^+ + \rho\tilde{\nu}^-\right)\left(1 + cb_2\right)\right) - \rho\tilde{\nu}^+ cb_2\right] \times \\ &\left(\frac{\left(\tilde{\nu}^- - \tilde{\nu}^+\right)\left(\Delta x^\pm\right)}{\left(l^\pm\right)^2} n_x + \frac{\left(\tilde{\nu}^- - \tilde{\nu}^+\right)\left(\Delta y^\pm\right)}{\left(l^\pm\right)^2} n_y\right)\end{aligned} \tag{2.6.5}$$

with

$$\begin{aligned}\Delta x^\pm &= x^- - x^+ \\ \Delta y^\pm &= y^- - y^+ \\ l^\pm &= \sqrt{\Delta x^{\pm 2} + \Delta y^{\pm 2}}\end{aligned} \tag{2.6.6}$$

$$\begin{aligned}\left(\mathcal{H}_v\right)^- = &\left[\frac{1}{2}\left(\mu^+ + \mu^- + \left(\rho\tilde{\nu}^+ + \rho\tilde{\nu}^-\right)\left(1 + cb_2\right)\right) - \rho\tilde{\nu}^- cb_2\right] \times \\ &\left(\frac{\left(\tilde{\nu}^- - \tilde{\nu}^+\right)\left(\Delta x^\pm\right)}{\left(l^\pm\right)^2} n_x + \frac{\left(\tilde{\nu}^- - \tilde{\nu}^+\right)\left(\Delta y^\pm\right)}{\left(l^\pm\right)^2} n_y\right)\end{aligned} \tag{2.6.7}$$

$\mathcal{H}_v^\pm$ are thin layer approximation viscous flux functions and are employed so that turbulence model gradients are not required. The same formulation of the turbulence model viscous flux is employed in [5]. Similarly, the boundary numerical flux is taken as:

$$\begin{aligned}\mathcal{H}_v^b = &\left[\frac{1}{2}\left(\mu^+ + \mu^b + \left(\rho\tilde{\nu}^+ + \rho\tilde{\nu}^b\right)\left(1 + cb_2\right)\right) - \rho\tilde{\nu}^+ cb_2\right] \times \\ &\left(\frac{\left(\tilde{\nu}^b - \tilde{\nu}^+\right)\left(\Delta x^\pm\right)}{\left(l^\pm\right)^2} n_x + \frac{\left(\tilde{\nu}^b - \tilde{\nu}^+\right)\left(\Delta y^\pm\right)}{\left(l^\pm\right)^2} n_y\right)\end{aligned} \tag{2.6.8}$$

H_v^b is the boundary thin layer viscous numerical flux. These flux formulations treat the SA turbulence model equation decoupled from the mean flow RANS equations.

The source term discretization is straight forward since it does not depend on the gradient of the turbulence model variable. While the convective discretization can be used with any numerical method, the diffusion discretization is specifically tailored for a first-order finite-volume method. Additionally the form of the continuous diffusion term used in this work is only advantageous for finite-volume and similar discretizations. This form of the diffusion term is not advantageous for DG discretizations because the manipulation used to obtain the non-conservative form of the diffusion term is undone upon integrating by parts to obtain a weak form DG discretization.

When employing this turbulence model discretization method, the turbulence model and the RANS equations discretizations occupy different function spaces, and a method by which to obtain $\mu^{\pm}$, $u_h^{\pm}$ etc. for use in the turbulence model discretization is required. Due to the first-order accurate turbulence model discretization, an appropriate quadrature rule consisting of a single Gauss point is chosen for the integrations. For volume integrals the Gauss point is at the cell-center and for edge integrals the Gauss point is at the edge mid-point. Mean flow quantities are projected from their modal representations to these quadrature points directly, via equation (2.2.11). This amounts to a "reconstruction" of the mean flow variables but not the turbulence model variables since the turbulence model variables are constant over the cell in question. The combination of this first-order finite-volume discretization for the turbulence model equation combined with a DG discretization of the mean flow equations is denoted as a hybrid discretization.

2.7 Artificial Viscosity Formulation

This work makes extensive use of artificial diffusion for shock wave capturing. The governing parameter for artificial diffusion is the artificial viscosity $\hat{\epsilon}$ in equation (2.1.10). Many researchers have developed artificial viscosity methods and this work has tested and implemented those of references [28, 34, 37, 38]. After much experimentation, two methods have been observed to be the most robust, these are the methods of references [34, 37]. The method of reference [38] was abandoned due to an extended numerical stencil and a dual

inconsistency(see Chapter 3) when using a DG method. The method of reference [28] was abandoned due to robustness problems at high-order. This work makes use of both the piecewise constant method of reference [34] and the PDE-based artificial viscosity of reference [37] to control the values of $\hat{\epsilon}$.

2.7.1 Piecewise Constant Artificial Viscosity

The piecewise constant artificial viscosity formulation is taken from reference [34]. The value of the artificial viscosity coefficient $\tilde{\epsilon}$ in each cell k is given as

$$\begin{aligned} \tilde{\epsilon}_k &= \begin{cases} 0 & s_k \leq s_0 - \kappa \\ \frac{1}{2}\epsilon_0 \left(1 + \sin\left(\frac{1}{2}\frac{\pi(s_k - s_0)}{\kappa}\right)\right) & s_0 - \kappa \leq s_k \leq s_0 + \kappa \\ \epsilon_0 & s_k \geq s_0 + \kappa \end{cases} \\ s_0 &= -4 \log\left(c_{s_0} p\right) \end{aligned} \tag{2.7.1}$$

where c_{s_0}, κ and ϵ_0 are user defined coefficients. The coefficient ϵ_0 controls the magnitude of the artificial viscosity coefficient $\tilde{\epsilon}$ and the coefficients κ and c_{s_0} control the minimum value of s_k that yields non-zero artificial viscosity. s_k for the cell k is given by the resolution indicator

$$s_k = log_{10}\left(\frac{\int_{\Omega_k}\left(P_h - \tilde{P}_h\right)^2 d\Omega_k}{\int_{\Omega_k} P_h^2}\right) \tag{2.7.2}$$

where P_h is the pressure computed from equation (2.1.7) using the number of modes in a discretization of order p and $\tilde{P}_h$ is computed using the number of modes in a discretization of order $p-1$. The function controlling $\tilde{\epsilon}_k$ is essentially a smooth min-max function, which limits the viscosity in the cell between two bounds in a smooth differentiable manner. Notice that the minimum and maximum values are determined based on the width κ of the smooth function that goes between the minimum and maximum values. The final artificial viscosity is given as

$$\begin{aligned} \hat{\epsilon} &= \lambda_{max}\frac{\bar{h}}{p}\tilde{\epsilon}_k \\ \lambda_{max} &= |u| + |v| + a \\ \bar{h} &= \frac{1}{2}\left(h_x + h_y\right) \end{aligned} \tag{2.7.3}$$

where λ_{max} is an estimate of the maximum eigenvalue of the Euler equations and h_x and h_y are anisotropic mesh size metrics, which are given in Section 2.7.3.

While the basic formulation is taken from reference [34], significant changes to the artificial diffusion flux given in equation (2.1.10) and to the definition of $\hat{\epsilon}_k$ have been made relative to those in reference [34]. Most notable is the addition of the λ_{max} scaling term, which allows for the method to automatically adjust the applied viscosity based on local flow conditions. Additionally, the use of anisotropic mesh size metrics allow for robust behavior on anisotropic viscous meshes. In contrast, reference [34] sets $\hat{\epsilon} = \tilde{\epsilon}_k \bar{h}/p$ and $h_x = h_y = \bar{h}$, which results in a significantly less robust method. The present method in equation (2.7.3) is capable of capturing shock waves within one element. Note that $\hat{\epsilon}$ is constant on an element k.

2.7.2 PDE-Based Artificial Viscosity

The piecewise constant artificially viscosity method can fail due to large jumps in $\hat{\epsilon}$ between elements for strong high Mach number shock waves, hence the PDE-based method of reference [37] was also implemented. The PDE-based artificial viscosity solves a nonlinear Poisson equation for the artificial viscosity coefficient ϵ, which is appended to the Navier-Stokes equations. Solving this diffusion equation results in a smooth artificial viscosity distribution throughout the mesh. This equation is discretized to the same order as the mean flow equations using a Symmetric Interior Penalty (SIP) method. This new system of equations is solved in a fully coupled manner. The diffusion equation governing the artificial

viscosity for two dimensional flow is given as:

$$\begin{aligned}
&\frac{\partial \epsilon}{\partial t} = \nabla \cdot \left(\left[\frac{\eta}{\tau} \right] \nabla \epsilon \right) + \frac{1}{\tau} \left(\frac{\bar{h}}{p} \lambda_{max} \tilde{s}_k - \epsilon \right) \\
&\tau = \frac{h_{min}}{c_1 p \lambda_{max}} \\
&\left[\frac{\eta}{\tau} \right] = \frac{c_1 c_2 p \lambda_{max}}{h_{min}} \begin{bmatrix} h_x^2 & 0 \\ 0 & h_y^2 \end{bmatrix} \\
&\lambda_{max} = |u| + |v| + a \\
&\bar{h} = \frac{1}{2} \left(h_x + h_y \right) \\
&h_{min} = min \left(h_x, h_y \right) \\
&c_1 = 3 \\
&c_1 c_2 = 15
\end{aligned} \tag{2.7.4}$$

In this case the source term $\tilde{s}_k$ in equation (2.7.4) is given by

$$\begin{aligned}
&\tilde{s}_k = \begin{cases} 0 & s_k \leq s_0 - \Delta\kappa \\ \frac{1}{2}\epsilon_0 \left(1 + \sin\left(\frac{1}{2} \frac{\pi (s_k - s_0)}{2\Delta\kappa} \right) \right) & s_0 - \Delta\kappa \leq s_k \leq s_0 + \Delta\kappa \\ \epsilon_0 & s_k \geq s_0 + \kappa \end{cases} \\
&s_0 = -\left(\kappa + c_{s_0} \log_{10}(p) \right) \\
&\Delta\kappa = \frac{1}{2}
\end{aligned} \tag{2.7.5}$$

with ϵ_0, c_{s_0}, and κ are user defined coefficients. The coefficient ϵ_0 controls the magnitude of the source term. The coefficients κ and c_{s_0} control minimum value of the shock detector that will trigger artificial viscosity as seen in equation (2.7.5). The value of s_k is given by the jump indicator of reference [67]

$$s_k = log_{10} \left(\frac{1}{|\partial\Omega_k|} \int_{\partial\Omega_k} \left| \frac{[\![P_h]\!] \cdot \vec{n}}{\{P_h\}} \right| ds \right) \tag{2.7.6}$$

This indicator is used to drive source term in equation (2.7.4) because it has proven to be more reliable for very strong shock waves and viscous flows. While using equation (2.7.6) to drive the piecewise constant artificial viscosity method is possible, using this indicator in the piecewise constant setting extends the numerical stencil beyond the nearest neighbors and

hence violates one of the constraints on shock capturing methods. The artificial viscosity coefficient, which is the solution to equation (2.7.4) is limited to give $\hat{\epsilon}$. The final expression for $\hat{\epsilon}$, which is a limited version of ϵ, is given as:

$$\begin{aligned}
\hat{\epsilon}_k &= \begin{cases} 0 & \epsilon \leq \hat{\epsilon}_{low} \\ \frac{1}{2}\left(1+\sin\left(\pi\left[\frac{\epsilon-\hat{\epsilon}_{low}}{\hat{\epsilon}_{hi}-\hat{\epsilon}_{low}}-\frac{1}{2}\right]\right)\right) & \hat{\epsilon}_{low} \leq \epsilon \leq \hat{\epsilon}_{hi} \\ \hat{\epsilon}_{hi} & \epsilon \geq \hat{\epsilon}_{hi} \end{cases} \\
\hat{\epsilon}_{low} &= .01\lambda_{max}\frac{\bar{h}}{p} \\
\hat{\epsilon}_{hi} &= \lambda_{max}\frac{\bar{h}}{p}
\end{aligned} \tag{2.7.7}$$

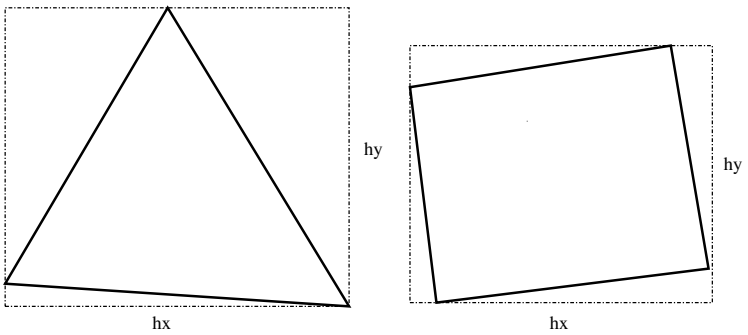
which acts as a limiter to bound the viscosity between minimum and maximum threshold values given by $\hat{\epsilon}_{low}$ and $\hat{\epsilon}_{hi}$ respectively. One should also note that $\epsilon = 0$ is an exact solution to the artificial viscosity diffusion equation if the source term vanishes. Thus if p is taken to infinity or the mesh size to zero the artificial viscosity will vanish, which implies consistency with the original governing PDEs.

2.7.3 Mesh Metrics

The artificial viscosity method in this work relies on the computation of mesh size metrics. The mesh size metric is computed based on finding the minimum Cartesian quadrilateral that encloses the cell. The size is then based on the sizes of the sides of the enclosing quadrilateral. Figure 2.11(a) and Figure 2.11(b) show graphical representations of the mesh sizes for triangles and quadrilaterals respectively. The mesh metrics are computed by finding the maximum distance between the element nodes in each Cartesian i.e. (x, y) direction separately for a given cell.

2.7.4 Artificial Viscosity Method Comparison

While both piecewise constant and PDE-based artificial viscosity methods are implemented within the DG solver, the PDE-based method has become the method of choice because it has demonstrated superior robustness when compared to the piecewise constant method.



(a) Triangle (b) Quadrilateral

Figure 2.11: Mesh size metrics for triangles and quadrilaterals.

In this work, robustness is measured in several ways. First of all, an artificial viscosity method is considered robust if it can adequately remove the Gibbs phenomena, caused by using high-order discretizations to approximate shock waves, over a large range of Mach numbers. For aerodynamic problems this range encompasses transonic to hypersonic flow regimes. Secondly, the method must avoid applying diffusion to smooth phenomena such as boundary layers (resulting in adverse accuracy implications), while still applying adequate diffusion to shock waves, even on coarse grids. In references [47] and [68] piecewise constant artificial viscosity was used in conjunction with *hp*-adaptation for inviscid transonic flow and viscous supersonic flow. Attempts to apply this method to a hypersonic flow at a Mach number $M_\infty = 6.0$ resulted in robustness problems as the piecewise constant artificial viscosity was unable to simultaneously avoid the boundary layer and adequately smooth the strong shock wave. However as will be shown, the PDE-based artificial viscosity can be applied to very high Mach number flows relatively robustly.

There is a third, somewhat more subtle, though just as critical, measure of robustness that must be considered for artificial viscosity methods in a high-order setting. This measure of robustness is related to setting the adjustable coefficients of the presented artificial viscosity methods. These coefficients control how aggressively the artificial viscosity attempts to smooth the discontinuity. The third measure of robustness is related to assigning the values of the adjustable coefficients as the discretization order p is raised (either adaptively or uniformly), a process known as p-enrichment. A robust artificial viscosity method should

be able to maintain the values of the adjustable coefficients across the p-enrichment range. This is of critical importance when considering grid convergence of functional outputs.

In order to compare the two artificial viscosity methods, the inviscid transonic flow over a NACA0012 airfoil is considered. The flow conditions are a free-stream Mach number $M_\infty = .80$ and an angle of attack $\alpha = 1.25^o$. This flow was computed using both piecewise constant and PDE-based artificial viscosities with $p = 1$ to $p = 4$. The computational mesh for this flow is depicted in Figure 2.12. Figure 2.13(a) through Figure 2.14(b) show the

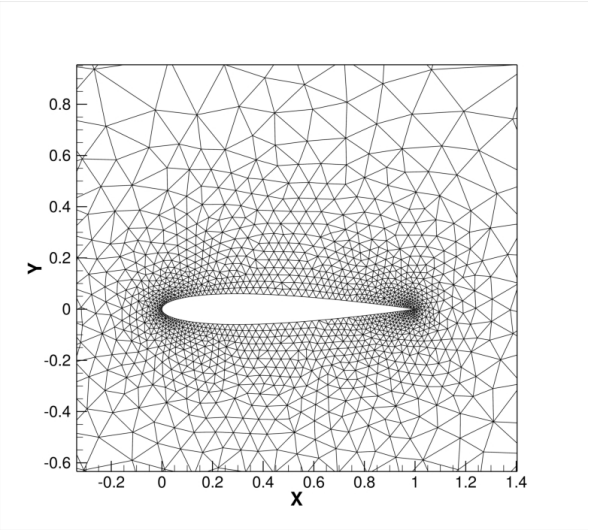
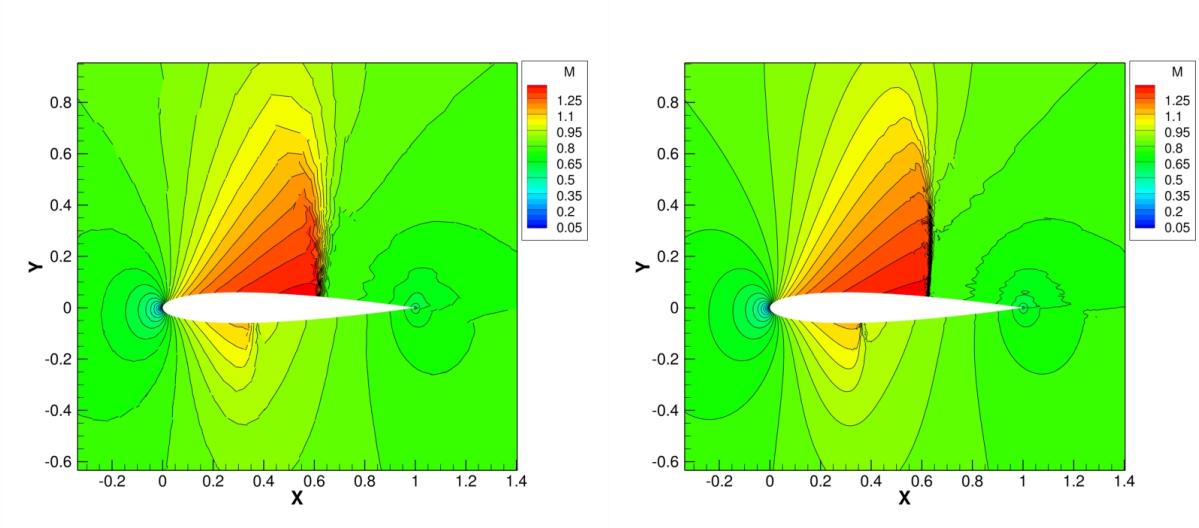


Figure 2.12: Computational mesh used for the transonic flow around a NACA0012 airfoil employing piecewise constant and PDE-based artificial viscosity

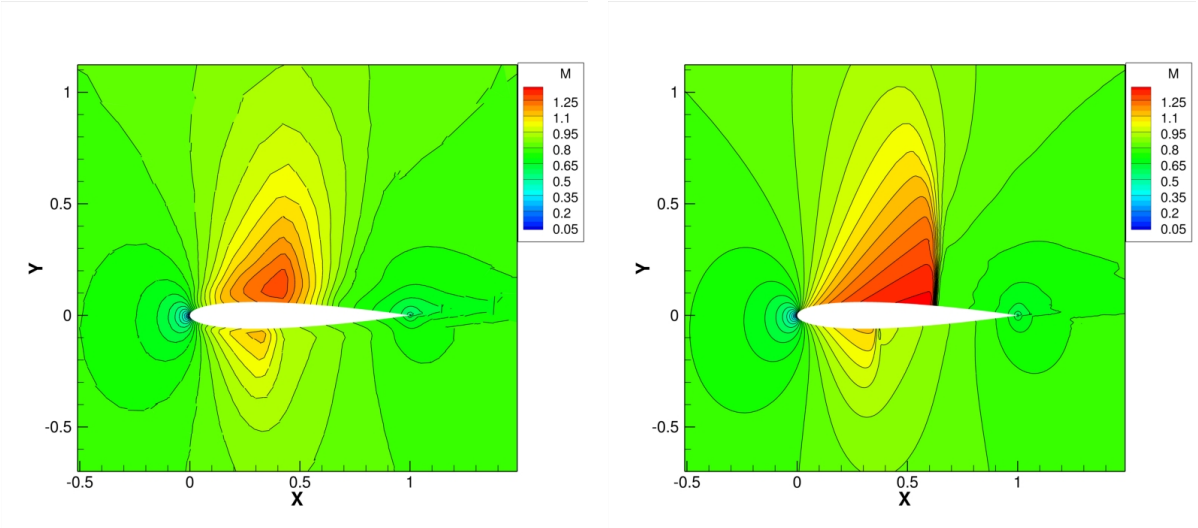
resulting computed Mach number contours for $p = 1$ and $p = 4$ using both artificial viscosity methods. As the discretization order p is increased the shock wave becomes significantly sharper, regardless of the artificial viscosity method employed. However, the PDE-based artificial viscosity spreads the shock wave over a wider area, as is also seen by examining the artificial viscosity contours in Figure 2.15(a) through Figure 2.16(b). Figure 2.17 shows the surface pressure distribution using both artificial viscosity methods at $p = 4$. Clearly the piecewise constant artificial viscosity yields a sharper shock wave. However, the piecewise constant artificial viscosity admits some oscillations at the lower surface shock wave, which is undesirable.

The computed lift coefficient is plotted as a function of the number of degrees of freedom



(a) $p = 1$ Discretization (b) $p = 4$ Discretization

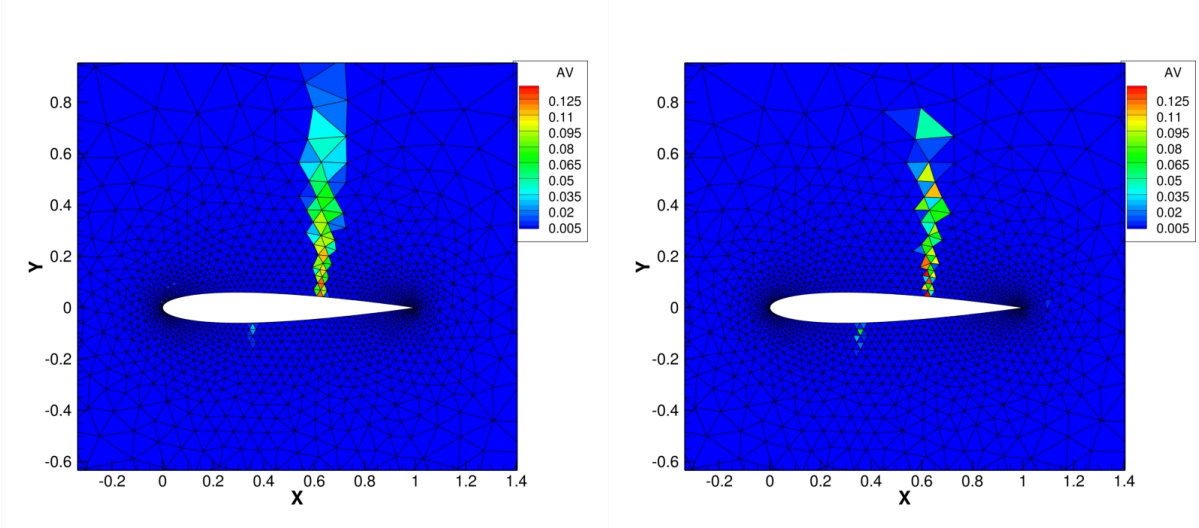
Figure 2.13: Mach number contours for an inviscid flow over a NACA0012 airfoil computed with piecewise constant artificial viscosity using $p = 1$ and $p = 4$.



(a) $p = 1$ (b) $p = 4$

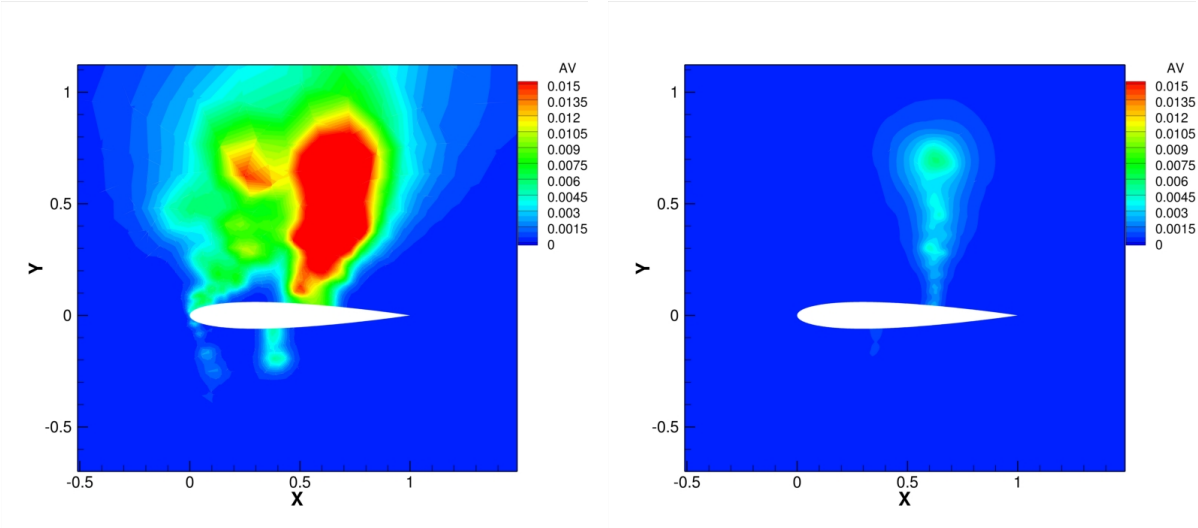
Figure 2.14: Mach number contours for an inviscid flow over a NACA0012 computed with PDE-based artificial viscosity using $p = 1$ and $p = 4$.

(N_{DoF}) is depicted in Figure 2.18. Figure 2.18 clearly demonstrates that the computed lift coefficient produced by the piecewise constant method does not converge towards a fixed value as p-enrichment is performed, as the values change dramatically as p-enrichment is performed. This is due to adjustments in the artificial viscosity coefficients at each refine-



(a) $p = 1$ (b) $p = 4$

Figure 2.15: Artificial viscosity contours for an inviscid flow over a NACA0012 computed with piecewise constant artificial viscosity using discretization orders $p = 1$ and $p = 4$.



(a) $p = 1$ (b) $p = 4$

Figure 2.16: Artificial viscosity contours for an inviscid flow over a NACA0012 computed with PDE-based artificial viscosity using discretization orders $p = 1$ and $p = 4$.

ment level, which was required in order to obtain a fully converged solution. Adjusting the coefficients for the artificial viscosity method has a significant impact on the computed lift coefficient. Note that the $p = 4$ solutions using both methods resolve the shock wave within one element. It is critical that the coefficients remain fixed during refinement procedures.

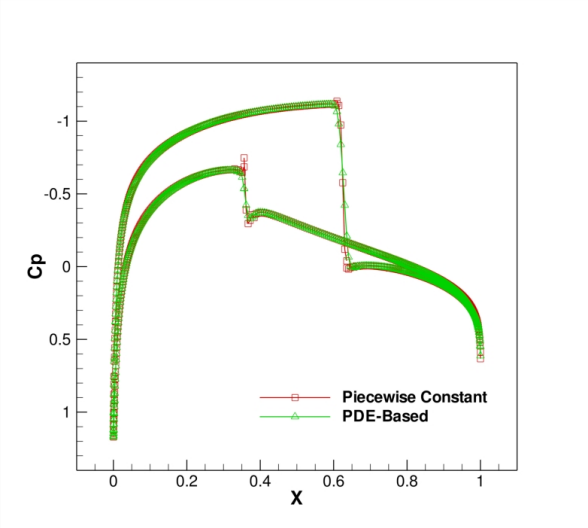


Figure 2.17: Surface pressure coefficient C_p comparison at a discretization order of $p = 4$, using PDE-based and piecewise constant artificial viscosities.

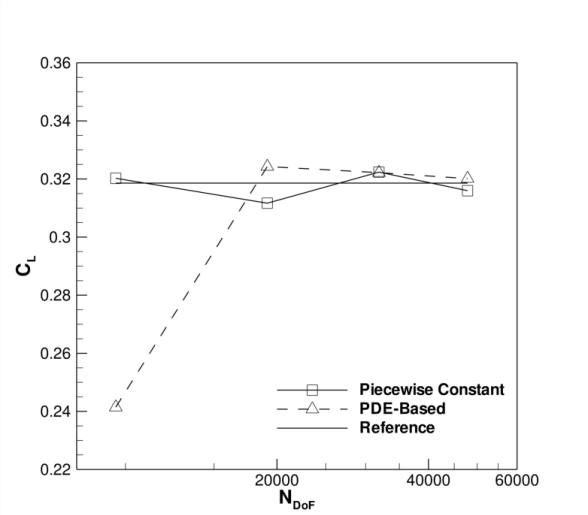


Figure 2.18: Computed lift coefficient(C_L) vs. N_{DoF} for the transonic flow over a NACA0012 airfoil using both piecewise constant and PDE-based artificial viscosities at a discretization order of $p = 4$.

Otherwise one can easily end up in a situation where higher discretization order solutions give nicer "looking" results, while producing no quantitative improvement in output functional accuracy. This is the case in Figure 2.13(a) and Figure 2.13(b) where the solution is more highly refined, while Figure 2.18 shows no improvement in accuracy of the computed lift coefficient. Figure 2.14(a) and Figure 2.14(b) show the computed Mach number contours using the PDE-based artificial viscosity with $p = 1$ and $p = 4$ respectively. While the

PDE-based artificial viscosity has smeared the shock wave over a larger distance (best seen in Figure 2.17) the functional is converging towards a fixed value, as the values change less between refinement levels as the discretization order is increased.

Of the two methods presented, the PDE-based artificial viscosity method has been found to be more robust than the piecewise constant method. This is due to the large amount and smooth distribution of the artificial viscosity produced by the diffusion equation controlling the artificial viscosity. In fact, observations made during the course of this work agree with those of [37, 69] with regard to the added robustness of smooth artificial viscosity distributions. The added robustness is also demonstrated by checking the grid convergence when uniformly raising the polynomial order or p-enrichment of a transonic flow.

As a result of this test, PDE-based artificial viscosity is considered almost exclusively for the remainder of this work. PDE-based artificial viscosity has proven to be significantly more robust by all the presented measures. It has excellent detection and limitation properties and can handle a very large range of Mach numbers, as will be demonstrated. In addition p-enrichment can be performed without adjusting the coefficients of the method. Two of the presented examples will make use of the piecewise constant artificial viscosity formulation in order to make a full assessment of this method.

2.8 Post-Processing

In the previous section, the computed Mach number contour lines are not always smooth and continuous, such as in Figure 2.14(a). These non-smooth contour lines are the result of post-processing artifacts, or rather a lack of post-processing of the solution. Most state-of-the-art CFD solvers store the solution at the element-center or at the nodes in the mesh. Contour lines are drawn by constructing interpolation functions using these uniquely specified data values. However, DG methods store the solution in a more general fashion. For example, this work makes use of modal basis functions, therefore the solution coefficients have no relevance in physical space. In order to visualize the solution, it must be projected from the modal space to physical space, which is accomplished by picking a set of output points (ξ, η)

in the standard element and obtaining the solution at these points via equation (2.2.11). In this work, a set of equally spaced points is specified for each element and these points are used to generate sub-elements for visualization purposes, as seen in Figure 2.19. In order

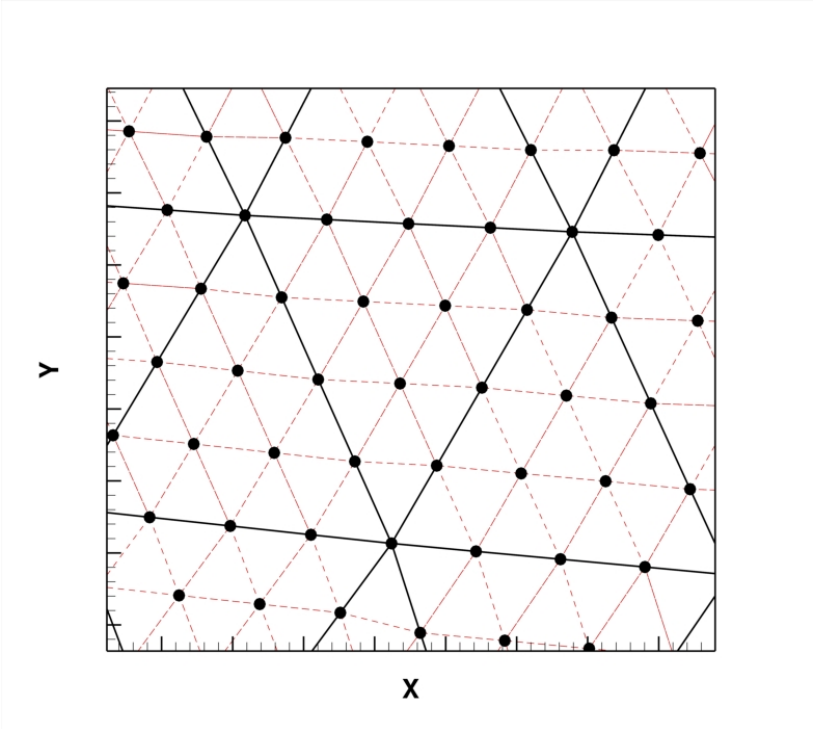


Figure 2.19: Example of output points. The circles are output points, the dashed lines are connections between output points and the solid lines are element boundaries.

to capture the solution over the full element, output points are specified along the element edges, which creates doubly valued solutions along the edges in the mesh(an edge is shared by two elements). The non-continuous behavior of the contours lines is the result of doubly valued solutions at the output points on the element edges. The post-processor is unable to display doubly valued data smoothly. However, this representation of the data is actually a more accurate representation of what the CFD solver produces, since the interpolation used to construct the contour lines cannot generate unique contour line values for data sets that contain doubly valued solutions. Therefore, if a figure contains smooth continuous contour lines such as in Figure 2.14(b), then the data produced by the CFD solver is nearly continuous at the element edges.

2.9 Boundary Conditions

A particularly attractive feature of compact high-order methods such as DG discretizations is the ability to incorporate high-order boundary conditions. However, the boundary conditions must be properly understood and implemented in order to obtain high-order super convergence of the functional outputs. For example, if the approximate Riemann solver, which is employed for the interior surface flux $\mathcal{H}_c$, is used on a slip wall then the functional super convergence will be lost, which is the result of dual inconsistency as discussed in Chapter 3. This section details the various boundary conditions used for the presented test cases. DG discretizations present a challenge for applying boundary conditions due to the presence of artificial diffusion and viscous terms on the boundary. Since gradients are not reconstructed, special boundary conditions are required for these operators such that they reflect the physics of the boundary while at the same time maintain proper mathematical treatment. In particular, the boundary conditions on the artificial diffusion operator are derived since the treatment of this operator at boundaries is often omitted from the literature [28, 34, 36–38, 70, 71].

When considering the mathematical treatment of boundaries one must be mindful of both accuracy and stability. Reference [3] discusses this issue in detail, with the main theme that stable boundary conditions are those that satisfy the direction of information propagation based on the characteristics. In this work, the characteristics of the two-dimensional Euler equations are utilized to derive the boundary conditions. These characteristics are given as:

$$\begin{aligned} \lambda_1 &= \vec{u} \cdot \vec{n} - a \\ \lambda_2 &= \vec{u} \cdot \vec{n} \\ \lambda_3 &= \vec{u} \cdot \vec{n} \\ \lambda_4 &= \vec{u} \cdot \vec{n} + a \end{aligned} \tag{2.9.1}$$

Figure 2.20 depicts these characteristics for a slip wall boundary condition. The characteristics enforce the constraints on how the information must be propagated at the boundaries. An outward(relative to the domain interior) pointing characteristic indicates that informa-

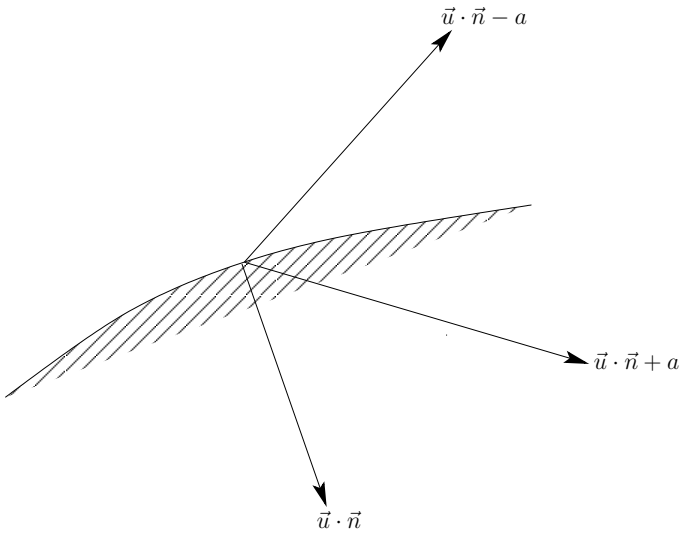


Figure 2.20: Illustration of characteristic directions at the boundary.

tion leaves the domain and hence the boundary state should use information from the interior of the domain, while an inward pointing characteristic implies the opposite and the boundary state should be fixed based on the physical boundary values. One simply counts the number of incoming and outgoing characteristics to obtain the correct number of boundary conditions that must be specified by the physical boundary. The constraint is that one must obey the characteristics and maintain the form of the boundary condition utilized in equation (2.2.4). Furthermore let $()^b$ represent the boundary state and let $()^+$ denote the state inside the element that is on the boundary. The various boundary conditions will be described individually beginning with a slip wall. The goal of this section is to obtain boundary conditions of the form

$$\mathbf{u_h^b}\left(\mathbf{u}_h^+\right) \tag{2.9.2}$$

for each type of boundary condition required for the numerical examples. Here $\mathbf{u}_h^b$ is the boundary state that is computed from $\mathbf{u}_h^+$, which is the solution from the element that lies on the boundary. The state $\mathbf{u}_h^b$ is evaluated at the quadrature points that lie directly on the boundary, which means that the state is defined on the boundary, not in a ghost element that is adjacent to the boundary. The boundary state $\mathbf{u}_h^b$ is then used in computing fluxes on the boundary using $\mathcal{H}_c^b$, $\mathcal{H}_v^b$, and $\mathcal{H}_{ad}^b$. In the description of the boundary conditions the $()_h$ will be dropped to simplify the notation.

2.9.1 Slip Wall

The inviscid slip wall boundary condition is used throughout this work for various applications in both viscous and inviscid flows. In the case of inviscid flows and curved walls this represents the wall boundary condition that the Euler equations must satisfy. However, in the case of viscous flows and walls that are aligned with the coordinate directions this represents a symmetry boundary condition, which is often used so that only half of the geometry needs to be specified. The slip wall specifies that the flow should be everywhere tangent to the surface as shown in Figure 2.21.

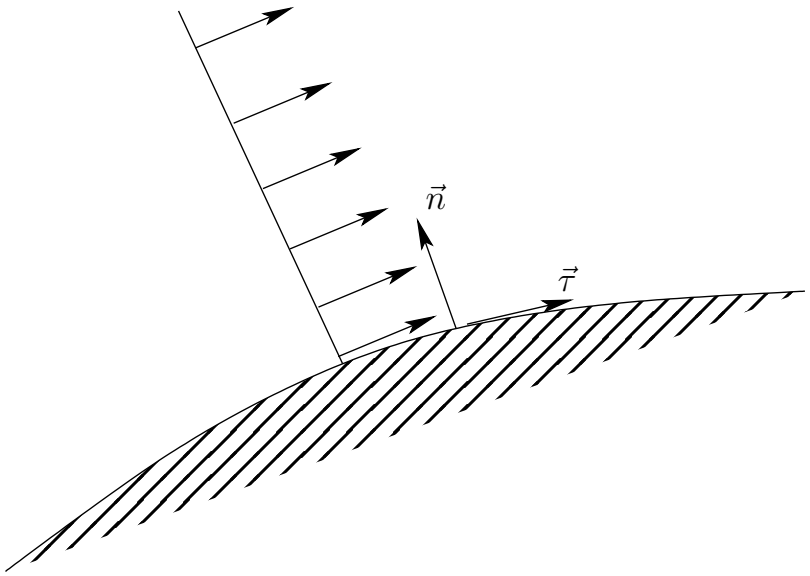


Figure 2.21: Illustration of slip wall boundary condition on velocity.

Flow tangent to the boundary requires that the velocity normal to the wall be zero and the velocity tangent to the wall be preserved,

$$\begin{aligned} \vec{u}^b \cdot \vec{n} &= 0 \\ \vec{u}^b \cdot \vec{\tau} &= \vec{u}^+ \cdot \vec{\tau} \end{aligned} \tag{2.9.3}$$

where $\vec{n}$ is the vector normal to the surface and $\vec{\tau}$ is the vector tangent to the surface as shown in Figure 2.21. For an arbitrary curved edge the coordinates $\vec{x}$ are represented as

$\vec{x}(s)$ where s is a parameterization variable. This gives the tangent vector as

$$\vec{\tau} = \left(\frac{dx}{ds}, \frac{dy}{ds} \right) \equiv (\tau_x, \tau_y)^T \tag{2.9.4}$$

and the normal vector $\vec{n}$ is defined such that:

$$\begin{aligned} \vec{n} \cdot \vec{\tau} &= 0 \\ \vec{n} &\equiv (n_x, n_y)^T \end{aligned} \tag{2.9.5}$$

The tangent vector components can be written in terms of the normal vector components as

$$\vec{\tau} = (n_y, -n_x) \tag{2.9.6}$$

which is used to simplify the notation when the tangent vector is employed.

The slip wall boundary condition results in one inward and one outward pointing characteristic as well as two ambiguous characteristics($\vec{u} \cdot \vec{n} = 0$), which allow for the information to be specified from either the boundary condition or from the element adjacent to the boundary. In this case the velocities are set according to the requirements of equation (2.9.3) and the density and total energy are taken from within the element. Enforcing the requirements on the velocity results in

$$\begin{aligned} u^b n_x + v^b n_y &= 0 \\ u^b n_y - v^b n_x &= u^+ n_y - v^+ n_x \end{aligned} \tag{2.9.7}$$

which are solved to give

$$\begin{aligned} u^b &= u^+ - \left(u^+ n_x + v^+ n_y \right) n_x \\ v^b &= v^+ - \left(u^+ n_x + v^+ n_y \right) n_y \end{aligned} \tag{2.9.8}$$

The resulting boundary state vector $\mathbf{u}^b$ is given as

$$\mathbf{u}^b = \begin{Bmatrix} \rho^+ \\ \rho^+ u^b \\ \rho^+ v^b \\ E_t^+ \end{Bmatrix} \tag{2.9.9}$$

For the SA model equation and PDE-based artificial viscosity the quantities are extrapolated from the interior (as was done with density and total energy).

$$\begin{aligned} (\rho\tilde{\nu})^b &= (\rho\tilde{\nu})^+ \\ \epsilon^b &= \epsilon^+ \end{aligned} \tag{2.9.10}$$

The viscous and artificial diffusion fluxes have yet to be specified. These are perhaps the more difficult boundary conditions to apply because no information about the gradient is given in the boundary condition. However, since this is essentially a symmetry boundary condition one can infer that no gradients are present at the wall. Rather than enforce the boundary condition on the gradient of the state variable, instead the boundary condition is enforced on the fluxes.

$$\begin{aligned} \mathcal{H}_v^b &= 0 \\ \mathcal{H}_{ad}^b &= 0 \end{aligned} \tag{2.9.11}$$

This results in no physical or artificial diffusion on the boundary, which is a stable and accurate boundary condition. Applying appropriate boundary conditions to the artificial diffusion is critical to maintaining a stable discretization.

2.9.2 No-slip Walls

For viscous flows a no slip boundary condition is applied as the wall boundary condition. There are two types of no slip wall boundary conditions, the no-slip adiabatic and the no-slip fixed temperature wall. The adiabatic wall is normally used in aerodynamic simulations while the fixed temperature wall may be used in aerothermodynamic simulations such as hypersonic flow. Both boundary conditions specify zero velocity at the wall. For an adiabatic wall, density is taken from the element and total energy is computed from these values. The Reynolds stresses are also zero at the wall, which implies that the SA working variable $\tilde{\nu}$ is zero at the wall. Treating the artificial viscosity for the PDE-based artificial viscosity as in

the slip-wall boundary condition, the boundary flux vector becomes

$$\begin{aligned}
\mathbf{u}^b &= \left\{ \begin{array}{c} \rho^+ \\ 0 \\ 0 \\ E_t^+ - \frac{1}{2}\rho^+ \left(\left(u^+\right)^2 + \left(v^+\right)^2 \right) \end{array} \right\} \\
\left(\rho\tilde{\nu}\right)^b &= 0 \\
\epsilon^b &= \epsilon^+
\end{aligned} \tag{2.9.12}$$

The viscous and artificial diffusion fluxes are computed using the state gradients from the element on the boundary with the exception of the energy equation viscous flux, which is set to zero.

$$\mathcal{H}_v^b\left(4\right) = 0 \tag{2.9.13}$$

In the case of the fixed temperature wall the state vector is given as

$$\begin{aligned}
\mathbf{u}^b &= \left\{ \begin{array}{c} \rho^b \\ 0 \\ 0 \\ \left(E_t\right)^b \end{array} \right\} \\
\left(\rho\tilde{\nu}\right)^b &= 0 \\
\epsilon^b &= \epsilon^+
\end{aligned} \tag{2.9.14}$$

where

$$\begin{aligned}
P^b &= \left(\gamma - 1\right)\left(E_t^+ - \frac{1}{2}\rho^+ \left(\left(u^+\right)^2 + \left(v^+\right)^2 \right) \right) \\
E^b &= \frac{T_{wall}}{\gamma\left(\gamma - 1\right)} \\
\rho^b &= \frac{P^b}{\gamma\left(\gamma - 1\right)} \\
\left(E_t\right)^b &= \rho^b E^b
\end{aligned} \tag{2.9.15}$$

where T_{wall} is the specified wall temperature. In this case the viscous and artificial diffusion fluxes are computed using this boundary state and the gradient from the element adjacent to the boundary without modification. For the fixed temperature wall the SA model variable is set to zero and the artificial viscosity is extrapolated from the interior.

2.9.3 Characteristic In/Out Flow

Due to the specified form of the boundary condition e.g. $\mathbf{u}^b\,(\mathbf{u}^+)$, the far-field boundary requires special attention. A typical implementation of the far-field boundary condition considers using the Riemann solver at the far-field boundary, which is simple and stable. However, this approach will not satisfy equation (2.9.2). In order to continue to use boundary conditions of the form specified in equation (2.9.2), a characteristics-based far-field boundary condition is employed. In this case, the Riemann invariants of an isentropic inviscid flow are considered, including a scalar transport equation for the turbulence model equations.

In order to find the boundary state $\mathbf{u}^b$, the Riemann invariants must be obtained. The full analysis is too lengthy to be considered in full here. Instead the Riemann invariants will be presented without derivation for two-dimensional flow with a scalar transport equation appended to the system. It will be shown how to derive far-field boundary conditions for sub/supersonic flows using the Riemann invariants. The Riemann invariants are given as:

$$\begin{aligned} w_1 &= \mathcal{S} \\ w_2 &= \vec{u} \cdot \vec{\tau} \\ w_3 &= s \\ w_4 &= \vec{u} \cdot \vec{n} + \frac{2a}{(\gamma - 1)} \equiv R_p \\ w_5 &= \vec{u} \cdot \vec{n} - \frac{2a}{(\gamma - 1)} \equiv R_m \end{aligned} \tag{2.9.16}$$

where

$\mathcal{S} \equiv$ entropy

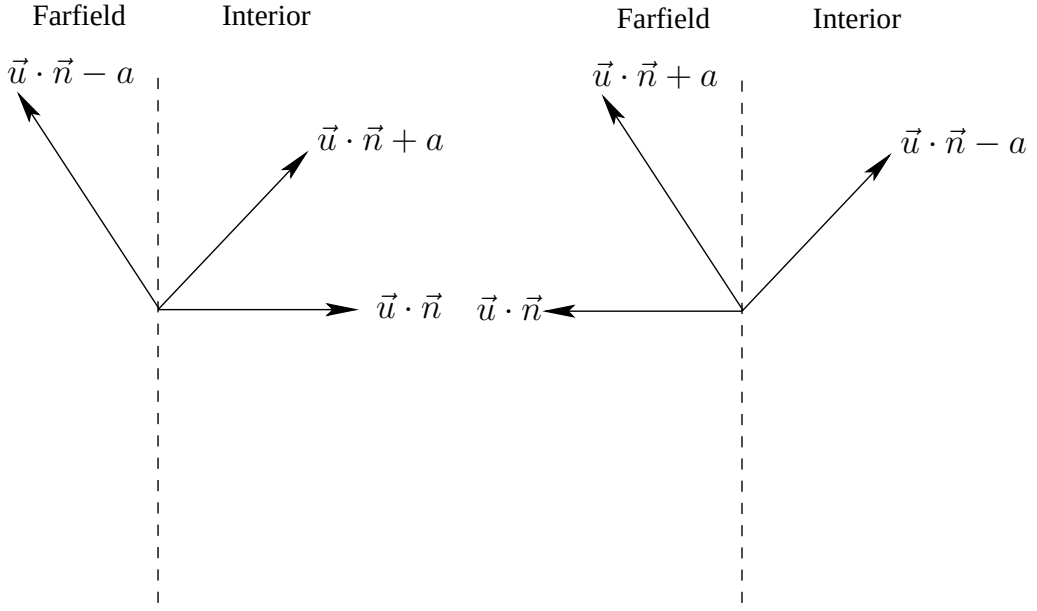
$s \equiv$ scalar

and must remain constant across the boundary. The actual boundary conditions will depend on whether the flow is subsonic or supersonic.

Subsonic Flow

In the case of subsonic flow, the number of characteristics pointing into or out of the domain depends on whether the boundary is an inflow or outflow boundary. The inflow and outflow

boundary situations are depicted pictorially in Figure 2.22(a) and Figure 2.22(b) respectively. In the case of subsonic flow the Riemann invariants are solved to determine the boundary



(a) Inflow (b) Outflow

Figure 2.22: Inflow and Outflow characteristic information propagation directions for subsonic flow.

state. For an inflow boundary (i.e. $\vec{u}\cdot\vec{n}<0$) the Riemann invariants are

$$\begin{cases} \mathcal{S} = \dfrac{P_\infty}{(\rho_\infty)^\gamma} \\ R_p = (\vec{u}\cdot\vec{n})^+ + \dfrac{2a^+}{(\gamma-1)} \\ R_m = (\vec{u}\cdot\vec{n})_\infty - \dfrac{2a_\infty}{(\gamma-1)} \qquad \vec{u}\cdot\vec{n}<0 \\ (\vec{u}\cdot\vec{\tau})^b = -u_\infty n_y + v_\infty n_x \\ s^b = s_\infty \end{cases} \tag{2.9.17}$$

where $()_\infty$ specifies the far-field or free-stream state. Similarly, the outflow (i.e. $\vec{u}\cdot\vec{n}>0$) Riemann invariants are

$$\begin{cases} \mathcal{S} = \dfrac{P^+}{(\rho^+)^\gamma} \\ R_p = (\vec{u}\cdot\vec{n})^+ + \dfrac{2a^+}{(\gamma-1)} \\ R_m = (\vec{u}\cdot\vec{n})_\infty - \dfrac{2a_\infty}{(\gamma-1)} \qquad \vec{u}\cdot\vec{n}>0 \\ (\vec{u}\cdot\vec{\tau})^b = -u^+ n_y + v^+ n_x \\ s^b = s^+ \end{cases} \tag{2.9.18}$$

The Riemann invariants are then used to compute the boundary state. First of all, the boundary normal velocity and sound speed are given as

$$\begin{aligned} a^b &= \frac{(\gamma-1)}{4\left(R_p - R_m\right)} \\ (\vec{u}\cdot\vec{n})^b &= \frac{1}{2}\left(R_p + R_m\right) \end{aligned} \tag{2.9.19}$$

which are intermediate quantities. The density is computed as

$$\rho^b = \left(\frac{\mathcal{S}\gamma}{\left(a^b\right)^2}\right)^{\frac{1}{(\gamma-1)}} \tag{2.9.20}$$

and the velocities are computed as

$$\begin{aligned} u^b &= n_x\left(\vec{u}\cdot\vec{n}\right)^b - n_y\left(\vec{u}\cdot\vec{\tau}\right)^b \\ v^b &= n_y\left(\vec{u}\cdot\vec{n}\right)^b + n_x\left(\vec{u}\cdot\vec{\tau}\right)^b \end{aligned} \tag{2.9.21}$$

Finally the total energy is given by:

$$\begin{aligned} P^b &= \mathcal{S}\left(\rho^b\right)^\gamma \\ \left(E_t\right)^b &= \frac{P^b}{(\gamma-1)} + \frac{1}{2}\rho^b\left(\left(u^b\right)^2 + \left(v^b\right)^2\right) \end{aligned} \tag{2.9.22}$$

Inserting the expressions for velocity, density, and energy into the state vector gives

$$\mathbf{u}^b = \begin{Bmatrix} \rho^b \\ \rho^b u^b \\ \rho^b v^b \\ \left(E_t\right)^b \end{Bmatrix} \tag{2.9.23}$$

$$\left(\rho\tilde{\nu}\right)^b = \rho^b s^b$$

where in this case the scalar s_b equals the turbulence model variable $\tilde{\nu}_b$. Viscous and artificial diffusion fluxes are set to zero at the outer boundary.

$$\begin{aligned} \mathcal{H}_v^b &= 0 \\ \mathcal{H}_{ad}^b &= 0 \end{aligned} \tag{2.9.24}$$

Supersonic Flow

The case of supersonic flow is quite simple since all the characteristics point in a single direction as shown in Figure 2.23(a) and Figure 2.23(b). For supersonic inflow all information comes from the far-field conditions and for supersonic outflow all information comes from the domain interior. Mathematically the supersonic inflow state vector is given by:

$$\begin{aligned} \mathbf{u}^b &= \left\{ \begin{array}{c} \rho_\infty \\ \rho_\infty u_\infty \\ \rho_\infty v_\infty \\ (E_t)_\infty \end{array} \right\} \\ (\rho\tilde{\nu})^b &= \rho_\infty s_\infty \end{aligned} \tag{2.9.25}$$

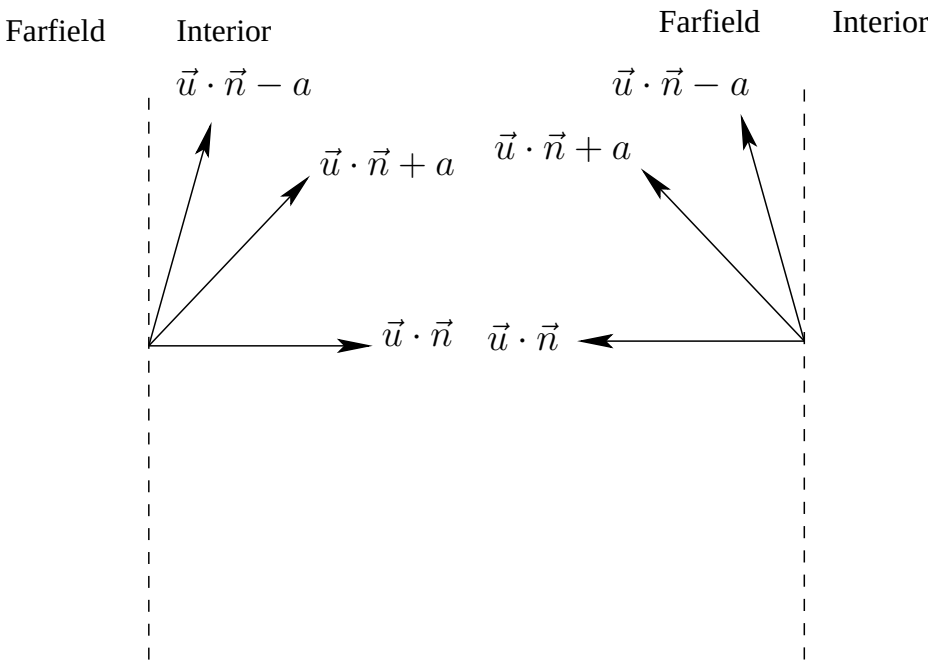
and the supersonic outflow case is given by

$$\begin{aligned} \mathbf{u}^b &= \left\{ \begin{array}{c} \rho^+ \\ \rho^+ u^+ \\ \rho^+ v^+ \\ (E_t)^+ \end{array} \right\} \\ (\rho\tilde{\nu})^b &= \rho^+ s^+ \end{aligned} \tag{2.9.26}$$

The artificial viscosity transport equation far-field boundary condition is different from that of the Navier-Stokes equations. Following reference [37], the far-field boundary condition of the artificial viscosity PDE is a Robin boundary condition applied such that

$$\frac{\partial \epsilon}{\partial \vec{n}} = \frac{\epsilon_\infty - \epsilon^+}{L} \tag{2.9.27}$$

where in this work $L = 10\sqrt{\left(\vec{h} \cdot \vec{n}\right)^2}$, $\vec{h} = (h_x, h_y)$ is the mesh size metric in Section 2.7.3,



(a) Inflow (b) Outflow

Figure 2.23: Inflow and Outflow characteristic information propagation directions for supersonic flow.

and $\epsilon_\infty = 0$. One such expression that satisfies this requirement is

$$\begin{aligned} \frac{\partial \epsilon}{\partial x}^b &= -\frac{\epsilon^+}{10\sqrt{\left(\vec{h}\cdot\vec{n}\right)^2}} n_x \\ \frac{\partial \epsilon}{\partial y}^b &= -\frac{\epsilon^+}{10\sqrt{\left(\vec{h}\cdot\vec{n}\right)^2}} n_y \end{aligned} \tag{2.9.28}$$

Then the flux for the artificial viscosity PDE on the far-field boundary is computed as usual but the symmetry and penalty terms are set to zero because this is not a Dirichlet boundary condition.

$$H^b_{ad}\left(f_{pde-av}\right) = \frac{\partial \epsilon}{\partial x}^b n_x + \frac{\partial \epsilon}{\partial y}^b n_y \tag{2.9.29}$$

where f_{pde-av} is the equation number corresponding to the artificial viscosity PDE.

Chapter 3

Dual Consistency of Discontinuous Galerkin Discretizations

When considering DG discretizations, dual (or adjoint) consistency is a key property that must be fulfilled by the discretization in order to obtain optimal error convergence properties [49, 72]. Dual consistency requires that the adjoint of the discretized equations satisfies the continuous adjoint equations. Through the dual consistency property one attains the so-called order doubling of functional error, where the functional error behaves asymptotically as $\mathcal{O}\left(h^{2p}\right)$ rather than $\mathcal{O}\left(h^{p}\right)$. For two dimensional problems, $h = \sqrt{N_{DoF}}$ where N_{DoF} is to the total number of degrees of freedom in the mesh. Dual consistency of the Euler and Navier-Stokes equations has been studied in references [49, 56]. References [20, 72] have studied the dual consistency of the SA turbulence model source terms and reference [71] has studied the dual consistency of "Bassi-Rebay 2" [73] discretizations of the artificial diffusion terms.

This work also considers the dual consistency issue, in particular for non-linear Poisson type diffusion terms such as those encountered in turbulence modeling and artificial diffusion schemes. The dual consistency of the proposed SIP discretization is analyzed and remedies for dual inconsistent discretizations are presented when possible. Additionally, the dual consistency of the Navier-Stokes discretization is studied via numerical experimentation to ensure that the implementation has been performed correctly. The proof of dual consistency

for the Navier-Stokes discretization is detailed in reference [49].

This section describes the analysis of dual consistency of the Symmetric Interior Penalty (SIP) discretization for the artificial diffusion operators in Section 2.1. The section very closely follows the analysis framework laid out in reference [49]. A non-linear Poisson equation is used as a model for this analysis, since this equation mimics both the artificial diffusion terms of the Navier-Stokes equations and the diffusion operator of the PDE-based artificial viscosity. This equation also mimics the diffusion and source terms of the SA turbulence model equation, which enables one to draw conclusions regarding the dual consistency of the turbulence model discretization.

In addition to analyzing the dual consistency of a non-linear Poisson equation, the dual consistency of the Euler equations is also established. The analysis demonstrates the importance of boundary condition treatment on dual consistency and illustrates the procedure employed for the dual consistency analysis of boundary conditions. The inviscid wall boundary condition specified in Chapter 2 is analyzed in order to justify the mathematical form of the boundary conditions in equation (2.9.2).

3.1 The Adjoint of Non-linear Operators

Since the dual or adjoint problem is inherently linear the dual problem of a non-linear operator must be defined in terms of an appropriate linearization. Consider a general Fréchet differentiable non-linear operator

$$Nu = 0 \quad \text{in } \Omega, \qquad Bu = 0 \quad \text{on } \Gamma \tag{3.1.1}$$

on the domain Ω with the boundary of Ω denoted as Γ. The problem defined in equation (3.1.1) is known as the primal problem. Let $J(u)$ be a non-linear functional given as

$$J(u) = \int_{\Omega} j_{\Omega}(u)\, d\Omega + \int_{\Gamma} j_{\Gamma}(Cu)\, ds \tag{3.1.2}$$

where Cu is a non-linear operator that takes the solution u and generates derived products for the output functional. For example, computing heat flux from temperature and thermal

conductivity is an example of using Cu to derive a product from the solution u. Let the Fréchet derivative about some point be denoted as,

$$\left(\frac{\partial F}{\partial u}\right)_{[u]}(w) = F'_{[u]}(w) \tag{3.1.3}$$

where the term in square brackets is the state about which the derivative is taken and w is the linear variation of u. The linearized functional is given by

$$J'_{[u]}(w) = \int_{\Omega} j_{\Omega}{}'_{[u]}(w)\, d\Omega + \int_{\Gamma} j_{\Gamma}{}'_{[Cu]}(w)\, ds \equiv J_{\Omega}{}'_{[u]}(w) + J_{\Gamma}{}'_{[Cu]}(w) \tag{3.1.4}$$

and the linearized non-linear operator as

$$N'_{[u]}(w) = 0 \quad \forall \vec{x} \in \Omega, \qquad B'_{[u]}(w) = 0 \quad \forall \vec{x} \in \Gamma \tag{3.1.5}$$

Using the linearized functional $J'_{[u]}(w)$ and operator $N'_{[u]}(w)$, the adjoint operator is defined as the operator $(N^*)'_{[u]}(\psi)$ that satisfies the following duality identity.

$$\begin{aligned} &\left\langle N'_{[u]}(w), \psi \right\rangle_{\Omega} + \left\langle B'_{[u]}(w), (C^*)'_{[u]}(\psi) \right\rangle_{\Gamma} = \\ &\left\langle w, (N^*)'_{[u]}(\psi) \right\rangle_{\Omega} + \left\langle C'_{[u]}(w), (B^*)'_{[u]}(\psi) \right\rangle_{\Gamma} = \\ &\left\langle w, j_{\Omega}{}'_{[u]} \right\rangle_{\Omega} + \left\langle w, j_{\Gamma}{}'_{[Cu]} \right\rangle_{\Gamma} \end{aligned} \tag{3.1.6}$$

where ψ is the adjoint variable. $\langle \cdot, \cdot \rangle_{\Omega}$ is an inner product defined on the domain Ω and $\langle \cdot, \cdot \rangle_{\Gamma}$ is an inner product defined on the boundary Γ of the domain. Note that $\left\langle w, j_{\Omega}{}'_{[u]} \right\rangle_{\Omega} + \left\langle w, j_{\Gamma}{}'_{[Cu]} \right\rangle_{\Gamma}$ is equivalent to $J'_{[u]}(w)$. In this identity B is the boundary operator and B^* is the adjoint boundary operator. This duality identity is not arbitrary, in fact the identity is chosen so that the operator controlling the linearized solution w and linearized functional $J'_{[u]}(w)$ satisfy the same duality statement, as seen in equation (3.1.6). This defines the adjoint which can also be thought of as the sensitivity of the functional with respect to the non-linear operator Nu. The adjoint is the variable ψ that satisfies the following equation

$$(N^*)'_{[u]}(\psi) = j_{\Omega}{}'_{[u]} \quad \vec{x} \in \Omega, \qquad (B^*)'_{[u]}(\psi) = j_{\Gamma}{}'_{[Cu]} \quad \vec{x} \in \Gamma \tag{3.1.7}$$

which is the so-called continuous adjoint equation. The continuous adjoint equation is used in dual consistency analysis. Furthermore, an adjoint variable is only defined for a particular output functional $J(u)$.

3.2 Definition of Dual Consistency

Dual consistency is the property that the continuous adjoint solution satisfies the discrete adjoint equation where the discrete adjoint equation is derived from the discretization primal equation and not by directly discretizing the continuous adjoint. The definition of dual consistency is such that the discrete adjoint equation derived from the the discrete primal problem does represent a discretization of the continuous adjoint equation. Obtaining a discrete adjoint equation via this approach is shown below.

In order to define dual consistency one must first introduce a discretization of the continuous problem Nu. Consider a discretization of Nu given by:
find $u_h \in \mathcal{V}_h$ such that

$$\mathcal{N}(u_h, v_h) = 0 \quad \forall v_h \in \mathcal{V}_h \tag{3.2.1}$$

where $\mathcal{N}(u_h, v_h)$ is a semi-linear form, which is linear in the second argument. The discretization is said to be consistent if

$$\mathcal{N}(u, v_h) = 0 \quad \forall v_h \in \mathcal{V}_h \tag{3.2.2}$$

where u is the exact solution satisfying equation (3.1.1), which is the same as the mesh size h going zero($h \to 0$). Given the linearization of $\mathcal{N}(u_h, v_h)$, denoted as $\mathcal{N}'_{[u_h]}(w_h, v_h)$, the discrete adjoint problem is given as:
find $\psi_h \in \mathcal{V}_h$ such that

$$\mathcal{N}'_{[u_h]}(w_h, \psi_h) = J'_{[u_h]}(w_h) \quad \forall w_h \in \mathcal{V}_h \tag{3.2.3}$$

Analogous to a consistent discretization, the statement of equation (3.2.2), a dual consistent discretization satisfies

$$\mathcal{N}'_{[u]}(w_h, \psi) = J'_{[u]}(w_h) \quad \forall w_h \in \mathcal{V}_h \tag{3.2.4}$$

where ψ is the exact solution of the continuous adjoint problem in equation (3.1.7). Armed with this definition of consistency and dual consistency, a simple analysis is carried out in order to determine the dual consistency properties of the discretization used in this work.

3.3 Importance of Dual Consistency

While dual consistency is important to primal problem accuracy as shown in reference [74]. The effect of dual consistency on functional accuracy is just as important. The ability to attain functional error super-convergence($\mathcal{O}\left(h^{2p}\right)$) is a beneficial property of high-order methods, which should be preserved by a high-order discretization.

To illustrate the importance of dual consistency a linear functional is considered in order to make the argument formality easier to work with. A similar analysis holds for non-linear functionals but requires more formality, which only serves to complicate the conclusion. Consider a weak bi-linear form $\mathcal{L}\left(u,v\right)$, which defines a linear problem.

$$\mathcal{L}\left(u,v\right) = \left\langle f,v\right\rangle_{\Omega} \quad \forall v \in \mathcal{V} \tag{3.3.1}$$

The corresponding dual problem which defines an output of interest (linear analogue of equation (3.2.4)) is given by:

$$\mathcal{L}\left(u,\psi\right) = \left\langle j,u\right\rangle_{\Omega} = J\left(u\right) \quad \forall u \in \mathcal{V} \tag{3.3.2}$$

The discrete problem is find $u_h \in \mathcal{V}_h$ such that

$$\mathcal{L}_h\left(u_h,v_h\right) = \left\langle f,v_h\right\rangle_{\Omega} \quad \forall v_h \in \mathcal{V}_h \tag{3.3.3}$$

By definition, the discrete dual problem for $J\left(u_h\right)$ is find $\psi_h \in \mathcal{V}_h$

$$\mathcal{L}_h\left(u_h,\psi_h\right) = \left\langle j,u_h\right\rangle_{\Omega} = J\left(u_h\right) \quad \forall u_h \in \mathcal{V}_h \tag{3.3.4}$$

Consider the error in the discrete functional

$$J\left(u\right) - J\left(u_h\right) = \mathcal{L}\left(u,\psi\right) - \mathcal{L}_h\left(u_h,\psi_h\right) \tag{3.3.5}$$

If one assumes that the continuous solution u and continuous adjoint ψ satisfy the discrete operator $\mathcal{L}_h$ then equation (3.3.5) can be written as

$$\begin{aligned} J\left(u\right) - J\left(u_h\right) &= \mathcal{L}_h\left(u,\psi\right) - \mathcal{L}_h\left(u_h,\psi_h\right) \\ &= \mathcal{L}_h\left(u,\psi\right) - \mathcal{L}_h\left(u_h,\psi\right) + \mathcal{L}_h\left(u_h,\psi\right) - \mathcal{L}_h\left(u_h,\psi_h\right) \end{aligned} \tag{3.3.6}$$

where zero was added in the form of $\mathcal{L}_h(u_h, \psi) - \mathcal{L}_h(u_h, \psi)$. This can be collapsed into

$$J(u) - J(u_h) = \mathcal{L}_h(u - u_h, \psi) + \mathcal{L}_h(u_h, \psi - \psi_h) \tag{3.3.7}$$

Adding zero in the form of $\mathcal{L}_h(u - u_h, \psi_h) - \mathcal{L}_h(u - u_h, \psi_h)$ results in

$$J(u) - J(u_h) = \mathcal{L}_h(u - u_h, \psi - \psi_h) + \mathcal{L}_h(u_h, \psi - \psi_h) + \mathcal{L}_h(u - u_h, \psi_h) \tag{3.3.8}$$

Now if $\mathcal{L}_h$ is consistent then

$$\mathcal{L}_h(u, v_h) = \langle f, v_h \rangle_\Omega \quad \forall v_h \in \mathcal{V}_h \tag{3.3.9}$$

and if $\mathcal{L}_h$ is dual consistent then

$$\mathcal{L}_h(u_h, \psi) = J(u_h) \quad \forall u_h \in \mathcal{V}_h \tag{3.3.10}$$

since $\psi_h \in \mathcal{V}_h$ and $u_h \in \mathcal{V}_h$, then the following is true.

$$\begin{aligned} \mathcal{L}_h(u_h, \psi - \psi_h) &= J(u_h) - J(u_h) = 0 \\ \mathcal{L}_h(u - u_h, \psi_h) &= \langle f, \psi_h \rangle_\Omega - \langle f, \psi_h \rangle_\Omega = 0 \\ J(u) - J(u_h) &= \mathcal{L}_h(u - u_h, \psi - \psi_h) \end{aligned} \tag{3.3.11}$$

From approximation theory the absolute value of the error can be bounded.

$$\begin{aligned} |J(u) - J(u_h)| &\leq C \parallel u - u_h \parallel \parallel \psi - \psi_h \parallel \\ \parallel u - u_h \parallel_{H^1} &\leq C\mathcal{O}(h^p) \\ \parallel \psi - \psi_h \parallel_{H^1} &\leq C\mathcal{O}(h^p) \\ |J(u) - J(u_h)| &\leq c\mathcal{O}(h^p)\mathcal{O}(h^p) \approx C\mathcal{O}\left(h^{2p}\right) \end{aligned} \tag{3.3.12}$$

where $\| \cdot \|_{H^1}$ is a Sobolev norm of order one. It follows that without dual consistency one cannot form the argument used to generate the $\mathcal{O}(h^{2p})$ functional error estimate. This is the reason dual consistency is important. One should also notice that primal consistency plays a role in obtaining the functional error estimate, thus both primal and adjoint consistency are needed to obtain optimal results. However, dual consistency is often the one that is over looked in the literature.

3.4 Dual Consistency of a Non-linear Poisson Equation

One of the strategies to improve solver robustness is the addition of artificial diffusion to the governing equations. As such one of the principal questions pertains to the effect of artificial diffusion on the discretization. The analysis of a non-linear Poisson equation serves as a model for the artificial diffusion used in this work.

As model problem, consider a non-linear Poisson equation

$$\nabla \cdot (\nu (u, \nabla u) \nabla u) + S (u, \nabla u) = 0 \tag{3.4.1}$$

subject to the boundary conditions

$$\begin{aligned} u &= a, \vec{x} \in \Gamma^D \\ \nabla u \cdot \vec{n} &= a_N, \vec{x} \in \Gamma^N \end{aligned} \tag{3.4.2}$$

where the diffusion coefficient ν and source term S are both non-linear functions of the solution and solution gradient. For this problem let the functional be defined as

$$\begin{aligned} J (u) &= \int_\Omega j_\Omega (u) \, d\Omega + \int_{\Gamma^D} j_D (Cu) \, ds + \int_{\Gamma^N} j_N (Cu) \, ds \\ Cu &= \nu (u, \nabla u) \nabla u \cdot \vec{n} \end{aligned} \tag{3.4.3}$$

with the corresponding linearized functional given as

$$\begin{aligned} J'_{[u]} (w) = \int_\Omega j_{\Omega'_{[u]}} (w) \, d\Omega + \int_{\Gamma^D} j_{D'_{[u]}} (\nu \nabla w \cdot \vec{n} + \nu_u w \nabla u \cdot \vec{n} + \nu_{\nabla u} \cdot \nabla w \nabla u \cdot \vec{n}) \, ds + \\ \int_{\Gamma^N} j_{N'_{[u]}} (\nu_u w \nabla u \cdot \vec{n} + \nu_{\nabla u} \cdot \nabla w \nabla u \cdot \vec{n}) \, ds \equiv J_{\Omega'_{[u]}} (w) + J_{\Gamma^D{}'_{[u]}} (w) + J_{\Gamma^N{}'_{[u]}} (w) \end{aligned} \tag{3.4.4}$$

This equation can model both the artificial diffusion flux for any component of the Navier-Stokes equations, by setting $S (u, \nabla u) = 0$ or the PDE controlling the artificial viscosity by including both. Furthermore, this model problem allows one to draw some conclusions about the SA turbulence model, due to the similarity between this equation and the diffusion and source terms of the SA model.

3.4.1 Continuous Adjoint

The derivation of the continuous adjoint begins by taking the Fréchet derivative, about the solution u, of the non-linear problem defined in equation (3.4.1). Dropping the formal

arguments of ν and S for clarity the Fréchet derivative is

$$\begin{aligned} N'_{[u]}(w) \equiv \nu\nabla^2 w + \frac{\partial \nu}{\partial u}_{[u,\nabla u]} \nabla w \cdot \nabla u + \nabla \cdot \left(\frac{\partial \nu}{\partial \nabla u}_{[u,\nabla u]} \cdot \nabla w \nabla u \right) + \\ \left(\frac{\partial S}{\partial u}_{[u,\nabla u]} w + \frac{\partial S}{\partial \nabla u}_{[u,\nabla u]} \cdot \nabla w \right) = \end{aligned} \tag{3.4.5}$$

For simplicity let $\frac{\partial(\cdot)}{\partial u}_{[u,\nabla u]}$ be represented as $(\cdot)_u$ and $\frac{\partial(\cdot)}{\partial \nabla u}_{[u,\nabla u]}$ as $(\cdot)_{\nabla u}$. Employing the duality identity of equation (3.1.6), with the inner product defined as an integral for continuous functions, the continuous adjoint is given as:

$$\begin{aligned} \int \psi \left[\nu\nabla^2 w + \nu_u \nabla w \cdot \nabla u + \nabla \cdot \left(\frac{\partial \nu}{\partial \nabla u}_{[u,\nabla u]} \cdot \nabla w \nabla u \right) \right] d\Omega + \\ \int \psi \left[S_u w + S_{\nabla u} \cdot \nabla w \right] d\Omega = 0 \end{aligned} \tag{3.4.6}$$

Integration by parts twice results in

$$\begin{aligned} \int w \left[\nabla \cdot (\nu \nabla \psi) - \nabla \psi \cdot \nu_u \nabla u + \nabla \cdot (\nabla \psi \cdot \nu_{\nabla u} \nabla u) \right] d\Omega + \\ \oint \psi \left[\nu \nabla w + \nu_u \nabla u w + \nu_{\nabla u} \cdot \nabla w \nabla u \right] \cdot \vec{n} ds - \oint w \left[\nabla \psi \nu + \nabla \psi \cdot \nu_{\nabla u} \nabla u \right] \cdot \vec{n} ds + \\ \int w \left[S_u \psi - \nabla \psi \cdot S_{\nabla u} \right] d\Omega + \oint \psi S_{\nabla u} w \cdot \vec{n} ds = 0 \end{aligned} \tag{3.4.7}$$

Using the duality identity, the continuous adjoint equation and boundary conditions are

$$\begin{aligned} \nabla \cdot (\nu \nabla \psi) - \nabla \psi \cdot \nu_u \nabla u + \nabla \cdot (\nabla \psi \cdot \nu_{\nabla u} \nabla u) + \psi S_u - \nabla \psi \cdot S_{\nabla u} = -j'_{\Omega[u]} \\ \psi = j'_{D[u]} \quad \vec{x} \in \Gamma^D \\ \nabla \psi \cdot \vec{n} = j'_{N[u]} \quad \vec{x} \in \Gamma^N \end{aligned} \tag{3.4.8}$$

by inspection.

3.4.2 Discrete Adjoint and Dual Consistency

The DG discretization of the model problem equation (3.4.1) in weak from (i.e. integrated by parts once) is given by:

$$\begin{aligned}
&\mathcal{N}\left(u_h, v_h\right) \equiv -\sum_{k\in\mathcal{T}_h}\int_{\Omega_k} \nu\nabla u_h\cdot\nabla v_h - v_h S\left(u_u, \nabla u_u\right) d\Omega_k + \\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i} \left\{\nu\nabla u_h\right\}\cdot\left[\!\left[v_h\right]\!\right] + \left\{\nu\nabla v_h\right\}\cdot\left[\!\left[u_h\right]\!\right] - \mu\left\{\nu\right\}\left[\!\left[u_h\right]\!\right]\cdot\left[\!\left[v_h\right]\!\right] ds + \\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^D} \nu^b\nabla u_h^+\cdot\vec{n}v_h^+ + \nu^b\nabla v_h^+\left(u_h^+ - a\right)\cdot\vec{n} - \mu\nu^b\left(u_h^+ - a\right)v_h^+ ds + \\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^N} \nu^b a_N v_h^+ ds = 0, \forall v_h\in\mathcal{V}_h
\end{aligned} \tag{3.4.9}$$

which is still a semi-linear form. In order to find the discrete adjoint, equation (3.4.9) is linearized.

$$\mathcal{N}'_{[u_h]}\left(w_h, v_h\right) \equiv \mathcal{N}_{\Omega}{}'_{[u_h]}\left(w_h, v_h\right) + \mathcal{N}_{\Gamma^i}{}'_{[u_h]}\left(w_h, v_h\right) + \mathcal{N}_{\Gamma^b}{}'_{[u_h]}\left(w_h, v_h\right) = 0 \quad \forall v_h\in\mathcal{V}_h \tag{3.4.10}$$

Where the individual terms are given by:

$$\begin{aligned}
\mathcal{N}_{\Omega}{}'_{[u_h]}\left(w_h, v_h\right) \equiv -\sum_{k\in\mathcal{T}_h}\int_{\Omega_k} &\nu\nabla w_h\cdot\nabla v_h + \nu_{u_h} w_h\nabla u_h\cdot\nabla v_h + \nu_{\nabla u}\cdot\nabla w_h\nabla u_h\cdot\nabla v_h - \\
&v_h S\left(u_u, \nabla u_u\right)_{u_h} w_h - v_h S\left(u_u, \nabla u_u\right)_{\nabla u_h}\cdot\nabla w_h d\Omega_k
\end{aligned} \tag{3.4.11}$$

$$\begin{aligned}
\mathcal{N}_{\Gamma^i}{}'_{[u_h]}\left(w_h, v_h\right) \equiv \sum_{i\in\mathcal{I}_h}\int_{\Gamma^i} &\left[\left\{\nu\nabla w_h\right\} + \left\{\nu_{u_h} w_h\nabla u_h\right\} + \left\{\nu_{\nabla u_h}\cdot\nabla w_h\nabla u_h\right\}\right]\cdot\left[\!\left[v_h\right]\!\right] \\
&- \left[\mu\left\{\nu\right\}\left[\!\left[w_h\right]\!\right] + \mu\left\{\nu_{u_h} w_h\right\}\left[\!\left[u_h\right]\!\right] + \mu\left\{\nu_{\nabla u_h}\cdot\nabla w_h\right\}\left[\!\left[u_h\right]\!\right]\right]\cdot\left[\!\left[v_h\right]\!\right] + \\
&\left\{\nu\nabla v_h\right\}\cdot\left[\!\left[w_h\right]\!\right] + \left\{\nu_{u_h} w_h\nabla v_h\right\}\cdot\left[\!\left[u_h\right]\!\right] + \left\{\nu_{\nabla u_h}\cdot\nabla w_h\nabla v_h\right\}\cdot\left[\!\left[u_h\right]\!\right] ds
\end{aligned} \tag{3.4.12}$$

$$\begin{aligned}
\mathcal{N}_{\Gamma^b}{}'_{[u_h]}\left(w_h, v_h\right) \equiv \sum_{b\in\mathcal{B}_h}\int_{\Gamma^D} &\left[\nu^b\nabla w_h^+ + \nu^b_{u^b} w_h^+\nabla u_h^+ + \nu^b_{\nabla u_h}\cdot\nabla w_h^+\nabla u_h^+\right] v_h^+\cdot\vec{n} + \\
&\nu^b\nabla v_h^+ w_h^+\cdot\vec{n} + \nu^b_{u^b} w_h^+\nabla v_h^+\left(u_h^+ - a\right)\cdot\vec{n} + \nu^b_{\nabla u_h}\cdot\nabla w_h^+\nabla v_h^+\left(u_h^+ - a\right)\cdot\vec{n} - \\
&\left[\mu\nu^b w_h^+\vec{n} + \mu\nu^b_{u^b} w_h^+\left(u_h^+ - a\right)\vec{n} + \mu\nu^b_{\nabla u_h}\nabla w_h^+\left(u_h^+ - a\right)\vec{n}\right] v_h^+\cdot\vec{n} ds + \\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^N} \nu^b_{u^b} w_h^+ a_N v_h^+ + \nu^b_{\nabla u_h}\cdot\nabla w_h^+ a_N v_h^+ ds = 0, \forall v_h\in\mathcal{V}_h
\end{aligned} \tag{3.4.13}$$

which are the integrals over volumes, interior faces and boundary faces respectively. The discrete adjoint is given by find $\psi_h \in \mathcal{V}_h$ such that

$$\begin{aligned}
&\mathcal{N}'_{[u_h]}(w_h, \psi_h) \equiv \\
&\mathcal{N}_{\Omega}{}'_{[u_h]}(w_h, \psi_h) + \mathcal{N}_{\Gamma^i}{}'_{[u_h]}(w_h, \psi_h) + \mathcal{N}_{\Gamma^b}{}'_{[u_h]}(w_h, \psi_h) = J'_{[u_h]}(w_h) \quad \forall w_h \in \mathcal{V}_h
\end{aligned} \tag{3.4.14}$$

For simplicity each integral will be handled separately. The discrete adjoint volume integral in equation (3.4.14) $\mathcal{N}_{\Omega}{}'_{[u_h]}(w_h, \psi_h)$ is integrated by parts to yield

$$\begin{aligned}
&\mathcal{N}_{\Omega}{}'_{[u_h]}(w_h, \psi_h) \equiv \sum_{k \in \mathcal{T}_h} \int_{\Omega_k} w_h \left[\nabla \cdot (\nu \nabla \psi_h) - \nabla \psi_h \cdot \nu_{u_h} \nabla u_h + \nabla \cdot (\nabla \psi_h \cdot \nu_{\nabla u} \nabla u_h)\right] + \\
&w_h \left[\psi_h S_{u_h} - \nabla \psi_h \cdot S_{\nabla u_h}\right] d\Omega_k + \mathcal{Z}_i + \mathcal{Z}_b \\
&\mathcal{Z}_i = \sum_{k \in \mathcal{T}_h} \oint_{\partial \Omega_k \setminus \Gamma^b} -\nu \nabla \psi_h^+ w_h^+ \cdot \vec{n} - \nabla \psi_h^+ \cdot \nu_{\nabla u_h} \nabla u_h^+ w_h^+ \cdot \vec{n} + \psi_h^+ S_{\nabla u_h} w_h^+ \cdot \vec{n} ds \\
&\mathcal{Z}_b = \sum_{b \in \mathcal{B}_h} \int_{\Gamma^D} -\nu^b \nabla \psi_h^+ w_h^+ \cdot \vec{n} - \nabla \psi_h^+ \cdot \nu^b_{\nabla u_h} \nabla u_h^+ w_h^+ \cdot \vec{n} + \psi_h^+ S_{\nabla u_h} w_h^+ \cdot \vec{n} ds
\end{aligned} \tag{3.4.15}$$

The $\mathcal{Z}_i$ term is added to the surface integrals and the $\mathcal{Z}_b$ term is added to the boundary surface integrals. Let $\psi_h \to \psi$ and $u_h \to u$, which results in the following for the volume integral.

$$\begin{aligned}
\mathcal{N}_{\Omega}{}'_{[u]}(w_h, \psi) \equiv &\sum_{k \in \mathcal{T}_h} \int_{\Omega_k} w_h \left[\nabla \cdot (\nu \nabla \psi) - \nabla \psi \cdot \nu_u \nabla u + \nabla \cdot (\nabla \psi \cdot \nu_{\nabla u} \nabla u)\right] + \\
&w_h \left[\psi S_u - \nabla \psi \cdot S_{\nabla u}\right] d\Omega_k + J'_{\Omega[u]}(w_h) = 0
\end{aligned} \tag{3.4.16}$$

The volume integral in equation (3.4.16) evaluated using the continuous adjoint variable is zero by definition of the continuous adjoint equation given in equation (3.4.8).

The surface integral in equation (3.4.14) given by $\mathcal{N}_{\Gamma^i}{}'_{[u_h]}(w_h, \psi_h)$ is

$$\begin{aligned}
\mathcal{N}_{\Gamma^i}{}'_{[u_h]}(w_h, \psi_h) \equiv &\sum_{i \in \mathcal{I}_h} \int_{\Gamma^i} \left[\{\nu \nabla w_h\} + \{\nu_{u_h} w_h \nabla u_h\} + \{\nu_{\nabla u_h} \cdot \nabla w_h \nabla u_h\}\right] \cdot [\![\psi_h]\!] \\
&- \left[\mu \{\nu\} [\![w_h]\!] + \mu \{\nu_{u_h} w_h\} [\![u_h]\!] + \mu \{\nu_{\nabla u_h} \cdot \nabla w_h\} [\![u_h]\!]\right] \cdot [\![\psi_h]\!] + \\
&\{\nu \nabla \psi_h\} \cdot [\![w_h]\!] + \{\nu_{u_h} w_h \nabla \psi_h\} \cdot [\![u_h]\!] + \{\nu_{\nabla u_h} \cdot \nabla w_h \nabla \psi_h\} \cdot [\![u_h]\!] ds
\end{aligned} \tag{3.4.17}$$

Translating this into an element based integral and adding the $\mathcal{Z}_i$ terms from equation (3.4.15) gives

$$\begin{aligned}\mathcal{N}_{\Gamma^i{}'_{[u_h]}}(w_h,\psi_h) \equiv \sum_{k\in\mathcal{T}_h}\oint_{\partial\Omega_k\setminus\Gamma^b}&\left[\frac{1}{2}\nu\nabla w_h+\frac{1}{2}\nu_{u_h}w_h\nabla u_h+\frac{1}{2}\nu_{\nabla u_h}\cdot\nabla w_h\nabla u_h\right]\cdot[\![\psi_h]\!]\\ -&\left[\mu\frac{1}{2}\nu[\![w_h]\!]+\mu\frac{1}{2}\nu_{u_hw_h}[\![u_h]\!]+\mu\frac{1}{2}\nu_{\nabla u_h}\cdot\nabla w_h[\![u_h]\!]\right]\cdot[\![\psi_h]\!]+\\ &\{\nu\nabla\psi_h\}w_h^+\cdot\vec{n}+\{\nu_{u_h}w_h\nabla\psi_h\}\cdot[\![u_h]\!]+\{\nu_{\nabla u_h}\cdot\nabla w_h\nabla\psi_h\}\cdot[\![u_h]\!]-\\ &\nu\nabla\psi_h^+w_h^+\cdot\vec{n}-\nabla\psi_h^+\cdot\nu_{\nabla u_h}\nabla u_h^+w_h^+\cdot\vec{n}+\psi_h^+S_{\nabla u_h}w_h^+\cdot\vec{n}ds\end{aligned}\tag{3.4.18}$$

Making use of the following identity

$$w_h^+\nabla\psi_h^+\cdot\vec{n}=\{\nabla\psi_h\}\cdot\vec{n}w_h^++\frac{1}{2}[\![\nabla\psi_h]\!]w_h^+\tag{3.4.19}$$

results in

$$\begin{aligned}\mathcal{N}_{\Gamma^i{}'_{[u_h]}}(w_h,\psi_h) \equiv \sum_{k\in\mathcal{T}_h}\oint_{\partial\Omega_k\setminus\Gamma^b}&\left[\frac{1}{2}\nu\nabla w_h+\frac{1}{2}\nu_{u_h}w_h\nabla u_h+\frac{1}{2}\nu_{\nabla u_h}\cdot\nabla w_h\nabla u_h\right]\cdot[\![\psi_h]\!]\\ -&\left[\mu\frac{1}{2}\nu[\![w_h]\!]+\mu\frac{1}{2}\nu_{u_h}w_h[\![u_h]\!]+\mu\frac{1}{2}\nu_{\nabla u_h}\cdot\nabla w_h[\![u_h]\!]\right]\cdot[\![\psi_h]\!]+\\ &\{\nu\nabla\psi_h\}w_h^+\cdot\vec{n}+\{\nu_{u_h}w_h\nabla\psi_h\}\cdot[\![u_h]\!]+\{\nu_{\nabla u_h}\cdot\nabla w_h\nabla\psi_h\}\cdot[\![u_h]\!]-\\ &\{\nu\nabla\psi_h\}w_h^+\cdot\vec{n}-\frac{1}{2}[\![\nu\nabla\psi_h]\!]w_h^+-\nabla\psi_h^+\cdot\nu_{\nabla u_h}\nabla u_h^+w_h^+\cdot\vec{n}+\\ &\psi_hS_{\nabla u_h}w_h^+\cdot\vec{n}ds\end{aligned}\tag{3.4.20}$$

Again let $\psi_h\to\psi$ and $u_h\to u$. From the definition of the jump in equation (2.2.6), it is clear that the jump in both exact solution u and exact adjoint ψ are zero($[\![\psi]\!]=0$ and $[\![u]\!]=0$) canceling many terms in equation (3.4.20). The remaining terms in equation (3.4.20) are

$$\mathcal{N}_{\Gamma^i{}'_{[u_h]}}(w_h,\psi)=\sum_{k\in\mathcal{T}_h}\oint_{\partial\Omega_k\setminus\Gamma^b}w_h^+\left(\psi S_{\nabla u}\cdot\vec{n}-\nabla\psi\cdot\nu_{\nabla u}\nabla u\cdot\vec{n}\right)ds\tag{3.4.21}$$

This expression should be equal to zero, but it is not due to the source term and viscosity dependence on state-variable gradients. This is one source of dual inconsistency.

The boundary integral in equation (3.4.14) $\mathcal{N}_{\Gamma^b{}'_{[u_h]}}(w_h,\psi_h)$ plus the boundary term $\mathcal{Z}_b$

resulting from the integration by parts of the volume term in equation (3.4.15) is

$$\begin{aligned}
&\mathcal{N}_{\Gamma^{b}{}'_{[u_h]}}(w_h,\psi_h) \equiv \sum_{b\in\mathcal{B}_h}\int_{\Gamma^D}\left[\nu^b\nabla w_h^+ + \nu^b_{u^b}w_h^+\nabla u_h^+ + \nu^b_{\nabla u_h}\cdot\nabla w_h^+\nabla u_h^+\right]\psi_h^+\cdot\vec{n}+\\
&\nu^b\nabla\psi_h^+ w_h^+\cdot\vec{n} + \nu^b_{u^b}w_h^+\nabla\psi_h^+\left(u_h^+-a\right)\cdot\vec{n} + \nu^b_{\nabla u_h}\cdot\nabla w_h^+\nabla\psi_h^+\left(u_h^+-a\right)\cdot\vec{n}-\\
&\left[\mu\nu^b w_h^+\vec{n} + \mu\nu^b_{u^b}w_h^+\left(u_h^+-a\right)\vec{n} + \mu\nu^b_{\nabla u_h}\nabla w_h^+\left(u_h^+-a\right)\vec{n}\right]\psi_h\cdot\vec{n}-\\
&\nu^b\nabla\psi_h^+ w_h^+\cdot\vec{n} - \nabla\psi_h^+\cdot\nu^b_{\nabla u_h}\nabla u_h^+ w_h^+\cdot\vec{n} + \psi_h^+ S_{\nabla u_h}w_h^+\cdot\vec{n}ds+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^N}\nu^b_{u^b}w_h^+ a_N\psi_h^+ + \nu^b_{\nabla u_h}\cdot\nabla w_h^+ a_N\psi_h^+ ds
\end{aligned} \tag{3.4.22}$$

Again let $\psi_h \to \psi$ and $u_h \to u$ and subtracting the boundary functional definitions, $\mathfrak{J}'_{\Gamma^D[u]}(w_h)$ in equation (3.4.24) and $J'_{\Gamma^N[u]}(w_h)$ in equation (3.4.4). results in the cancellation almost every term, the remaining terms are given by:

$$\begin{aligned}
&\mathcal{N}_{\Gamma^{b}{}'_{[u]}}(w_h,\psi) - \mathfrak{J}'_{\Gamma^D[u]}(w_h) - J'_{\Gamma^N[u]}(w_h) =\\
&\mathcal{N}_{\Gamma^{b}{}'_{[u]}}(w_h,\psi) - \sum_{b\in\mathcal{B}_h}\int_{\Gamma^D} j_{D}{}'_{[u]}\left(\nu^b\nabla w_h^+\cdot\vec{n} + \nu^b_u w_h^+\nabla u\cdot\vec{n} + \nu^b_{\nabla u}\cdot\nabla w_h^+\nabla u\cdot\vec{n}\right)+\\
&j_{D}{}'_{[u]}\left(-\mu\nu^b w_h^+ - \mu\nu^b_{u^b}w_h^+\left(u_h^+-a\right) - \mu\nu^b_{\nabla u_h}\cdot\nabla w_h^+\left(u_h^+-a\right)\right)ds-\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^N} j_{N}{}'_{[u]}\left(\nu^b_u w_h^+\nabla u\cdot\vec{n} + \nu^b_{\nabla u}\cdot\nabla w_h^+\nabla u\cdot\vec{n}\right)ds =\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^D} w_h^+\left(-\nabla\psi\cdot\nu_{\nabla u}\nabla u + \psi S_{\nabla u}\right)\cdot\vec{n}ds
\end{aligned} \tag{3.4.23}$$

where the target functional $J'_{\Gamma[u]}(w_h)$ has been modified according to reference [49]. Note that as $\psi_h \to \psi$, then $\psi \to j'_D$ on a Dirichlet boundary and $\nabla\psi\cdot\vec{n} \to j'_N$ on a Neumann boundary. Likewise as $u_h \to u$ on the boundary, then $u \to a$ on a Dirichlet boundary and $\nabla u\cdot\vec{n} \to a_N$ on a Neumann boundary. The modified target function $\mathfrak{J}'_{\Gamma^D[u]}(w_h)$ is given as

$$\begin{aligned}
&\mathfrak{J}'_{\Gamma^D[u]}(w_h) = J'_{\Gamma^D[u]}(w_h)+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^D} -\mu\nu^b w_h^+ - \mu\nu^b_{u^b}w_h^+\left(u_h^+-a\right) - \mu\nu^b_{\nabla u_h}\cdot\nabla w_h^+\left(u_h^+-a\right)ds
\end{aligned} \tag{3.4.24}$$

which adds the penalty term to the functional definition in equation (3.4.4), in order cancel the penalty contribution to the Dirichlet boundary integral in equation (3.4.22). This modification is not arbitrary and has an intuitive explanation. Essentially when evaluating the

functional on the surface one should use the gradient from the element plus the so-called "penalty gradient" $(\mu (u_h - a))$. In other words, in order to obtain a dual consistent discretization one needs to define the functional based on how the boundary surface integral is computed for the discretization in equation (3.4.9).

From the above analysis it is clear that dual inconsistency can occur for this problem and discretization, as seen in equation (3.4.25).

$$\begin{aligned}
&\mathcal{N}'_{[u]}(w_h, \psi) \equiv \mathcal{N}_{\Omega}{}'_{[u]}(w_h, \psi) + \mathcal{N}_{\Gamma^i}{}'_{[u]}(w_h, \psi) + \mathcal{N}_{\Gamma^b}{}'_{[u]}(w_h, \psi) = \\
&0 + \sum_{k \in \mathcal{T}_h} \oint_{\partial \Omega_k \setminus \Gamma^b} w_h^+ \left(\psi S_{\nabla u} \cdot \vec{n} - \nabla \psi \cdot \nu_{\nabla u} \nabla u \cdot \vec{n}\right) ds + \\
&\sum_{b \in \mathcal{B}_h} \int_{\Gamma^D} w_h^+ \left(-\nabla \psi \cdot \nu_{\nabla u} \nabla u + \psi S_{\nabla u}\right) \cdot \vec{n} ds \quad \forall w_h \in \mathcal{V}_h
\end{aligned} \tag{3.4.25}$$

However, the sources of inconsistency are relegated to state-variable gradient dependencies of the viscosity and the source term. If one eliminates the dependence of these terms on state-variable gradients, then the discretization is dual consistent. Either of the artificial viscosity methods employed for shock capturing is dual consistent because neither the viscosity or the source term has any dependence on the state-variable gradient. However, this is not true of the SA turbulence model(equation (2.1.2)) used to close the RANS equations in this work. The SA turbulence model has a source term that depends on the state-variable gradient causing the model to be dual inconsistent. Attempts were made to eliminate the dual inconsistency by following the methods in reference [42]. Unfortunately, applying these methods increased computational time by a factor of two and resulted in a significantly less robust solver.

The viscosity dependence on the state-variable gradient has implications beyond artificial viscosity. Large Eddy Simulation(LES) often employs a sub-grid scale(SGS) viscosity, which is algebraic and depends on the gradients of the state-variables. The dual consistency analysis of the model problem implies that SIP discretizations of LES models employing algebraic SGS models, which depend on state-variable gradients, will result in a dual inconsistent discretization. Hence extending the presented SIP method to LES will not be straight forward, as optimal accuracy will not be obtained for these types of algebraic SGS models.

3.5 Dual Consistency of Boundary Conditions for the Euler Equations

An important aspect of the discretization is the effect of boundary conditions on dual consistency. Dual consistency gives guidelines for constructing optimally accurate boundary conditions, both from a solution error point of view and functional error point of view. This section will detail the analysis of the wall boundary condition applied to the Euler equations. The purpose of this section is to give an example of how the analysis is conducted and what conclusions can be drawn from this analysis.

Consider the compressible Euler equations written in a general flux form equation (2.1.12). Recall that **bold** face symbols are vectors in the number of unknowns and vectors with arrows over the top are vectors in the number of physical dimensions. Therefore a bold face symbol with an arrow over it is a rank 2 tensor. Neglecting the temporal derivative $\frac{\partial \mathbf{u}}{\partial t}$ and artificial diffusion fluxes $\vec{\mathbf{F}}_{ad}$ in equation (2.1.12), the steady-state Euler equations are written as

$$\nabla \cdot \vec{\mathbf{F}}(\mathbf{u}) = 0 \tag{3.5.1}$$

where the subscript c is dropped because it is not necessary for this analysis. Also consider the following functional,

$$J(\mathbf{u}) = \int_{\Gamma} j(C(\mathbf{u}))ds \tag{3.5.2}$$

which is defined only on the surface, since for aerodynamic flows this is the principal type of functional, examples of surface based functionals are lift or drag. Here $C(\mathbf{u})$ is a non-linear function that can change the state vector into derived products such as pressure or temperature. j is a surface sampling function indicating what surface to use and how to use it. For example if the desired target is lift then

$$\begin{aligned} C(\mathbf{u}) &\equiv P(\mathbf{u}) \\ j &\equiv \vec{n} \cdot (-sin(\alpha), cos(\alpha))^T = \vec{n} \cdot \vec{z}_{wall} \\ \vec{z}_{wall} &= (-sin(\alpha), cos(\alpha))^T \end{aligned} \tag{3.5.3}$$

where $P(\mathbf{u})$ is the pressure, α is the angle of attack and the integral is done on Γ_{wall}, which

is the wall boundary.

3.5.1 Continuous Adjoint of Euler Equations

The steady-state Euler equations represent a non-linear operator, which is linearized to obtain the continuous adjoint equation. Linearizing the steady-state Euler equations, given in equation (3.5.1), around the state $\mathbf{u}$.

$$\nabla \cdot \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} (\mathbf{w}) \right) = \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} \right) \cdot \nabla \mathbf{w} = 0 \tag{3.5.4}$$

Linearizing the functional about $\mathbf{u}$ as well, results in the following.

$$J'_{[\mathbf{u}]} (\mathbf{w}) = \int_{\Gamma} j'_{[C\mathbf{u}]} C'_{[\mathbf{u}]} \mathbf{w} ds \tag{3.5.5}$$

Taking the inner product of the adjoint variable with the linearized PDE results in

$$\int_{\Omega} \boldsymbol{\psi}^T \nabla \cdot \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} (\mathbf{w}) \right) d\Omega = 0 \tag{3.5.6}$$

which is integrated by parts once to obtain

$$- \int_{\Omega} \nabla \boldsymbol{\psi}^T \cdot \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} (\mathbf{w}) \right) d\Omega + \int_{\Gamma} \boldsymbol{\psi}^T \vec{\mathbf{F}}'_{[\mathbf{u}]} \mathbf{w} \cdot \vec{n} ds = J'_{[\mathbf{u}]} (\mathbf{w}) \tag{3.5.7}$$

and is transposed to give

$$- \int_{\Omega} \mathbf{w}^T \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} \right)^T \cdot \nabla \boldsymbol{\psi} d\Omega + \int_{\Gamma} \mathbf{w}^T \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} \right)^T \boldsymbol{\psi} \cdot \vec{n} ds = J'_{[\mathbf{u}]} (\mathbf{w}) \tag{3.5.8}$$

This results in the continuous adjoint equation, found by analogy to the duality identity in equation (3.1.6)

$$\left(\vec{\mathbf{F}}'_{[\mathbf{u}]} \right)^T \cdot \nabla \boldsymbol{\psi} = 0 \tag{3.5.9}$$

where the right hand side is zero because in this case there are no volume sampled objectives. The adjoint boundary conditions are a bit more difficult to obtain. Recall the part of the identity in equation (3.1.6) $\left\langle C'_{[u]} (w) , (B^*)'_{[u]} (\psi) \right\rangle_{\Gamma}$ and recall the definition of $C(\mathbf{u}) = P(\mathbf{u})$, which is the pressure, hence the the linearization of $C(\mathbf{u})$

$$C'_{[\mathbf{u}]} = P'_{[\mathbf{u}]} \tag{3.5.10}$$

The flux at the slip wall boundary is pressure alone and thus

$$\int_{\Gamma} \mathbf{w} \left(\vec{\mathbf{F}}'_{[\mathbf{u}]} \right)^T \boldsymbol{\psi} \cdot \vec{n} ds = \int_{\Gamma} \mathbf{w} P'^{T}_{[\mathbf{u}]} \vec{n} \cdot \vec{z}_{wall} ds \tag{3.5.11}$$

which allows for the deduction of the continuous adjoint boundary condition. Considering that at the wall $\vec{\mathbf{F}} = (0, P\vec{n}, 0)$, one can deduce the continuous adjoint boundary condition as

$$\vec{\psi} \cdot \vec{n} = \vec{n} \cdot \vec{z}_{wall} \tag{3.5.12}$$

which pertains to the momentum adjoint variables. The density and energy adjoint variables have Neumann conditions just like the primal density and energy as in equation (2.9.9). Now that the continuous adjoint equation and the associated adjoint variable wall boundary condition are known, a dual consistency analysis is conducted.

3.5.2 Dual Consistency of Discrete Euler Equations

Consider the DG discretized Euler equations with the possibility of employing an approximate Riemann solver(discussed in Chapter 2) on the boundary. Note that during this analysis the $()'$ notation will not be used, for the clarity of linearizing the approximate Riemann fluxes, $\mathcal{H}$. The discretized steady-state Euler equations from Chapter 2 equation (2.2.3) are given as

$$\begin{aligned} \mathcal{N}(\mathbf{u}_h, \mathbf{v}_h) \equiv & -\sum_{k \in \mathcal{T}_h} \int_{\Omega_k} \nabla \mathbf{v}_h^T \cdot \vec{\mathbf{F}}(\mathbf{u}_h) \, d\Omega_k + \sum_{k \in \mathcal{T}_h} \oint_{\partial \Omega_k \setminus \Gamma^b} \left(\mathbf{v}_h^+ \right)^T \mathcal{H}\left(\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n} \right) ds + \\ & \sum_{b \in \mathcal{B}_h} \int_{\Gamma^b} \left(\mathbf{v}_h^+ \right)^T \mathcal{H}_b\left(\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+ \right), \vec{n} \right) ds = 0 \quad \forall \mathbf{v}_h \in \mathcal{V}_h \end{aligned} \tag{3.5.13}$$

where the viscous($\vec{\mathbf{F}}_v$), artificial diffusion($\vec{\mathbf{F}}_{ad}$), and source(S) terms in equation (2.2.3) are set to zero. Equation 3.5.13 is linearized to obtain

$$\begin{aligned}
&\mathcal{N}'_{[\mathbf{u}_h]}\left(\mathbf{w}_h, \mathbf{v}_h\right) \equiv \sum_{k\in\mathcal{T}_h} \int_{\Omega_k} \nabla \mathbf{v}_h^T \cdot \frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{u}_h}_{[\mathbf{u}_h]} \mathbf{w}_h d\Omega_k + \\
&\sum_{k\in\mathcal{T}_h} \oint_{\partial\Omega_k\setminus\Gamma^b} \left(\left(\mathbf{v}_h^+\right)^T \frac{\partial \mathcal{H}}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right]} \mathbf{w}_h^+ + \frac{\partial \mathcal{H}}{\partial \mathbf{u}_h^-}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right]} \mathbf{w}_h^- \right) ds \\
&\sum_{b\in\mathcal{B}_h} \int_{\Gamma^b} \left(\mathbf{v}_h^+\right)^T \left(\frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \vec{n}\right]} + \frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_b}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \vec{n}\right]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+\right]} \right) \mathbf{w}_h^+ ds = 0 \quad \forall \mathbf{v}_h \in \mathcal{V}_h
\end{aligned} \tag{3.5.14}$$

which is a bi-linear form for $\mathbf{w}_h$. The corresponding discrete adjoint equation is

$$\begin{aligned}
&\mathcal{N}'_{[\mathbf{u}_h]}\left(\mathbf{w}_h, \boldsymbol{\psi}_h\right) \equiv \sum_{k\in\mathcal{T}_h} \int_{\Omega_k} \left(\mathbf{w}_h^+\right)^T \left[\frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{u}_h}_{[\mathbf{u}_h]} \right]^T \cdot \nabla \boldsymbol{\psi}_h d\Omega_k + \\
&\sum_{k\in\mathcal{T}_h} \oint_{\partial\Omega_k\setminus\Gamma^b} \left(\left(\mathbf{w}_h^+\right)^T \left[\frac{\partial \mathcal{H}}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right]} \right]^T + \left(\mathbf{w}_h^-\right)^T \left[\frac{\partial \mathcal{H}}{\partial \mathbf{u}_h^-}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right]} \right]^T \right) \boldsymbol{\psi}_h^+ ds \\
&\sum_{b\in\mathcal{B}_h} \int_{\Gamma^b} \left(\mathbf{w}_h^+\right)^T \left(\frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \vec{n}\right]} + \frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_b}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \vec{n}\right]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+\right]} \right)^T \boldsymbol{\psi}_h^+ ds = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h
\end{aligned} \tag{3.5.15}$$

Following the previous section dual consistency is shown one integral at at time and the continuous solutions will be inserted into these small pieces. First consider the volume integral of the discrete adjoint equations in equation (3.5.15).

$$\mathcal{N}_{\Omega}{}'_{[\mathbf{u}_h]}\left(\mathbf{w}_h, \boldsymbol{\psi}_h\right) \equiv \sum_{k\in\mathcal{T}_h} \int_{\Omega_k} \mathbf{w}_h^T \left[\frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{u}_h}_{[\mathbf{u}_h]} \right]^T \cdot \nabla \boldsymbol{\psi}_h d\Omega_k \quad \forall \mathbf{w}_h \in \mathcal{V}_h \tag{3.5.16}$$

To show dual consistency let $\mathbf{u}_h \to \mathbf{u}$ and $\boldsymbol{\psi}_h \to \boldsymbol{\psi}$

$$\mathcal{N}_{\Omega}{}'_{[\mathbf{u}]}\left(\mathbf{w}_h, \boldsymbol{\psi}\right) \equiv \sum_{k\in\mathcal{T}_h} \int_{\Omega_k} \mathbf{w}_h^T \left[\frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{u}}_{[\mathbf{u}]} \right]^T \cdot \nabla \boldsymbol{\psi} d\Omega_k = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h \tag{3.5.17}$$

by definition of the continuous adjoint equation (3.5.9). The interior surface term in equation (3.5.15) can be re-written in a face based from.

$$\mathcal{N}_{\Gamma^i}{}'_{[\mathbf{u}_h]}\left(\mathbf{w}_h, \boldsymbol{\psi}_h\right) \equiv \sum_{i\in\mathcal{I}_h} \int_{\Gamma^i} \left(\mathbf{w}_h^+\right)^T \left[\frac{\partial \mathcal{H}}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right]} \right]^T [\![\boldsymbol{\psi}_h]\!] \cdot \vec{n} ds \quad \forall \mathbf{w}_h \in \mathcal{V}_h \tag{3.5.18}$$

Again let let $\mathbf{u}_h \to \mathbf{u}$ and $\boldsymbol{\psi}_h \to \boldsymbol{\psi}$

$$\mathcal{N}_{\Gamma^i{}'_{[\mathbf{u}]}}(\mathbf{w}_h, \boldsymbol{\psi}) \equiv \sum_{i \in \mathcal{I}_h} \int_{\Gamma^i} \left(\mathbf{w}_h^+\right)^T \left[\frac{\partial \mathcal{H}}{\partial \mathbf{u}^+}_{[\mathbf{u},\mathbf{u},\vec{n}]}\right]^T [\![\boldsymbol{\psi}]\!] \cdot \vec{n} ds = 0 \quad \forall \mathbf{w}_h \in \mathcal{V}_h \tag{3.5.19}$$

which is zero by $[\![\boldsymbol{\psi}]\!] = 0$. Finally the boundary terms in equation (3.5.15) are given as

$$\begin{aligned} &\mathcal{N}_{\Gamma^b{}'_{[\mathbf{u}_h]}}(\mathbf{w}_h, \boldsymbol{\psi}_h) \equiv \\ &\sum_{b \in \mathcal{B}_h} \int_{\Gamma^b} \left(\mathbf{w}_h^+\right)^T \left(\frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \vec{n}\right]} + \frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_b}_{\left[\mathbf{u}_h^+, \mathbf{u}_h^b\left(\mathbf{u}_h^+\right), \vec{n}\right]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}_h^+}_{\left[\mathbf{u}_h^+\right]}\right)^T \boldsymbol{\psi}_h^+ ds \\ &\forall \mathbf{w}_h \in \mathcal{V}_h \end{aligned} \tag{3.5.20}$$

into which the substitution $\mathbf{u}_h \to \mathbf{u}$ and $\boldsymbol{\psi}_h \to \boldsymbol{\psi}$ is made.

$$\begin{aligned} \mathcal{N}_{\Gamma^b{}'_{[\mathbf{u}]}}(\mathbf{w}_h, \boldsymbol{\psi}) \equiv \sum_{b \in \mathcal{B}_h} \int_{\Gamma^b} \left(\mathbf{w}_h^+\right)^T \left(\frac{\partial \mathcal{H}_b}{\partial \mathbf{u}}_{\left[\mathbf{u}, \mathbf{u}^b(\mathbf{u}), \vec{n}\right]} + \frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_b}_{\left[\mathbf{u}, \mathbf{u}^b(\mathbf{u}), \vec{n}\right]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}}_{[\mathbf{u}]}\right)^T \boldsymbol{\psi} ds = \\ \sum_{b \in \mathcal{B}_h} \int_{\Gamma^b} j'_{[C\mathbf{u}]} C'_{[\mathbf{u}]} \mathbf{w_h} ds \quad \forall \mathbf{w}_h \in \mathcal{V}_h \end{aligned} \tag{3.5.21}$$

If the functional definition $j(C(\mathbf{u}))$ is modified so that it is evaluated at the boundary state then one obtains $j\left(C\left(\mathbf{u}^b\right)\right)$ as:

$$j'_{[u]} = \frac{\partial P}{\partial \mathbf{u}^b}_{[u]} \frac{\partial \mathbf{u}^b}{\partial \mathbf{u}}_{[\mathbf{u}]}^T (0, \vec{n}, 0) \cdot \vec{z}_{wall} \tag{3.5.22}$$

Recall the continuous adjoint boundary condition $\vec{\psi} \cdot \vec{n} = \vec{n} \cdot \vec{z}_{wall}$. Therefore on the left hand side one needs the following to make this equality true.

$$\left(\frac{\partial \mathcal{H}_b}{\partial \mathbf{u}}_{\left[\mathbf{u}, \mathbf{u}^b(\mathbf{u}), \vec{n}\right]} + \frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_b}_{\left[\mathbf{u}, \mathbf{u}^b(\mathbf{u}), \vec{n}\right]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}}_{[\mathbf{u}]}\right)^T = \frac{\partial P}{\partial \mathbf{u}_b}_{[\mathbf{u}]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}}_{[\mathbf{u}]} \tag{3.5.23}$$

This is only possible if

$$\mathcal{H}_b\left(\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right) = \vec{n} \cdot \vec{\mathbf{F}}\left(\mathbf{u}_b\left(\mathbf{u}_h^+\right)\right) \tag{3.5.24}$$

which upon linearization and substitution with the continuous solution $\mathbf{u}$, one obtains

$$\begin{aligned} &\left(\mathbf{w}_h^+\right)^T \left(\frac{\partial \mathcal{H}_b}{\partial \mathbf{u}}_{\left[\mathbf{u}, \mathbf{u}^b(\mathbf{u}), \vec{n}\right]} + \frac{\partial \mathcal{H}_b}{\partial \mathbf{u}_b}_{\left[\mathbf{u}, \mathbf{u}^b(\mathbf{u}), \vec{n}\right]} \frac{\partial \mathbf{u}_b}{\partial \mathbf{u}}_{[\mathbf{u}]}\right)^T \boldsymbol{\psi} = \\ &\frac{\partial P}{\partial \mathbf{u}^b}_{[u]} \frac{\partial \mathbf{u}^b}{\partial \mathbf{u}}_{[\mathbf{u}]}^T \mathbf{w}_h^+ \boldsymbol{\psi} \end{aligned} \tag{3.5.25}$$

Using equation (3.5.25),equation (3.5.22) and the definition of the adjoint boundary condition equation (3.5.12) in equation (3.5.21) gives

$$\begin{aligned}&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\left[\frac{\partial P}{\partial \mathbf{u}^b}_{[u]}\frac{\partial \mathbf{u}^b}{\partial \mathbf{u}}_{[\mathbf{u}]}\right]^T \mathbf{w}_h^+\left(0,\vec{n},0\right)\cdot\vec{z}_{wall}ds=\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}j'_{[u]}ds=\\&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\left[\frac{\partial P}{\partial \mathbf{u}^b}_{[u]}\frac{\partial \mathbf{u}^b}{\partial \mathbf{u}}_{[\mathbf{u}]}\right]^T \mathbf{w}_h^+\left(0,\vec{n},0\right)\cdot\vec{z}_{wall}ds\end{aligned}\tag{3.5.26}$$

$$\begin{aligned}&\mathcal{N}'_{[u]}\left(w_h,\psi\right)\equiv\mathcal{N}_{\Omega}{}'_{[u]}\left(w_h,\psi\right)+\mathcal{N}_{\Gamma^i}{}'_{[u]}\left(w_h,\psi\right)+\mathcal{N}_{\Gamma^b}{}'_{[u]}\left(w_h,\psi\right)=\\&0+0+\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}j'_{[u]}ds=\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}j'_{[u]}ds\quad\forall\mathbf{w}_h\in\mathcal{V}_h\end{aligned}\tag{3.5.27}$$

Thus the scheme is dual consistent under this type of boundary condition as seen in equation (3.5.27). While it is technically possible to devise a boundary condition of the form $\mathcal{H}_b\left(\mathbf{u}_h^+,\mathbf{u}_b\left(\mathbf{u}_h^+\right),\vec{n}\right)$ that is dual consistent. It is much simpler to form a boundary condition of the form $\mathcal{H}_b\left(\mathbf{u}_b\left(\mathbf{u}_h^+\right),\vec{n}\right)$, which is dual consistent by the presented analysis.

While the analysis of dual consistency is shown for model problems, the discretization of the Navier-Stokes equations including stabilization terms and boundary conditions is dual consistent. The proof of this is presented very nicely in reference [49].

3.6 Numerical Examples

While dual consistency analysis is helpful it does not necessarily prove that the implementation of the boundary conditions and other terms is dual consistent. Therefore a numerical example is computed to demonstrate the dual consistency of the actual computer program. References [56, 74] have shown that dual consistency errors only affect the functional error convergence rate of even p values i.e. $p = 2, 4, 6....$ For a dual inconsistent discretization the even p values show error convergence properties of the odd p value one order lower than p. For example, a $p = 2$ dual inconsistent discretization would have a functional error convergence rate of 2 instead of 4. Therefore to demonstrate that a scheme is dual consistent it is sufficient to show the error convergence rate of a $p = 2$ discretization, compared to a $p = 1$ discretization. In addition to examining function accuracy, some sample contour plots

of the adjoint solutions are shown. Smooth adjoint solution contours are a qualitative sign of adjoint consistency [56].

3.6.1 Laminar Viscous NACA0012 Airfoil: Drag Error

To demonstrate the dual consistency of the implementation the laminar viscous flow over a NACA0012 is considered. Uniform mesh refinement is conducted at $p = 1$ and $p = 2$ and the drag error is computed at each uniform refinement. The flow conditions are free-stream Mach number $M_\infty = .5$, angle of attack $\alpha = 1^o$, and Reynolds number based on airfoil chord length $Re = 5,000$. The reference drag value is computed at a discretization order of $p = 4$ with $250,000$ unknowns. Figure 3.1 demonstrates the drag error convergence rate for $p = 1$

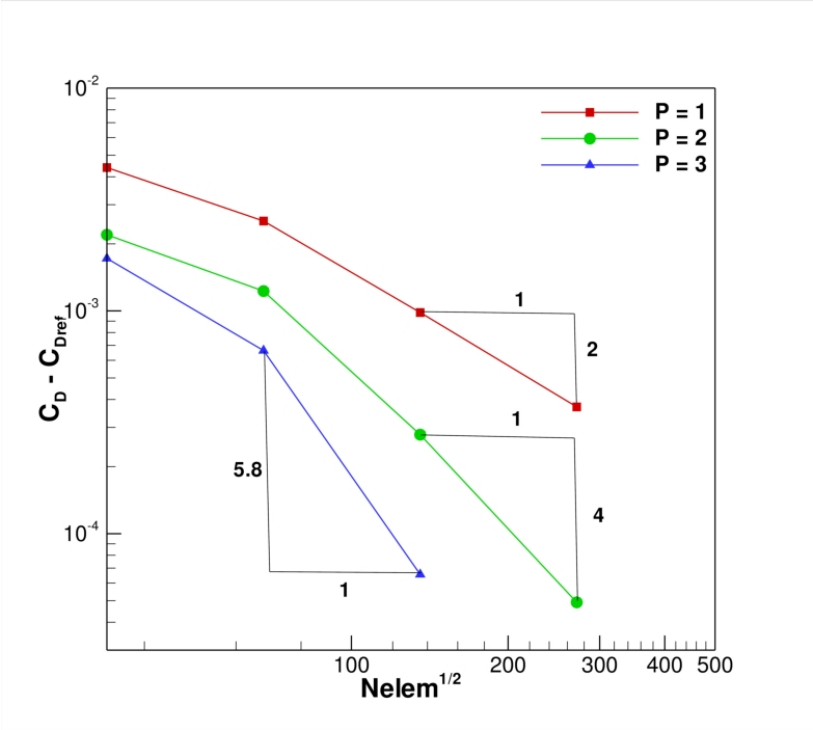


Figure 3.1: Drag error convergence for the flow over a NACA0012 airfoil computed at $p = 1$ and $p = 2$ to demonstrate dual consistency.

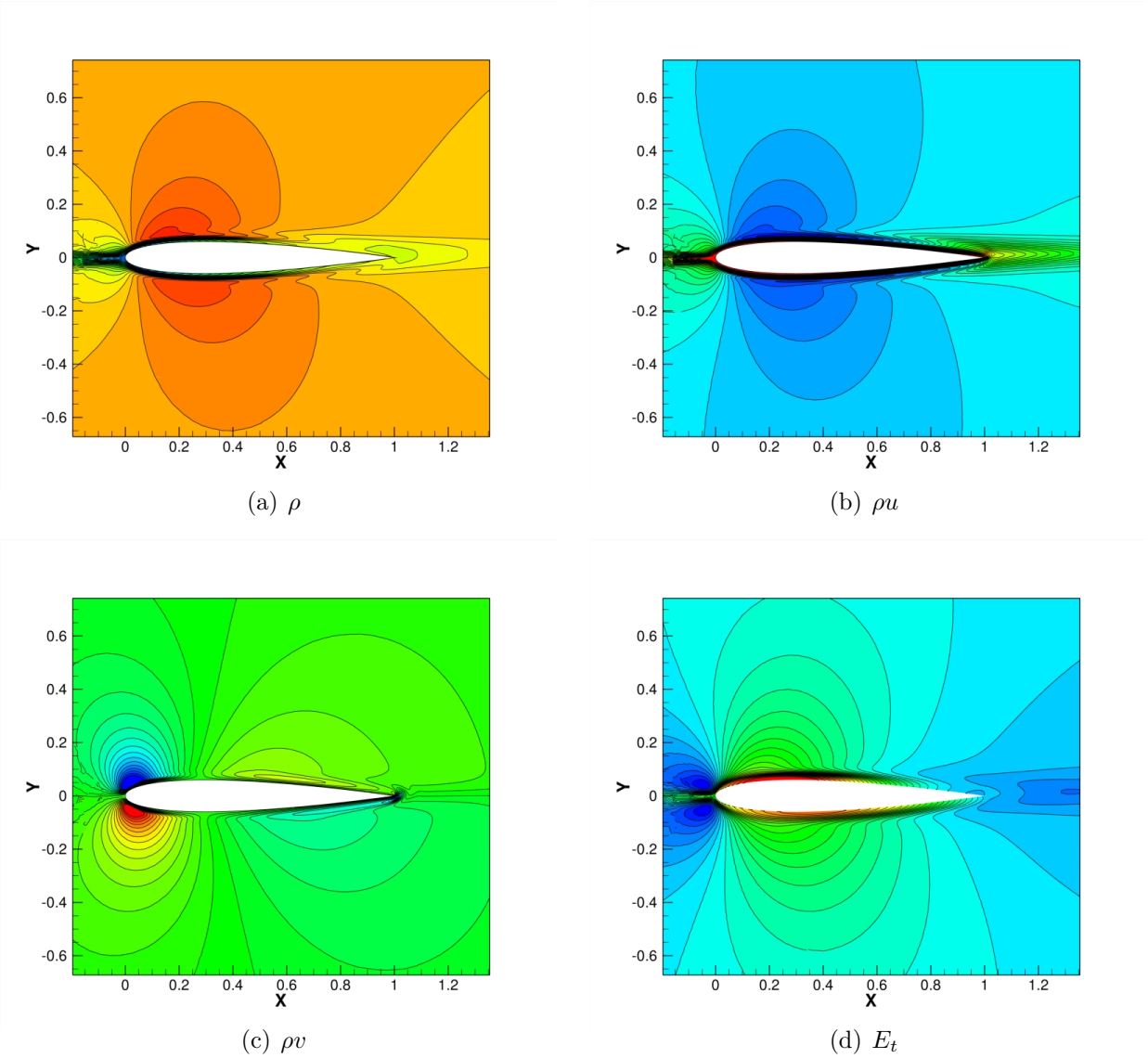
and $p = 2$ discretizations, using uniform mesh refinement. The scheme is dual consistent since the $p = 1$ result converges with order 2 while the $p = 2$ result converges with order 4.

3.6.2 Example Adjoint Solutions

This section presents some sample adjoint solutions for a viscous laminar flow and an inviscid transonic flow. These give a qualitative picture of the adjoint variable distributions in space for two flow problems.

Figure 3.2 shows the contours for the adjoint of drag for the laminar flow over a NACA0012 airfoil at $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$. Notice that all the adjoint solutions are smooth near the boundary as well as in the wake. The leading edge looks a little non-smooth, which is a post processing artifact due to coarse mesh resolution at the leading edge as well as the inability of the plotting program to handle high-order data sets adequately(Section 2.8). Smooth adjoint contours are a qualitative sign of a dual consistent discretization, which is free of noisy adjoint variables that give improper error estimates. Also notice the "reverse wake" coming off the leading edge of the airfoil. The adjoint shows that the upstream region is important for the functional of drag. This is non-intuitive, which is why one appeals to the adjoint for adaptive refinement guidance.

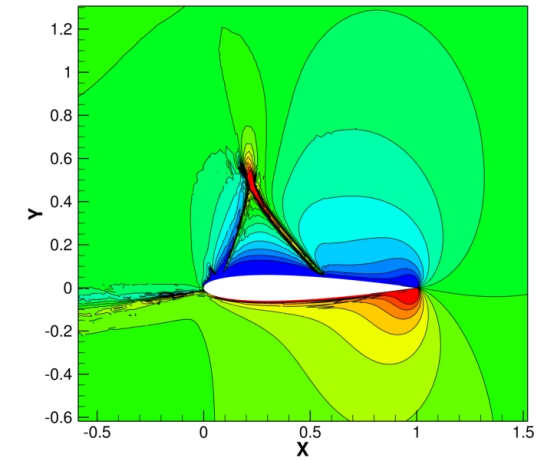
Figure 3.3 shows the lift adjoint contours for an inviscid transonic flow computed with the PDE based artificial viscosity. The flow conditions are $M_\infty = .75$ and $\alpha = 3.5^o$. Notice that even in the triangle shaped region upstream of the shock smooth adjoint contours are obtained. Again the lift shows more sensitivity to the upstream flow field than to the down stream flow field.



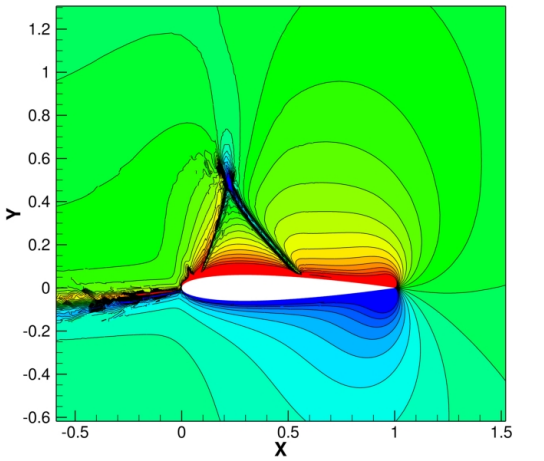
(a) ρ (b) ρu

(c) ρv (d) E_t

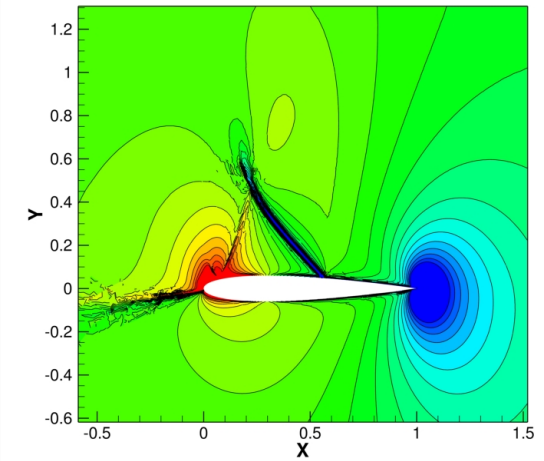
Figure 3.2: Drag adjoint contours for laminar flow over a NACA0012 airfoil at $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$.



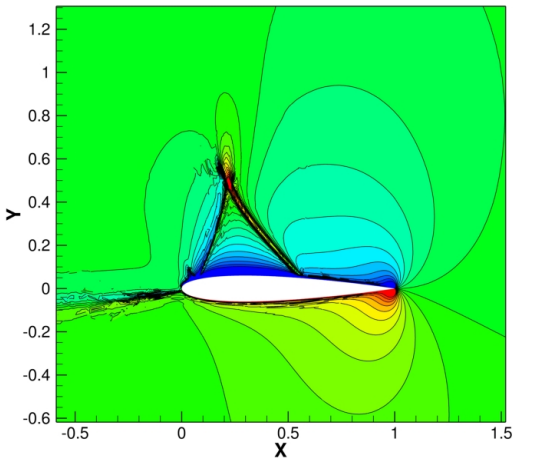
(a) ρ



(b) ρu



(c) ρv



(d) E_t

Figure 3.3: Lift adjoint contours for inviscid transonic flow over a NACA0012 airfoil using PDE-based artificial viscosity, with $M_\infty = .75$, $\alpha = 3.5^o$.

Chapter 4

Solution Methods

An efficient and robust CFD solver requires the consideration of many coupled tasks. One such task that is often overlooked or glossed over is the non-linear solver. In high-order methods research, one often finds explicit time-stepping is used to solve the non-linear equations, even for steady-state flows [36, 38, 75]. While this is easy to program and generally robust, explicit time-stepping is prohibitively slow due to the time-step restriction based on the Courant-Friedrichs-Lewy(CFL) constraint, which becomes more restrictive as the discretization order increases. Fully implicit solvers offer a significantly faster alternative to explicit time-stepping because there is no time-step constraint.

Implicit solution techniques are gaining popularity within the high-order methods research community. References [30, 31, 33, 76–78] pioneered the use of multigrid methods for solving the linear system arising from the application of Newton's method to the flow problem, as well as for solving the non-linear problem directly. As a follow-up to the multigrid solvers, numerical experiments with the application of Krylov methods were conducted in references [15, 57, 79].

In this work, implicit solvers are used exclusively for all test problems. Newton's method is used to solve the non-linear algebraic system of equations resulting from the spatial discretization of the steady-state form of the governing equations in equation (2.2.4). In order to solve the linear problem arising at each Newton iteration, the Generalized Minimum Residual (GMRES) method is employed. Preconditioning is a key component of any GM-

RES method, and this section details the results of various preconditioners. In particular, preconditioning for anisotropic viscous meshes is considered in detail.

High Reynolds number viscous flows form a large part of this work. These flows require highly anisotropic grids that necessitate curving not only the boundary elements in the mesh but several layers of interior elements as well. Anisotropic grids induce significant stiffness into the discrete equations, and one method to alleviate the stiffness is line-implicit relaxation. The line creation and line-implicit relaxation methods are described in detail. To make the solver more efficient, mixed-element meshes are employed using quadrilaterals in the boundary layer and triangles in outer regions. The mixed-element grid generation method is explained, and comparisons between triangular and quadrilateral meshes for boundary layer flows are discussed. This chapter establishes the baseline numerical results of the high-order DG CFD solver.

4.1 Grid Generation and Manipulation

The unstructured meshes in this work are generated using the UMESH2D unstructured mesh generator of reference [80]. UMESH2D is an advancing front delaunay triangulation unstructured mesh generator that generates triangular meshes. The meshes can have highly stretched elements in the boundary layer and wake regions that are suitable for Reynolds Averaged Navier-Stokes (RANS) computations around single and multi-element airfoils. Figure 4.1 shows an example mesh generated for a typical multi-element airfoil configuration. In order to improve transformation computation cost and relieve stretched mesh stiffness, mixed-element meshes with quadrilaterals in the boundary layer are employed.

4.1.1 Mixed-element Meshes

In the finite-volume literature, mixed-element meshes are employed to improve the accuracy of gradient reconstruction in the boundary layer [81]. This is not the case with DG methods, as gradients are computed via the derivatives of the basis functions and do not rely on the neighbor stencil (gradients do however rely on the transformation metrics). However, the use

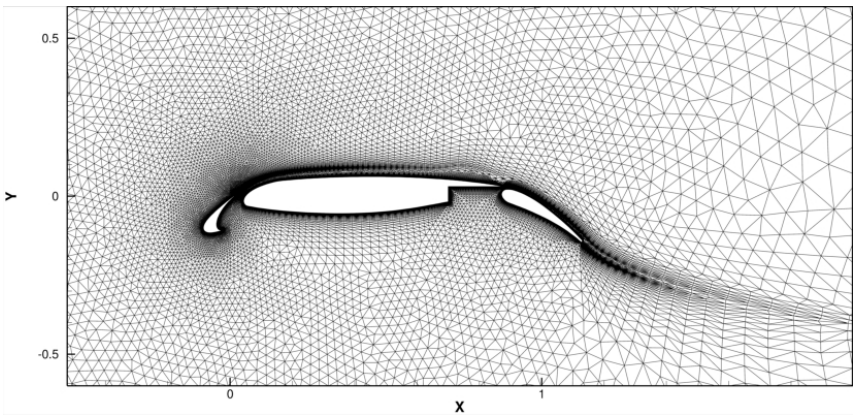


Figure 4.1: Unstructured mesh around multi-element airfoil configuration generated using UMESH2D.

of quadrilateral elements for DG discretizations can still be beneficial, since quadrilaterals enable equivalent accuracy with fewer elements (a single quadrilateral can replace two similar triangles). Figure 4.2 shows the mesh from Figure 4.1 with the boundary layer and wake region triangles merged into quadrilaterals, resulting in an example of a mixed-element mesh employed in this work.

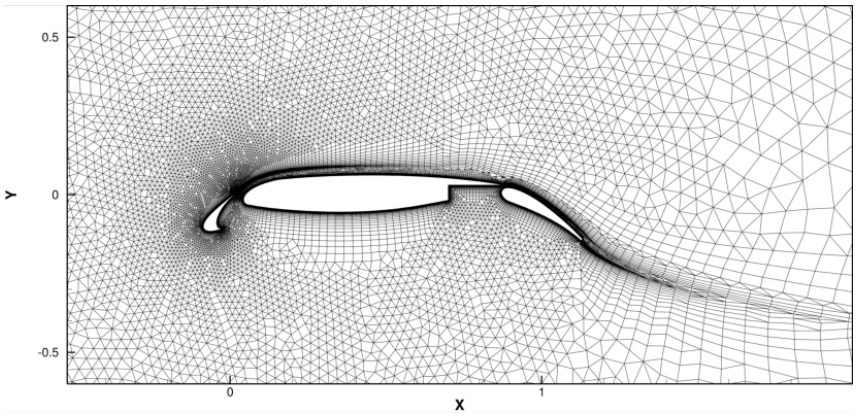


Figure 4.2: Unstructured mixed-element mesh around multi-element airfoil configuration generated using UMESH2D with boundary layer and wake regions merged into quadrilaterals.

Additionally, quadrilateral elements are more flexible for computations involving curved geometries in anisotropic regions of the mesh. In this work, super-parametrically mapped elements are employed, where the element boundaries are mapped geometrically to order $p+1$ for a solution of order p. In the case of anisotropic meshes, this often necessitates the curving of interior elements to avoid inconsistent mesh cross-overs. A variety of strategies

exist for curving the interior elements of anisotropic meshes. For example, references [20, 82] have used elasticity-based node movement schemes. These methods have the advantage that they can produce highly stretched meshes with positive element transform Jacobians everywhere. On the other hand these elasticity-based methods lead to the entire mesh being mapped to higher-order. This is unnecessary as once the mesh has become isotropic and does not touch a curved boundary, straight-sided elements corresponding to linear mappings are sufficient. As such this work employs a simpler approach.

To generate curved elements in anisotropic meshes, mappings of each of the curved boundary faces are generated. Previously formed lines (for the line-implicit Jacobi smoother described subsequently in Section 4.3.2) are then used to copy this curvature straight up the line away from the surface as seen Figure 4.3. This approach ensures that only a small subset of the mesh is mapped to higher than $p = 1$, saving computational cost when computing the transformation quantities. This method does however have a disadvantage, since the element Jacobian can become negative, especially for triangular meshes, which is the result of edge-crossing for highly curved triangles. It is for this reason that quadrilaterals are employed in the boundary layer. Quadrilateral elements can tolerate much higher curvature and aspect-ratios without negative transform Jacobians. In fact, this curvature method has yet to produce a mesh in this study where quadrilaterals have generated negative transformation Jacobians. Employing mixed element meshes for DG discretizations allows for minimal transformation computational cost and the use of highly curved high aspect-ratio elements.

4.1.2 Merging Triangular Meshes

Since UMESH2D generates purely triangular meshes, the mixed-element meshes employed in this work are generated by post-processing the UMESH2D meshes. This post-processing proceedure is known as merging and is designed to recover quadrilaterals in highly anisotropic regions of the mesh. The merging proceedure employed is based on the one proposed in reference [83]. In this work, the distance between triangle circumcenters relative to the Vornoi perimeter is used as the metric to merge triangular elements. The circumcenter is

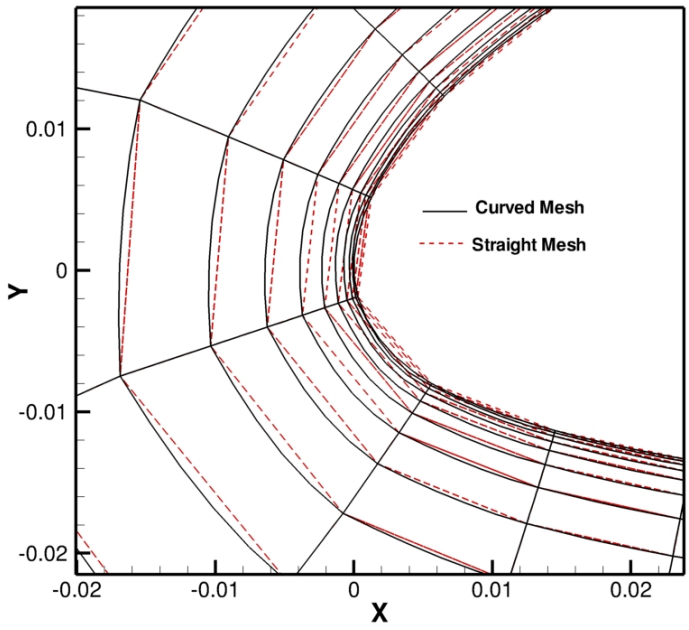


Figure 4.3: Example of stretched curved mesh where several layers of cells near the boundary must be curved in order to prevent edge cross over.

the center of the circumcircle of the triangle, which is the circle that contains all three nodes of the triangle. An example circumcircle and the corresponding circumcenter is shown in Figure 4.4. The Vornoi perimeter is the perimeter of the dual cell centered at the nodes of the mesh. An example Vornoi perimeter is shown in Figure 4.5.

For highly stretched elements that are built in layers, the circumcenters are nearly coincident. This means that the distance between circumcenters along certain edges of the Vornoi diagram will make up a very small fraction of the Vornoi perimeter for the nodes that define the edge. Hence these edges can be removed to form quadrilaterals. The algorithm loops over the grid and removes all the edges where the ratio between circumcenter distance and Vornoi perimeter of either node which defines the edge is less than 0.1. For example, any edge in the isotropic stencil shown in Figure 4.5 would not be removed since each edge makes up an almost equal portion of the Vornoi perimeter. Contrarily, Figure 4.6(a) shows an example stencil where one of the edges is removed by the algorithm, as this edge makes up a small portion of the Vornoi perimeter of each node defining the edge. Figure 4.6(b) shows

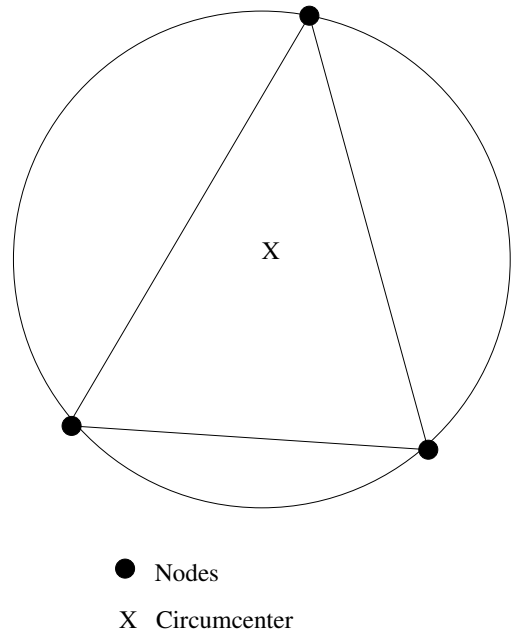


Figure 4.4: Example of a circumcircle for a single triangle.

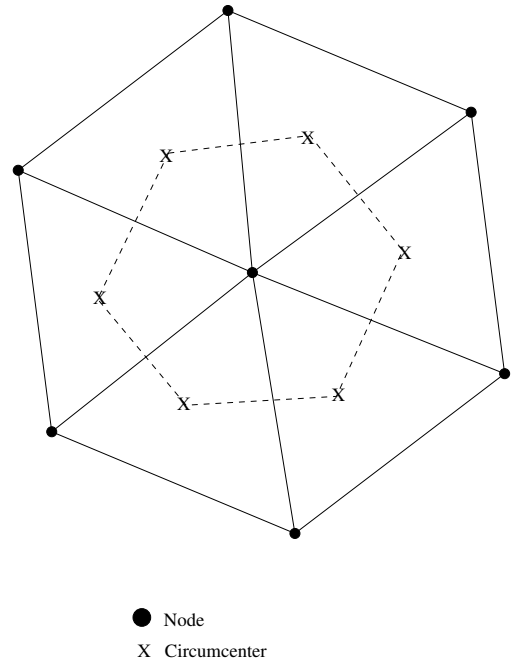
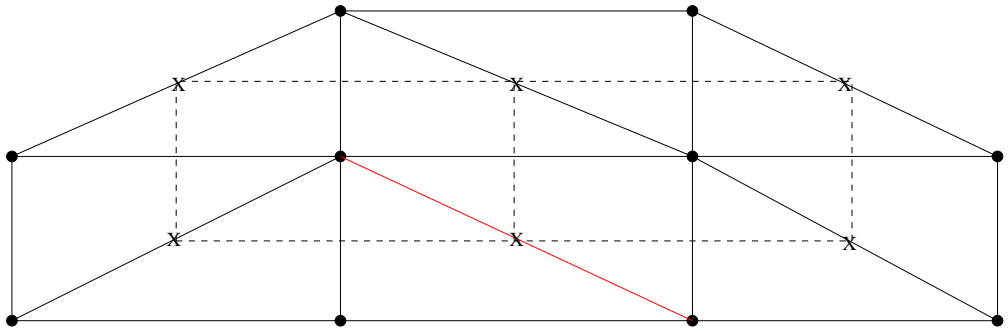


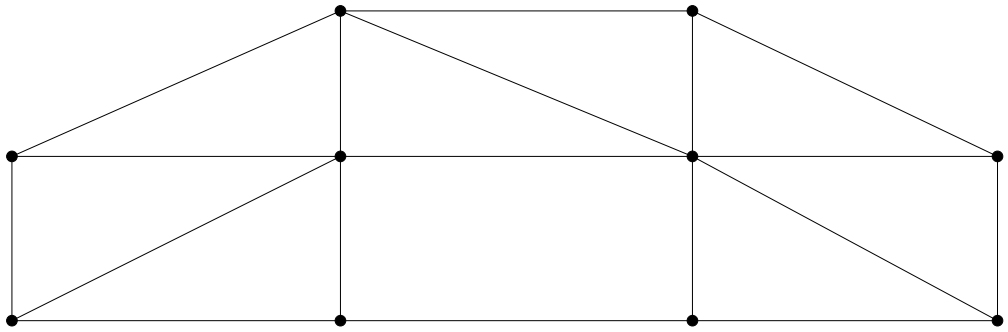
Figure 4.5: Example of a Vornoi perimeter for the node at the center of the stencil.

the results of removing the marked edge in Figure 4.6(a). The algorithm proceeds through each edge in the mesh and removes similar edges, until all such edges have been removed. To ensure that no poorly shaped quadrilaterals are generated the algorithm is not allowed

to remove any edges that would cause an internal angle of the resulting quadrilateral to be greater than 115^o.



(a) Before merge: red edge to be removed



(b) After merge: red edge removed

Figure 4.6: Example of stencil where an edge has a vanishing Vornoi perimeter contribution. The red edge in (a) would is removed via the mering algorithm.

4.2 Implicit Solver Formulation

The steady-state flow equations are solved using a Newton method where the linear system is solved approximately at each Newton step with a GMRES solver. By neglecting the temporal derivative in equation (2.2.3) the system of equations becomes

$$\mathbf{R}_h(\mathbf{u}_h) = 0 \tag{4.2.1}$$

where $\mathbf{R}_h(\mathbf{u}_h)$ is the non-linear residual in equation (2.2.4), which represents the spatial discretization. This set of non-linear equations is solved using Newton's method:

$$\begin{aligned} \left[\frac{\partial \mathbf{R}_h}{\partial \hat{\mathbf{u}}_h}\right]^n \Delta\hat{\mathbf{u}}_h^{n+1} &= -\mathbf{R}_h(\mathbf{u}_h^n) \\ \hat{\mathbf{u}}_h^{n+1} &= \hat{\mathbf{u}}_h^n + \Delta\hat{\mathbf{u}}_h^{n+1} \end{aligned} \tag{4.2.2}$$

Newton's method will diverge if the initial guess is too far from the final solution. Thus the flux Jacobian matrix is augmented with a damping term to increase robustness. The damped Newton iteration is given as:

$$\begin{aligned} \left[\frac{[M]}{\Delta t_e^n} + \frac{\partial \mathbf{R}_h}{\partial \hat{\mathbf{u}}_h}\right]^n \Delta\hat{\mathbf{u}}_h^{n+1} &= -\mathbf{R}_h(\mathbf{u}_h^n) \\ \hat{\mathbf{u}}_h^{n+1} &= \hat{\mathbf{u}}_h^n + \Delta\hat{\mathbf{u}}_h^{n+1} \\ M_{ij_e} &= \int_{\Omega_e} \phi_i \phi_j d\Omega_e \quad \forall e \in \mathcal{T}_h \end{aligned} \tag{4.2.3}$$

where M_{ij_e} is the mass matrix resulting from spatial discretization of the temporal derivative. The mass matrix defined per-element where ϕ_i is the solution basis function described in Section 2.3. Δt_e is an element-wise timestep used as a damping factor:

$$\begin{aligned} CFL &= min\left(CFL_{min}\left(\frac{\parallel \mathbf{R}_h\left(\mathbf{u}_h^0\right) \parallel_2}{\parallel \mathbf{R}_h\left(\mathbf{u}_h^n\right) \parallel_2}\right)^r, CFL_{max}\right) \\ \Delta t_e^n &= \frac{CFL|\Omega_e|}{|\partial\Omega_e|(|\vec{u}_e| + a_e)} \quad \forall e \in \mathcal{T}_h \end{aligned} \tag{4.2.4}$$

where the CFL is the Courant-Friedrichs-Lewy number and a is the sound speed. Due to the block-sparse nature and size of the matrix an iterative method will be used to solve the linear system arising from Newton's method. In this work the CFL is used to aid in selecting stable damping parameters for startup, not as a true time-step restriction that must be satisfied throughout the solution process. Typical settings are

$$CFL_{min} = 1.0$$

$$r = 1.25$$

$$CFL_{max} = 1.0e^{16}$$

which can be adjusted as needed for a particular problem. The absolute minimum CFL_{min} in any test case in this work is $1.0e^{-2}$ and $CFL_{max} = 1.0e^{16}$ for all test cases.

4.3 Linear Solvers

The matrix arising from Newton's methods consists of a sparse block matrix. Since the matrix is quite large, iterative methods become the method of choice for inverting the linear system at each Newton iteration. Consider the linear system $[A]\,x = b$ where $[A], x, b$ are short hand for the damped flux Jacobian, linear solution update and right hand side of equation (4.2.3) respectively. One way to derive iterative methods is to consider splitting the matrix $[A]$. Nastase and Mavriplis [31] review some approximations and splittings of the flux Jacobian. For this work three splittings are employed, namely the so called linearized element Jacobi splitting [31], line-implicit Jacobi splitting and line-implicit colored Gauss-Seidel splitting. Consider the linear system $[A]\,x = b$ and let $[A]$ be split as $[A] = [M] + [N]$. Thus for iteration k one can write the update to x as

$$x^{k+1} = (1-\omega)x^k + \omega\,[M]^{-1}\left(b - [N]\,x^k\right) \tag{4.3.1}$$

where ω is an under relaxation factor $\omega \in (0, 1]$. The forms of $[M]$ and $[N]$ give different iterative methods.

In this work three splittings are considered. In the first, $[M]$ is taken as the block diagonal of the full Jacobian matrix and $[N]$ corresponds to all the block off-diagonals, which yields the so-called linearized element Jacobi (LEJ) [31] scheme. The second splitting yields the line-implicit Jacobi(LIJ) scheme where $[M]$ is now taken as the part of the flux Jacobian corresponding to the diagonal and off-diagonal blocks contained in a set of lines drawn through the highly stretched anisotropic regions of the mesh, and $[N]$ represents all remaining off-diagonal block elements. Note that in regions of isotropic cells (lines containing a single element) this splitting reverts to the LEJ splitting since the line length reduces to a single element. The final splitting is the line-implicit colored Gauss-Seidel(CGS) splitting, which uses the same lines as the LIJ splitting but treats the off-diagonal entries contained in $[N]$ in a different manner. Any of these splittings can be used as a solver by itself. However, reference [57] has shown that they are better utilized as smoothers for multigrid methods, as preconditioners, or as part of preconditioners for Krylov subspace methods (GMRES).

4.3.1 Linearized Element Jacobi (LEJ)

The linearized-element Jacobi splitting consists of the following

$$\begin{aligned} [M] &= [D] \\ [N] &= [O] \\ x^{k+1} &= (1-\omega)\, x^k + \omega\, [D]^{-1} \left(b - [O]\, x^k\right) \end{aligned} \tag{4.3.2}$$

where $[D]$ represents the diagonal block and $[O]$ represents the off diagonal blocks of the Jacobian matrix on the left hand side of equation (4.2.3). Thus only the diagonal block is factorized and inverted. In this case direct LU factorization is employed with forward/backward substitution to give the effect of $[D]^{-1}$. The LEJ smoother is especially simple to construct but does not provide adequate convergence rates on highly stretched meshes. In order to over come this a more involved splitting is devised.

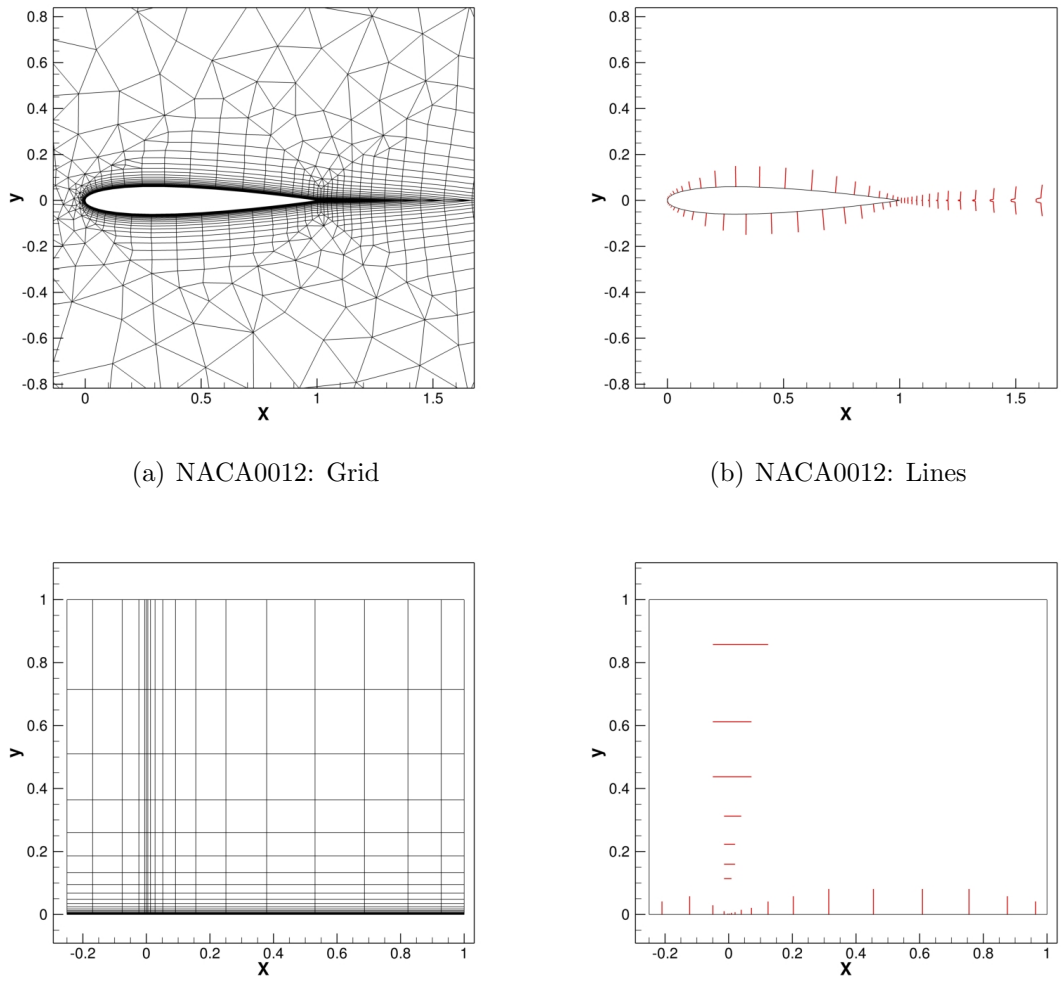
4.3.2 Line-Implicit Jacobi (LIJ)

In order to maintain fast convergence rates on anisotropic meshes a line-implicit smoother can be used [84]. With this in mind, a line creation algorithm and line-implicit Jacobi smoother have been devised and implemented to enable efficient solution techniques on anisotropic meshes.

Line creation is accomplished using a two pass approach. Since the anisotropic cells are used to capture boundary layers, anisotropic regions are found attached to the boundary faces of the mesh that are solid surfaces. The line creation algorithm begins by considering all the boundary faces in the mesh attached to solid surfaces. Lines that originate from the solid surfaces are formed by taking each surface face and computing the corresponding normal vector. Then the angle between the surface normal and all other faces attached to the boundary element are computed. If any of the angles are less then 10 degrees then that face is added to the line and the neighboring element across that face becomes the element to be searched. The processes is repeated until all the current element's face normals make an angle with the surface normal greater than 10 degrees. This is repeated for each boundary face in the mesh that is a solid surface.

Once the lines attached to the solid surfaces are formed there can still be areas in the mesh that are highly anisotropic (i.e. wake regions). Lines in these regions are created using a weighted graph algorithm [85], which has been slightly modified for cell-centered rather then node-centered discretizations. This algorithm assigns a weight to each graph edge (in this case a graph edge is a line connecting two cell-centers). This edge-weight is taken as the inverse of the graph edge length. The ratio of maximum to average edge-weight is precomputed for each cell in the mesh. The cells are then ordered according to this ratio and stored as a heap-list. The top element in the heap is chosen as the starting point for a line, provided the element is not already part of a line. The most strongly connected neighbor is added to the line provided that it is not already part of a line and that the ratio of maximum to minimum connection strength is greater then β ($\beta = 3$ for all cases). The element that was just added to the line now becomes the current element. The line is terminated when the current element's ratio of maximum to minimum connection strength violates the threshold on β. As these lines are not attached to a surface the process is repeated for the original seed element with the second most strongly connected neighbor and proceeding as before, adding to the same line. Figure 4.7(a) and Figure 4.7(b) shows the grid and corresponding lines created around a NACA0012 airfoil. Figure 4.7(c) and Figure 4.7(d) show the lines created on a flat plate geometry of zero thickness. These two examples demonstrate the ability of the line creation algorithm to find lines through all the anisotropic regions of the mesh. Examination of Figure 4.7(b) shows that the combined algorithm has generated both lines connected to the surface and and in the wake. Note that for the flat plate mesh in Figure 4.7(d) the weighted graph algorithm has formed lines in the convective direction off the solid surface. This is due to the cell clustering at the plate leading edge. In fact, if these lines are not created the convergence rate of the solver degrades significantly, showing that lines are needed in highly anisotropic regions of the mesh regardless of the physical phenomena present in those regions.

Once the lines have been created the line-implicit smoother is straight-forward to implement. Implicit lines form a block tri-diagonal matrix for each line. Each row of this block tri-diagonal matrix contains the block diagonal matrix of an element and at least one



(a) NACA0012: Grid

(b) NACA0012: Lines

(c) Flat plate: Grid

(d) Flat plate: Lines

Figure 4.7: Illustration of lines used in line-implicit Jacobi smoother, as well as the grids from which the lines are generated.

(usually two) block off-diagonal matrices. This is how $[M]$ and $[N]$ in equation (4.3.1) are defined. $[M]$ contains all the block matrices in a line and $[N]$ contains those block off-diagonal matrices that are not in a line. Figure 4.8 shows an example of three lines drawn vertically through a sample stencil.

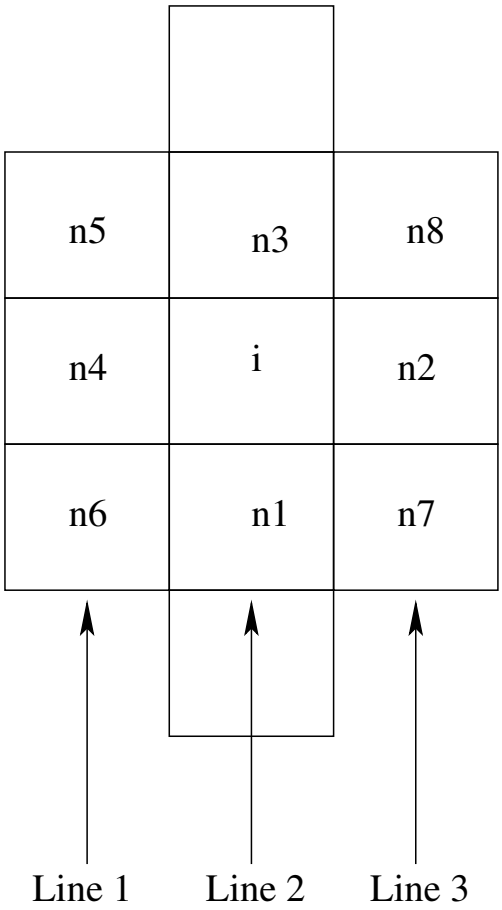


Figure 4.8: Sample line-implicit stencil with three lines defined vertically in the figure.

A sample of the split matrices $[M]$ and $[N]$ for lines in Figure 4.8 are given as:

$$[M] = \begin{bmatrix} \ddots & & & & \\ \left[\frac{\partial \mathbf{R}_{hn1}}{\partial \mathbf{u}_{hn1}}\right] & \left[\frac{\partial \mathbf{R}_{hn1}}{\partial \mathbf{u}_{hi}}\right] & 0 & \dots & \\ \left[\frac{\partial \mathbf{R}_{hi}}{\partial \mathbf{u}_{hn1}}\right] & \left[\frac{\partial \mathbf{R}_{hi}}{\partial \mathbf{u}_{hi}}\right] & \left[\frac{\partial \mathbf{R}_{hi}}{\partial \mathbf{u}_{hn3}}\right] & 0 & \dots \\ 0 & \left[\frac{\partial \mathbf{R}_{hn3}}{\partial \mathbf{u}_{hi}}\right] & \left[\frac{\partial \mathbf{R}_{hn3}}{\partial \mathbf{u}_{hn3}}\right] & 0 & \dots \\ & & & \ddots & \end{bmatrix} \tag{4.3.3}$$

$$[N] = \begin{bmatrix} \ddots & & & & & & & & & \\ \dots & 0 & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{hn1}}{\partial \mathbf{u}_{hn6}}\right] & \left[\frac{\partial \mathbf{R}_{hn1}}{\partial \mathbf{u}_{hn7}}\right] & 0 & \dots \\ \dots & 0 & \left[\frac{\partial \mathbf{R}_{hi}}{\partial \mathbf{u}_{hn2}}\right] & 0 & \left[\frac{\partial \mathbf{R}_{hi}}{\partial \mathbf{u}_{hn4}}\right] & 0 & 0 & 0 & 0 & \dots \\ \dots & 0 & 0 & 0 & 0 & \left[\frac{\partial \mathbf{R}_{hn3}}{\partial \mathbf{u}_{hn5}}\right] & 0 & 0 & \left[\frac{\partial \mathbf{R}_{hn3}}{\partial \mathbf{u}_{hn8}}\right] & \dots \\ & & & & & & & & & \ddots \end{bmatrix} \tag{4.3.4}$$

In Figure 4.8 the stencil is centered about cell i with neighbors $n1$ to $n8$. The $[M]$ matrix is made up of lines connecting the diagonal of an element with two of the element's off-diagonal blocks as shown in equations equation (4.3.3). The $[N]$ are the remaining off-diagonal elements not in $[M]$ for each element in a line.

Using this matrix splitting the linear system is now solved using a block variant of the Thomas algorithm [9]. Consider a block tri-diagonal matrix given as

$$\left[\left[A_i\right], \left[B_i\right], \left[C_i\right]\right] x_i = r_i \quad \forall i = 1, N \tag{4.3.5}$$

where $[A_i]$ is a block matrix in the i^{th} row of the full matrix and x_i is a block vector in the i^{th} row of the full vector. The r_i vector on the right hand side is a place holder for all the right hand side terms in the LIJ splitting. The block Thomas factorization algorithm is given by Algorithm (1). For a given number of iterations the factorization is performed only once

Algorithm 1 :Block Thomas Factorization

$\left[B_1'\right] = \left[B_1\right]$
$\left[B_1'\right] = LU\left(\left[B_1'\right]\right)$
for $j = 2, N$ **do**
  $\left[B_j'\right] = \left[B_j\right] - \left[A_j\right]\left(\left[B_{j-1}'\right]^{-1}\left[C_{j-1}\right]\right)$
  $\left[B_j'\right] = LU\left(\left[B_j'\right]\right)$
end for

and reused for several iterations of the solver. Algorithm (2) shows the solution algorithm utilized for the LIJ splitting.

The line-implicit smoother is used to ensure that the convergence rate observed for isotropic meshes is maintained for similar anisotropic meshes. To verify that this is the case, the LIJ smoother is implemented within a linear multigrid algorithm and used to solve Poisson's equation. Two meshes each of which contain $N = 1,102$ elements are used. The first mesh is made of isotropic evenly spaced triangles and the second is made of anisotropic stretched triangles(maximum aspect ratio = 26912 : 1). The results of solving Poisson's equation with a $p = 3$ DG discretization using linear multigrid are shown in Table 4.1, where the average rate refers to the average decrease in the linear system residual $\parallel b - [A]\, x \parallel_2$

Algorithm 2 :Block Thomas Iteration

for $k = 1, N_{iteration}$ **do**
  $y_1 = \left[B_1^{'}\right]^{-1} r_1$
  for $j = 2, N-1$ **do**
    $y_j = \left[B_j^{'}\right]^{-1} \left(r_j - \left[A_j\right] y_{j-1}\right)$
  end for
  $y_N = \left[B_N^{'}\right]^{-1} \left(r_N - \left[A_N\right] y_{N-1}\right)$
  $x_N = y_N$
  for $j = N-1, 1$ **do**
    $x_j = y_j - \left(\left[B_j^{'}\right]^{-1} \left[C_j\right]\right) x_{j+1}$
  end for
end for

Table 4.1: Comparison of convergence rates for Poisson problem, using $N = 1,102$ elements with and without stretching for $p = 3$.

Mesh/Smoother	Average rate
Isotropic/LEJ	.49
Anisotropic/LEJ	.98
Anisotropic/LIJ	.55

over the convergence history. Mathematically this is given by

$$\text{Average rate} = \sum_{n=1}^{N_{iter}} \left(\frac{\left| \| \, b - [A] \, x \, \|_2^n - \| \, b - [A] \, x \, \|_2^{n+1} \right|}{\| \, b - [A] \, \|_2^n} \right) \tag{4.3.6}$$

where N_{iter} is the total number of linear iterations utilized to solve the problem. These results very clearly show that without the line-implicit smoother the convergence rate drops significantly. On the other hand, if the line-implicit smoother is employed the convergence rate is almost unchanged from the isotropic case.

Additionally a line-implicit colored Gauss-Seidel relaxation method has also been implemented as a stand alone solver and single-level preconditioner to GMRES.

4.3.3 Colored Gauss-Seidel(CGS)

In order to apply a Gauss-Seidel relaxation strategy in parallel the mesh must be "colored". Coloring is a process by which elements in the mesh are numbered in groups that are independent of one another. Gauss-Seidel smoothing is then performed over the groups. Algorithm

Algorithm 3 :Colored Gauss-Seidel Algorithm

for $k = 1, N_{iteration}$ **do**
  for $c = 1, N_{color}$ **do**
    Line smooth over lines on boundary: $x_c^{k+1} = [M]^{-1}\left(b - [N]\, x_c^k\right)$
    Post non-blocking send and receives
    Line smooth over all interior lines: $x_c^{k+1} = [M]^{-1}\left(b - [N_{color>=c}]\, x_c^k - [N_{color<c}]\, x_c^{k+1}\right)$
    MPI WAITALL
    Solution is now updated correctly on each processor
  end for
end for

(3) illustrates how the colored Gauss-Seidel(CGS) algorithm is used to solve the linear system in a parallel computing framework using the message passing interface (MPI). One should immediately notice that special attention has been paid to overlapping communication with computation. Additionally, CGS allows for a fully parallel Gauss-Seidel method provided the colors are created such that no element within one color requires data from the same color.

Lines are colored via a greedy algorithm, which loops over all the lines formed for the line-implicit solver assigning color integers to the lines. The algorithm begins by looping over lines and assigning colors one at a time to lines that have not been colored and whose neighbors have not been flagged due to a coloring. If a line has not been colored and has no neighbor, that has been tagged from a coloring, then the line is given a color(in the form of an integer). Once a line is given a color the line and the neighbors of the line are tagged and not allowed to be colored again by this same color. Once all lines have been checked the color index increases and the process loops over the lines again for the next color proceeding as before. This is shown by Algorithm (4). The algorithm continues to create new colors until all the lines in the mesh have been assigned a color. This results in lines that are colored such that no color has lines that are neighbors in the mesh. An example of a colored mesh with lines that are of length 1 and higher is given in Figure 4.9. In Figure 4.9 there are several elements that have the same color. These are elements in the same line.

Algorithm 4 :Coloring

N_{line}-$colored = 0$
$color2line\,(:) = 0$
$N_{color} = 0$
while N_{line}-$colored < n_{line}$ **do**
  $line\text{-}touch\,(:) = 0$
  $N_{color} = N_{color} + 1$
  for $l = 1, N_{lines}$ **do**
    $color = .true.$
    if $color2line(l) == 0$ **then**
      Check neighbors *line-touch*, if $line\text{-}touch\,(neighbor) > 0$ then $color = .false.$
    end if
    if $color == .true.$ **then**
      $color2line\,(l) = N_{color}$
      $line\text{-}touch\,(l) = 1$
      N_{line}-$colored = N_{line}$-$colored + 1$
      For all neighbors set $line\text{-}touch\,(l) = 1$
    end if
  end for
end while

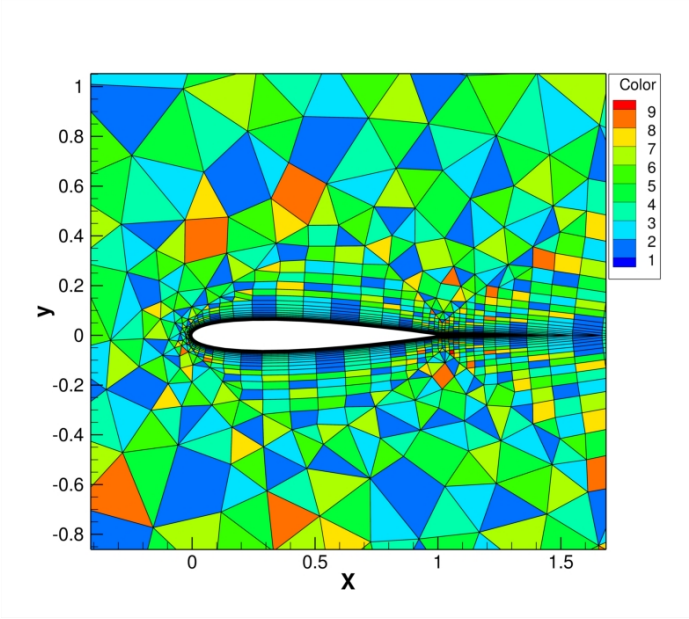


Figure 4.9: Colors generated for a mixed-element stretched mesh around a NACA0012 airfoil.

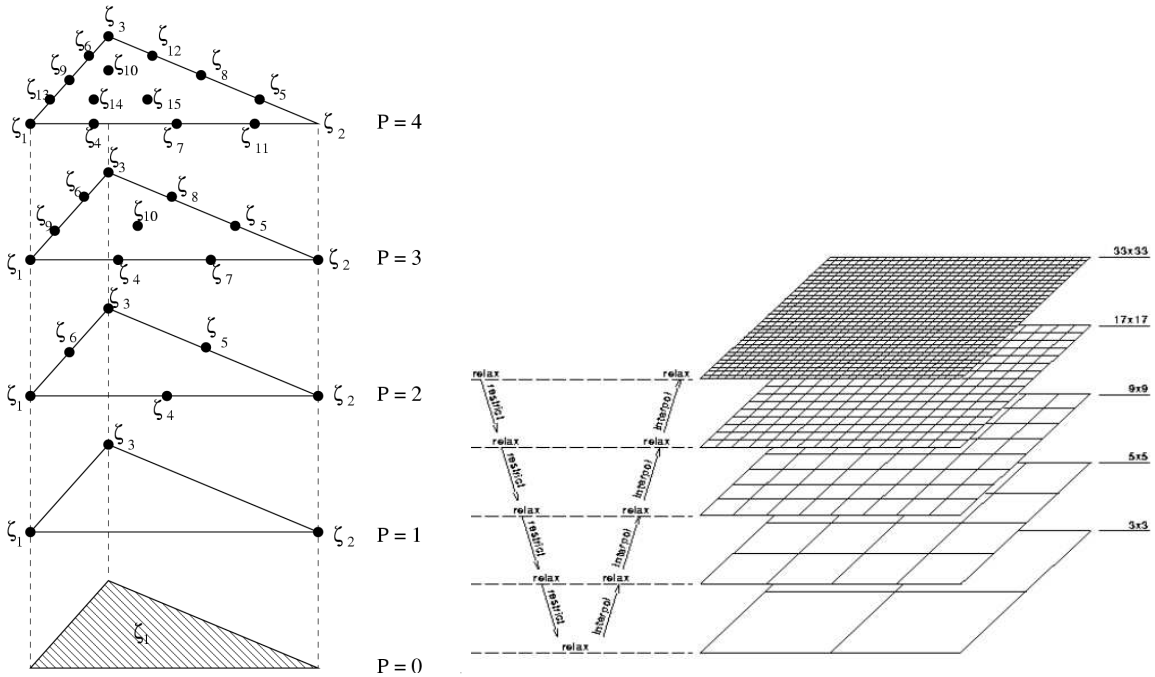
Once the mesh is colored, the Gauss-Seidel iteration proceeds similarly to the LIJ iteration. The factorization is the same but the $[N]$ matrix multiplies the most up to date neighbor information. For some elements the vector x is at the k^{th} iteration and for other elements the vector x is at the $(k+1)^{th}$ iteration. The elements that belong to a color value less than c use information from iteration $k+1$ and color values greater than c use information from iteration k as seen in Algorithm (3), where k is the linear iteration index and c is the color index.

4.4 Multigrid Methods

Multigrid methods are known as efficient techniques for accelerating convergence to steady-state for both linear and non-linear problems [86, 87], and can be applied with a suitable existing relaxation technique. The rapid convergence property relies on an efficient reduction of the solution error on a nested sequence of coarse grids.

4.4.1 The *hp*-Multigrid Approach

The spectral multigrid approach [30, 76] is based on the same concepts as a traditional *h*-multigrid method, but makes use of "coarser" levels which are constructed by reducing the order of accuracy of the discretization, rather than using physically coarser grids with fewer elements. Thus, all grid levels contain the same number of elements, which alleviates the need to perform complex interpolation between grid levels and/or to implement agglomeration-type procedures [87]. Furthermore, the formulation of the interpolation operators, between fine and coarse grid levels, is greatly simplified when a hierarchical basis set is employed for the solution approximation. The main advantage of a hierarchical basis set is that the lower-order basis functions are a subset of the higher-order basis (*i.e.* hierarchical) and the *restriction* and *prolongation* operators become simple projection operators into a lower- and higher-order space, respectively [88]. Therefore their formulation is obtained by a simple deletion or augmentation of the basis set. The *restriction* from fine level to coarse level is obtained by disregarding the higher-order modal coefficients and transferring the values of



(a) p-Multigrid (b) h-Multigrid

Figure 4.10: Illustration of hp-multigrid levels.

the low-order modal coefficients exactly. Similarly, the *prolongation* from coarse to fine levels is obtained by setting the high-order modes to zero and injecting the values of the low-order coefficients exactly. Figure 4.10(a) shows various p-levels for a reference triangular element.

Multigrid strategies are based on a recursive application of a two-level solution mechanism, where the second (coarser) grid is solved exactly, and used to accelerate the solution on the finer grid [86]. The exact solution of the coarse grid problem at each multigrid cycle is most often prohibitively expensive, therefore the recursive application of multigrid to solve the coarse grid problem offers the preferred approach for minimizing the computational cost of the multigrid cycle, thus resulting in a complete sequence of coarser grids. For spectral (p)-multigrid methods, the recursive application of lower-order discretizations ends with the $p = 0$ discretization on the same grid as the fine level problem.

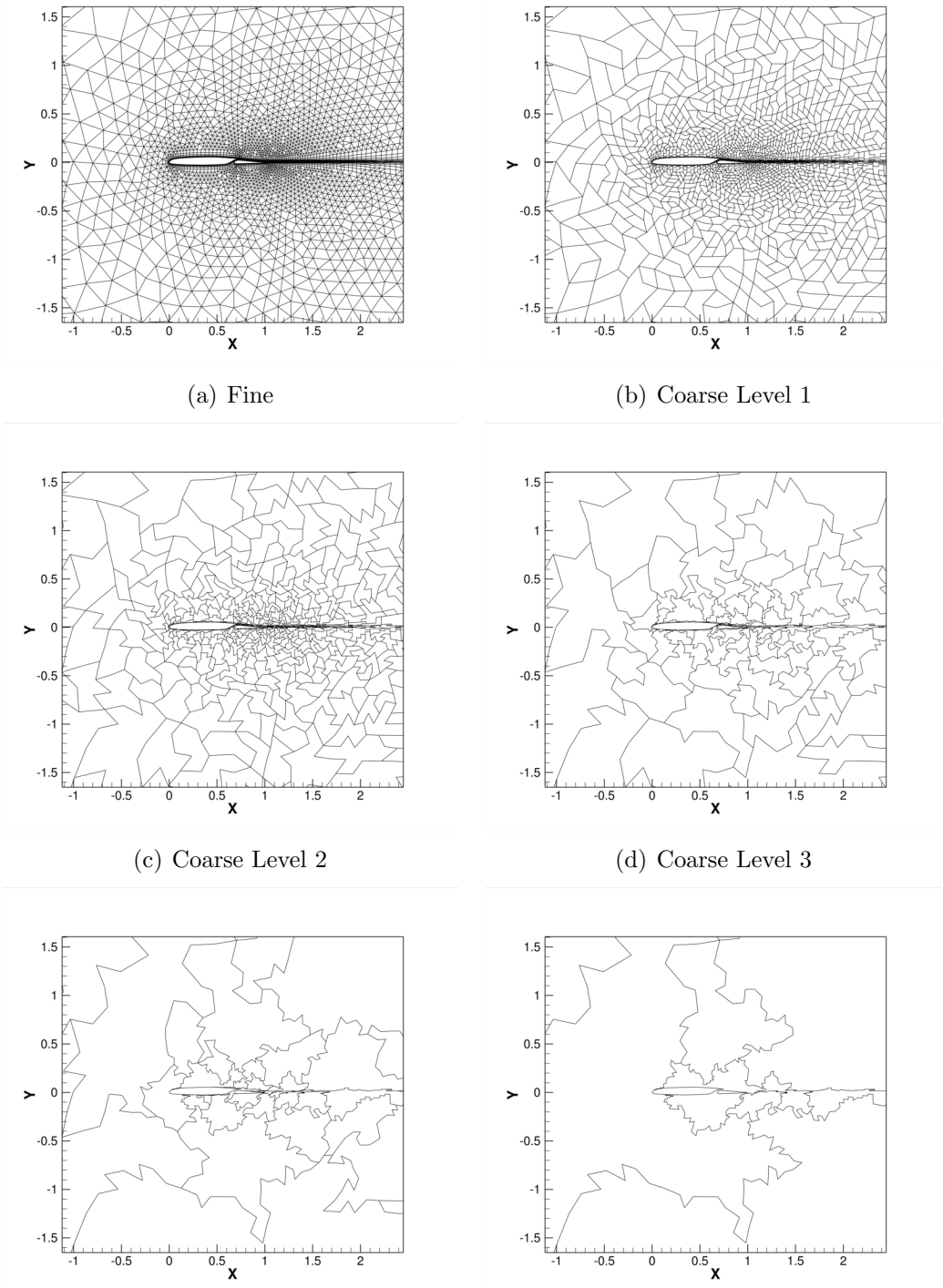
For relatively fine meshes, the (exact) solution of this $p = 0$ problem at each multigrid cycle can become expensive, and may impede the h-independence property of the multigrid strategy. The $p = 0$ problem can either be solved approximately by employing the same

number of smoothing cycles on this level as on the finer p levels, or the $p = 0$ problem can be solved more accurately by performing a larger number of smoothing cycles at each visit to this coarsest level. In either case, the convergence efficiency will be compromised, either due to inadequate coarse level convergence, or to excessive coarse level solution cost. An alternative is to employ an h-multigrid procedure to solve the coarse level problem at each multigrid cycle. In this scenario, the p-multigrid scheme reverts to an agglomeration multigrid scheme once the $p = 0$ level has been reached, making use of a complete sequence of physically coarser agglomerated grids, thus the designation hp-multigrid. Agglomeration multigrid methods make use of an automatically generated sequence of coarser level meshes, formed by merging together neighboring fine grid elements, using a graph algorithm. Figure 4.10(b) shows various h-levels for a Cartesian quadrilateral mesh. Figure 4.11 shows a sequence of coarse meshes generated using the graph agglomeration approach. First-order accurate ($p = 0$) agglomeration multigrid methods for unstructured meshes are well established and deliver near optimal grid independent convergence rates [89].

The hp-multigrid procedure [31,33] has been shown to result in an h- and p-independent solution strategy for high-order accurate discontinuous Galerkin discretizations of the Euler equations, in both two- and three-dimensions [31, 33]. For robustness it is important to augment the resulting multi-level hp-multigrid scheme with a full multigrid (FMG) technique, in order to provide a good initial guess for the fine level problem. Moreover, the use of FMG is critically important in the case of the linear multigrid scheme for it is known that the Newton iteration will diverge if the initial guess is not close enough to the final solution. In the hp-multigrid approach, the solution process begins at the coarsest grid level ($p = 0$), using all the h-levels available, and ends at the fine level where all the p- and h-levels are used to advance the solution to the desired accuracy, as depicted in Figure 4.12.

4.5 Newton-GMRES

Krylov Subspace Methods represent an alternative technique for solving the linear system given by Newton's method. Since the linear system is non-symmetric, the Generalized Min-



(a) Fine

(b) Coarse Level 1

(c) Coarse Level 2

(d) Coarse Level 3

(e) Coarse Level 4

(f) Coarse Level 5

Figure 4.11: Coarse mesh levels generated via agglomeration of a slotted airfoil mesh.

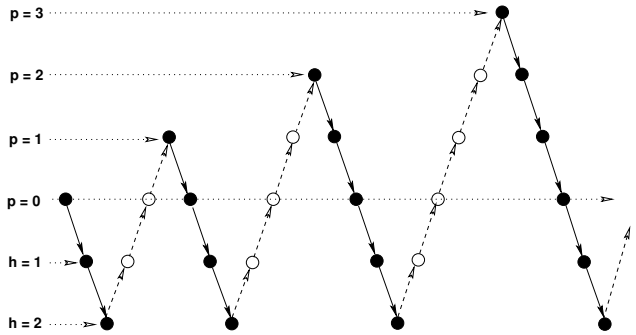


Figure 4.12: Illustration of full *hp*-multigrid (FMG) levels for $p = 3$ and $h = 2$ (− restriction, - - prolongation, • smoothing, ∘ update).

imal Residual method (GMRES) is employed. The convergence of GMRES is often accelerated using a preconditioner. In this work, the previously described linear multigrid algorithm and colored Gauss-Seidel(CGS) are used as preconditioners. Reference [57] has shown that this combination is a nearly optimal method for isotropic meshes, and showed some promising preliminary results for triangular anisotropic meshes. In this work the GMRES bases solver of reference [57] is extended to hybrid anisotropic meshes.

The linear-system in equation (4.2.3) is again written as

$$[A]\, x = b \tag{4.5.1}$$

where $[A]$ is the flux Jacobian, x is the Newton update and b is the negative of the flow residual. GMRES seeks to minimize the L_2 norm of the residual of the linear system($r = [A]x - b$) over the space $span\{r_0, [A]r_0, [A]^2 r_0, \ldots, A^{k-1} r_0\}$. There are many excellent texts on Krylov subspace methods [90–92] where details on the theory of Krylov subspace methods are presented. For a right-preconditioned system one obtains the following

$$[A]\,[P]^{-1}\, y = b, \quad x = [P]^{-1}\, y \tag{4.5.2}$$

where $[P]$ is the preconditioner. In this case the Krylov subspace is
$span\{r_0, [A]\,[P]^{-1}\, r_0, ([A]\,[P]^{-1})^2 r_0, \ldots, ([A]\,[P]^{-1})^{k-1} r_0\}$. To compute the Krylov subspace basis a linear system of the form

$$[P]\, z = q \tag{4.5.3}$$

must be solved, where q and z are the Krylov and preconditioned Krylov vectors respectively. This equation is solved approximately using a few cycles of multigrid. Using multigrid as the

preconditioner yields an algorithm denoted as multigrid preconditioned GMRES (MGPC-GMRES). Note that the preconditioning matrix is the full flux Jacobian and is never fully inverted. Also note that GMRES requires additional storage for the Krylov basis. However, it is expected that the multigrid preconditioner will be very effective so the size of the Krylov basis will remain small. The details of the GMRES solver are given in Algorithm (5).

In this work the GGS linear solver is also used a preconditioner for the GMRES method. Using the CGS linear solver has no impact on the GMRES algorithm beyond changing how the approximate solution to equation (4.5.3) is generated.

4.6 Robustness Enhancement via Local Order-Reduction

One of the principal concerns when developing DG solvers is the robustness of the solver. The author and others [20] have noted that under mesh-resolved regions of the flow can adversely impact the convergence of the flow solver. If regions of smooth extrema are under resolved, the smooth extrema can become non-smooth extrema. When these extrema become non-smooth the DG discretization may produce unphysical oscillations, which may lead to solver failure(as is shown in Section: 4.7.3).

A simple way of dealing with under-mesh resolved regions is to refine the mesh in the under resolved region, thus causing the extrema to become smooth again. However, the merits of direct limitation are also worth investigating. To decrease the oscillations that result from these under-resolved areas, one can add additional diffusion to the equations to smooth out the solution. There are a variety of ways to add additional diffusion to the discrete equations. For example, artificial diffusion can be added, similar to the way shock waves are treated in this work (Section 2.7). Yet another approach, is to recognize that as the jumps between elements increase, so does the artificial diffusion associated with the upwinding. In order to increase the jumps between elements a local element order-reduction technique based on the element resolution detector developed in reference [34] is used. Decreasing the discretization order of the element increases the inter-element jumps and thus the artificial diffusion of the flux function. At the same time, reducing the discretization order p reduces

Algorithm 5 :Multigrid preconditioned fGRMES algorithm

N = # of Krylov vectors, M = # of linear iterations
Given x_1, compute $r_1 = b - [A]\, x_1$
i = 1
while $i <= M$ .and. $\|r_i\| > tol$ **do**
  set $q_1 = \frac{r_1}{\|r_1\|}$
  set $\zeta(:) = 0$
  set $\zeta(1) = \|r_1\|$
  n = 1
  while $n <= N$ .and. $res > tol$ **do**
    Solve $[P]\, z_n = q_n$ using linear multigrid
    $[Z(:,n)]_i = z_n$
    $v = [A]\, z_n$
    % Compute new Krylov basis vector via Arnoldi procedure
    for $j = 1, n$ **do**
      $h(j,n) = v \cdot q_j$
      $v = v - h(j,n) q_j$
    end for
    $h(n+1,n) = \|v\|$
    $q_{n+1} = v / h(n+1,n)$
    % Make the upper Hessenberg matrix upper triangular via givens rotations
    for $j = 1, n-1$ **do**
      $temp = cs(j)h(j,n) + sn(j)h(j+1,n)$
      $h(j+1,n) = -sn(j)h(j,n) + cs(j)h(j+1,n)$
      $h(j,n) = temp$
    end for
    % Compute the n^{th} givens rotation matrix(a very standard approach given in) [93]
    call get_rot(h(n,n), h(n + 1, n), cs(n), sn(n))
    $h(n,n) = cs(n)h(n,n) + sn(n)h(n+1,n)$
    $temp = cs(n)zeta(n)$
    $\zeta(n+1) = -sn(n)zeta(n)$
    $\zeta(n) = temp$
    res = $\frac{|\zeta(n+1)|}{\|b\|}$
  end while
  solve $[h]\, y = \zeta$
  $x_{i+1} = x_i + [Z_i]\, y_i$
  % where the n^{th} column of $[Z_i]$ is z_n
  $i = i + 1$
  $r_i = b - [A]\, x_i$
end while

the oscillations because p is lower. The detector for an element is given in equation (2.7.2) with density used as the detection quantity instead of pressure. The value of s_k for each element k in the mesh is computed and if $log_{10}(s_k)$ is greater then -4 for an element, the discretization order of that element is reduced to $p_k = max(p_k - 1, 1)$ (here $p = 1$ is the lowest possible value to retain all of the discrete viscous terms). This approach is used here to test the concept of treating these kinds of under resolved regions as though they were flow discontinuities. Furthermore, this highly simplified approach gives an example of how a limiter would behave if one were applied.

4.7 Numerical Results

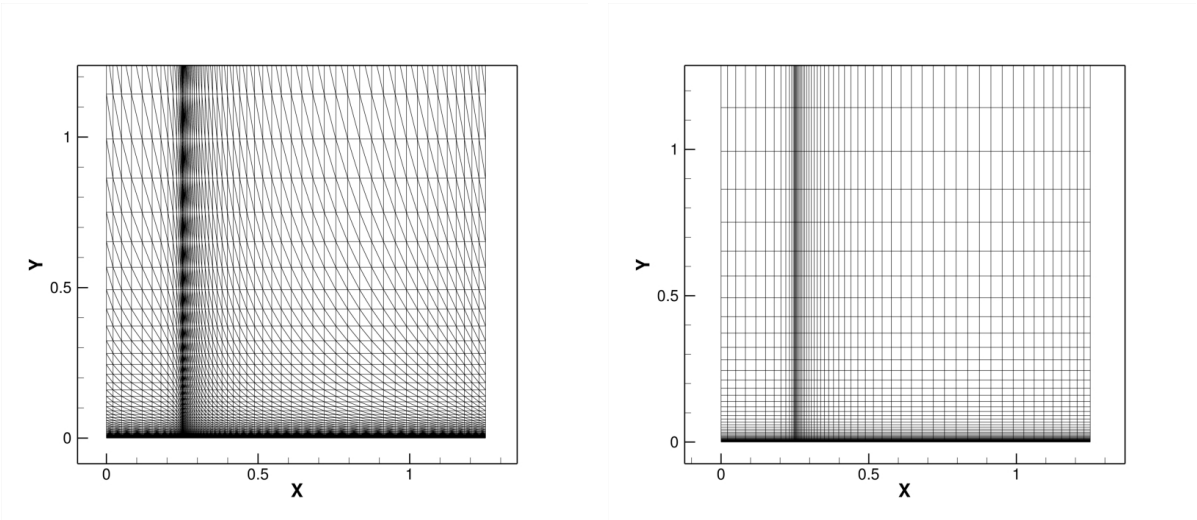
The flow solver is validated and tested using three separate test cases, a laminar flat plate, a NACA0012 airfoil, and a two-element airfoil. The laminar flat plate test case is used to validate the Navier-Stokes terms in the flow solver. The efficiency of the MGPC-GMRES solver is tested using a NACA0012 airfoil and a two-element airfoil.

4.7.1 Laminar Flat-Plate

The laminar viscous flow solver is validated using a zero pressure gradient flat plate boundary layer at the following conditions $M_\infty = .1$, $\alpha = 0^o$, and and Reynolds number based on chord $Re = 200,000$. This flow is computed using discretization orders $p = 0$ through $p = 3$ for both a triangular mesh and a quadrilateral mesh with $N = 6,372$ and $N = 3,186$ elements respectively. The results are compared against the well known Blasius boundary layer solution. The triangular mesh is built from the quadrilateral mesh with each quadrilateral divided into two triangles, in order to maintain the spacing normal to the wall and the cell height to length ratios between the two meshes. The meshes for this case are shown in Figures 4.13(a) and 4.13(b).

Figures 4.14(a) through 4.15(b) show the computed u-velocity profiles with the Blasius solution plotted as a reference. The computed velocity profiles agree very well for discretization orders $p = 1$ and greater. For $p = 0$ the velocity profile is extremely inaccurate. This is

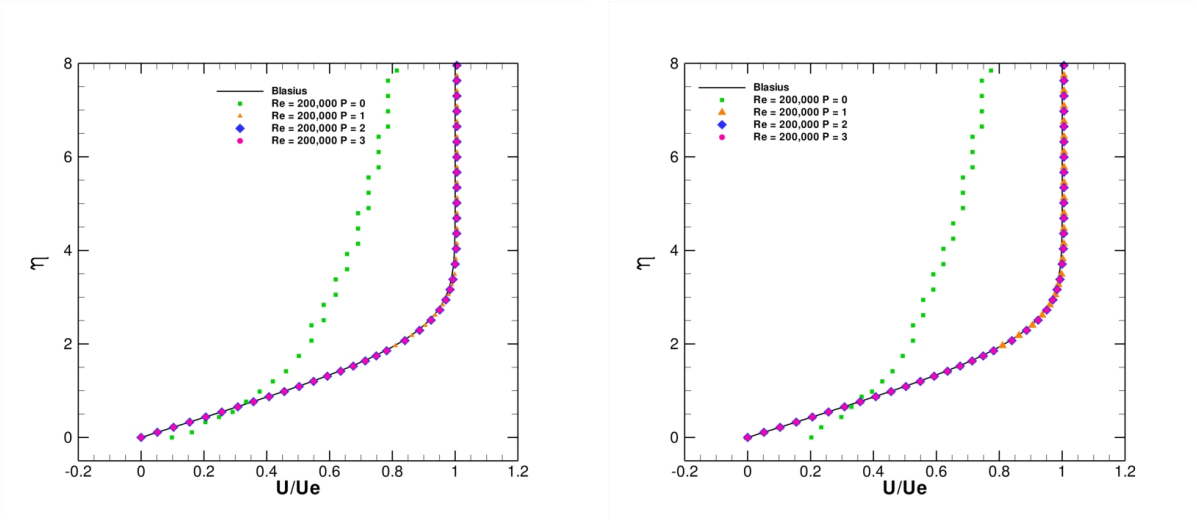
due to the fact that the DG discretization of the viscous operator reduces to an inconsistent edge-only approximation for the viscous terms. These results are included simply to show that $p = 0$ is not sufficient to fully capture the viscous operator. The u-velocity profile matches the Blasius profile for all discretization orders higher than $p = 1$. The v-velocity profile is the harder of the two profiles to capture accurately. The second-order result on the triangular mesh shows a considerable deviation from the Blasius solution in this profile, while higher-order results show improved agreement. Additionally, note that above $p = 1$, there is essentially no difference in the agreement between the computed velocity profiles on the triangular and quadrilateral meshes. Also note that for $p = 3$ the quadrilateral meshes contains 25% fewer degrees of freedom(DoFs) than the triangular mesh. Therefore, quadrilateral meshes are capable of delivering equivalent or better accuracy compared to self-similar triangular meshes at a lower computational cost.



(a) $N = 6,372$ triangles

(b) $N = 3,186$ quadrilaterals

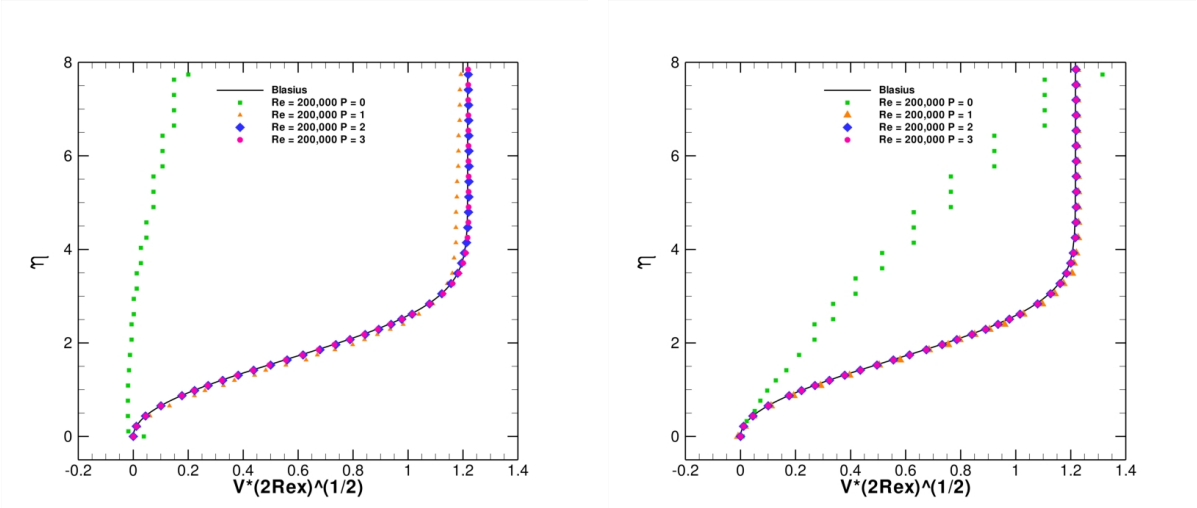
Figure 4.13: Meshes used for computing the laminar flow past a flat plate.



(a) Triangular Mesh

(b) Quadrilateral Mesh

Figure 4.14: Laminar flat plate u-velocity profile computed using a DG discretization for $p = 0$ to $p = 3$ compared to the Blasius u-velocity profile.



(a) Triangular mesh

(b) Quadrilateral mesh

Figure 4.15: Laminar flat plate v-velocity profile computed using a DG discretization for $p = 0$ to $p = 3$ compared to the Blasius v-velocity profile.

4.7.2 NACA0012 Airfoil

The effectiveness and efficiency of the MGPC-GMRES algorithm is studied using two test cases. The first is a NACA0012 airfoil at $M_\infty = .5$, $\alpha = 0^o$, and $Re = 5,000$, which is used to verify the h-independence and p-independence of the solver for purely triangular meshes. In each case an iteration represents one Newton iteration. For each Newton iteration the MGPC-GRMES solver convergences the linear system obtained via Newton's method to the tolerance given by equation (4.7.1) or until 15 Krylov vectors are generated.

$$\begin{aligned} tol &= max\left(min\left(\frac{\|\mathbf{b}\|_{L_2}}{2.5^{(n-nc-p)}}, .1\|\mathbf{b}\|_{L_2}\right), 1.0e-14\right) \\ nc &= 8 + 2\,(p-1) \end{aligned} \tag{4.7.1}$$

In equation (4.7.1) p is the discretization order, $\mathbf{b}$ is the right-hand side of the linear system, and n is the Newton iteration number. For each Krylov vector the preconditioning system given by equation (4.5.3) is solved with four cycles of the linear multigrid solver using the LIJ solver as the multigrid smoother.

Two purely triangular meshes are used for this case and contain $N = 2,250$ and $N = 7,750$ triangles, with a maximum aspect ratio of 82:1 and 238:1, and with average line lengths of 10 and 25 cells respectively. Due to the variance in aspect ratio, a slight h-dependence is expected, since aspect-ratio can affect the stiffness of the problem. Figures 4.17(a) and 4.17(b) show the respective convergence rate obtained and demonstrate relatively h-independent and p-independent behavior. These figures demonstrate that the MGPC-GMRES solver gives nearly h-independent and p-independent results, despite the difference in aspect-ratio. To ensure that the above results are also valid for mixed-element anisotropic meshes, the NACA0012 airfoil test case is computed on a mixed-element anisotropic mesh containing $N = 4,964$ elements (Figure 4.18(a)) with a maximum aspect-ratio of 65:1 for DG discretization orders $p = 1$ to $p = 4$. Figure 4.18(b) depicts the convergence history using this mixed-element mesh. This figure demonstrates that the MGPC-GMRES solver retains its p-independence on mixed-element anisotropic meshes.

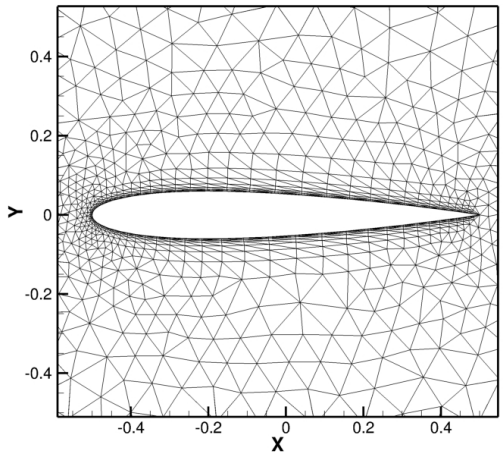
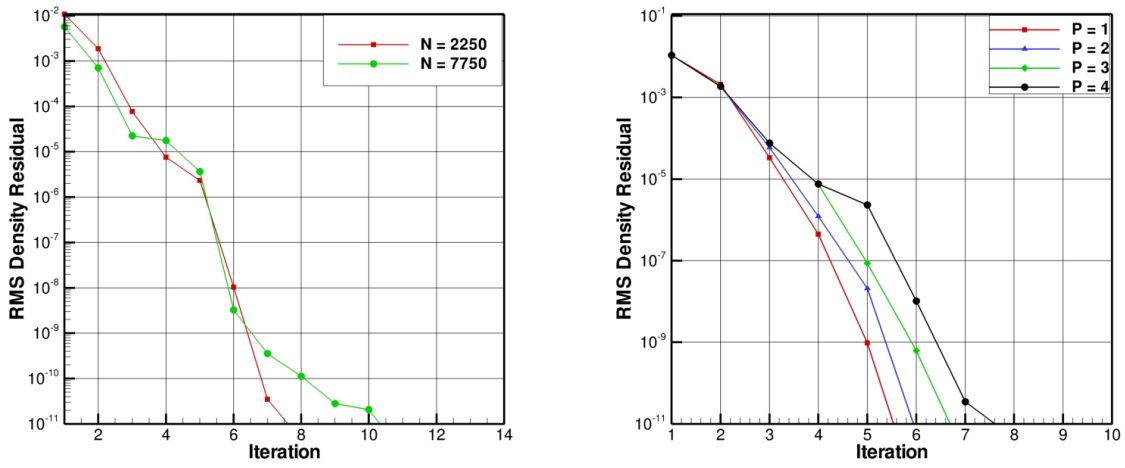


Figure 4.16: NACA0012 mesh with $N = 2,250$ elements.



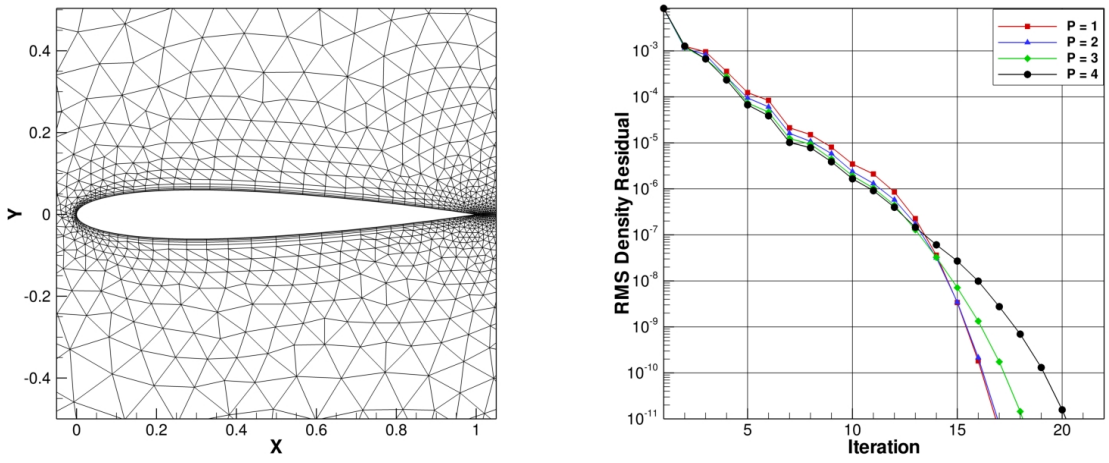
(a) h-independence

(b) p-independence

Figure 4.17: MGPC-GMRES convergence rates for the solution of the NACA0012 case on two meshes with $N = 2,250$ and $N = 7,750$ elements and for DG discretization orders $p = 1$ to $p = 4$.

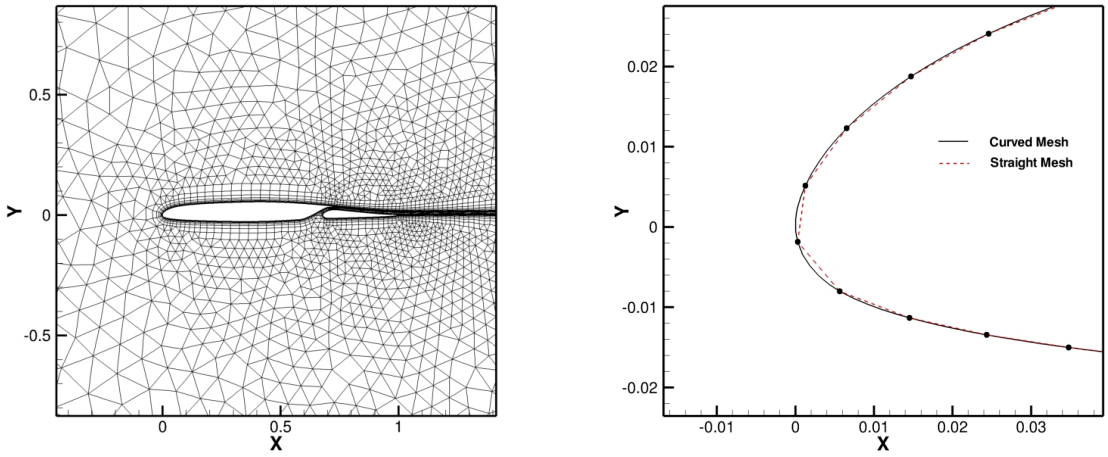
4.7.3 Two-Element Airfoil

The third test case consists of the flow over a two-element airfoil depicted in Figure 4.19(a). The flow conditions for this case are $M_\infty = .3$, $\alpha = 0^o$, and $Re = 5,000$. Solutions for discretization orders $p = 1$ through $p = 4$ are computed using a mesh of $N = 7,902$ elements (5,266 triangles and 2,636 quadrilaterals), with a maximum aspect ratio of 265:1 and using



(a) Mixed-element mesh with $N = 4,964$ elements (b) p-independence

Figure 4.18: Mixed-element mesh containing $N = 4,964$ elements (220 quadrilaterals and 4,744 triangles) and MGPC-GMRES convergence rates for the solution of the NACA0012 case for DG discretization orders $p = 1$ to $p = 4$.



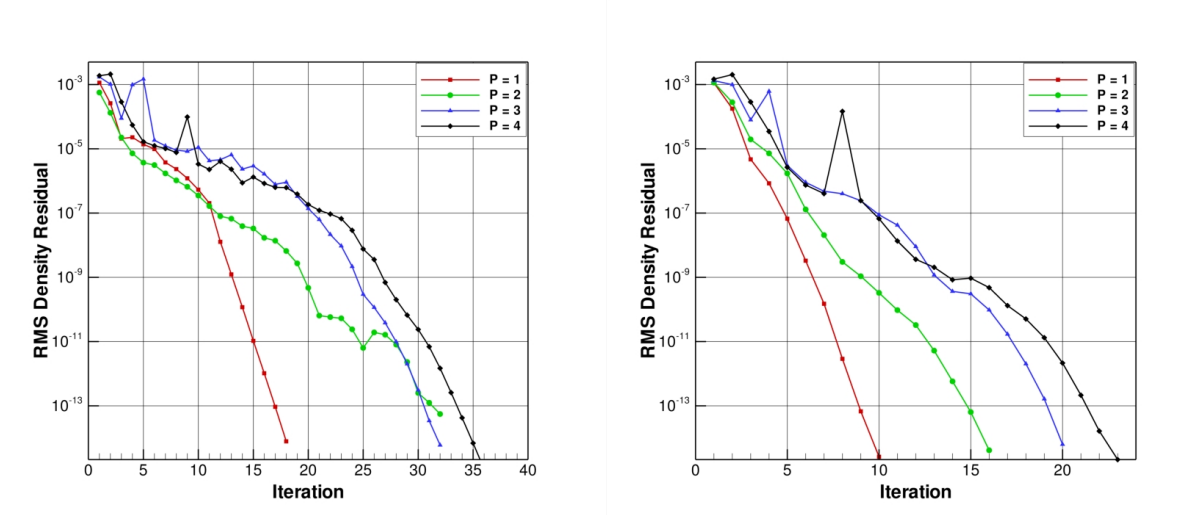
(a) Mesh with $N = 7,902$ elements (b) Surface curvature close-up

Figure 4.19: Two-element airfoil mixed element mesh used for solver comparison and local order-reduction robustness improvement.

252 lines with an average length of 15 cells per line. This two-element airfoil test case is used to compare the performance of linear multigrid vs. MGPC-GMRES linear solvers. Additionally, a CPU time comparison between the LEJ smoother and LIJ smoother in the context

of the linear multigrid solver is conducted. The presented results are generated in a manner different from the NACA0012 airfoil test case. To ensure accurate CPU time comparisons both the linear multigrid and MGPC-GMRES solvers are run such that each Newton iteration consisted of ten multigrid cycles. For the linear multigrid solver this constitutes ten multigrid cycles while for the MGPC-GMRES solver this constitutes five Krylov vectors with two cycles of multigrid for preconditioning each Krylov vector.

Solver Comparison



(a) Linear multigrid

(b) MGPC-GMRES

Figure 4.20: Convergence rate of the two-element airfoil case for orders $p = 1$ to $p = 4$ using linear multigrid and MGPC-GMRES.

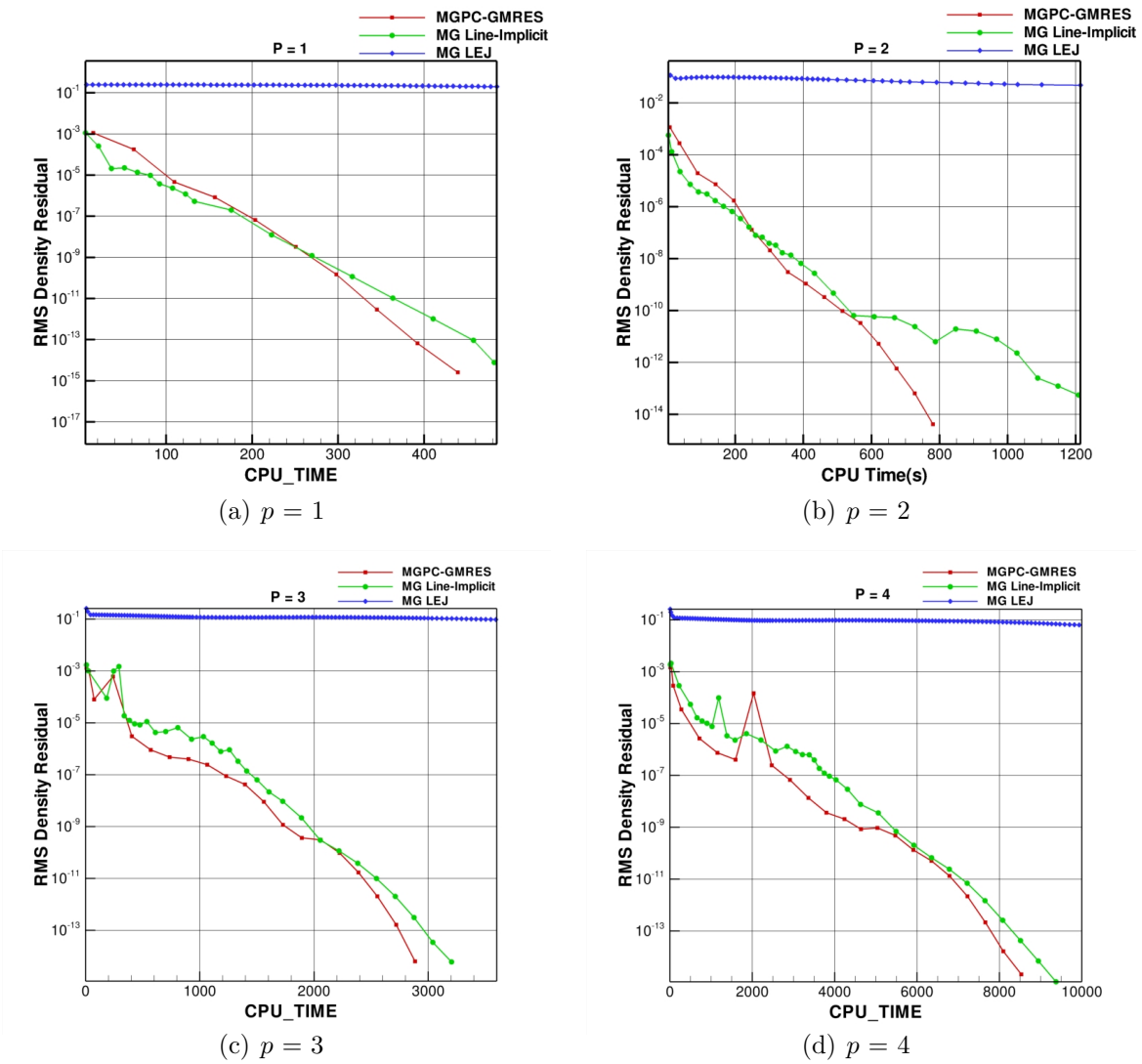
First consider the p-independence of the solvers. Figure 4.20(a) shows the convergence of such a solver for $p = 1$ through $p = 4$. The number of iterations required to solve $p = 2$ and higher is nearly twice that of $p = 1$, indicating a relatively strong p-dependence between $p = 1$ and higher-order ($p \geq 2$) solutions. The same problem is solved using MGPC-GMRES and the convergence is depicted in Figure 4.20(b). The p-independence for the MGPC-GMRES solver shows a slight improvement over that of the linear multigrid solver because the convergence history of the $p = 2$ solution is closer to that of the $p = 1$ solution. The overall p-independence is not improved very much for several reasons. The

$p = 3$ and $p = 4$ solutions required the application of the local order-reduction technique, which combined with the fixed small number of linear iterations caused a degradation of the p-independence compared to the NACA0012 case. Furthermore, each solver has a similar number of rises in the residual over the convergence history, which affects the MGPC-GMRES solver more severely because a single Newton iteration represents double the percentage of the total number of Newton iterations compared to the linear multigrid solver. However, the NACA0012 airfoil results and the $p = 2$ result here show very good p-independence, leading one to conclude that the results for $p = 3$ and $p = 4$ cases are affected by the under-resolved stagnation point and/or the application of local order-reduction without which the $p = 3$ and $p = 4$ solutions could not converge.

Though the MGPC-GMRES solver does not obtain textbook p-independence, this solver is preferred to the linear multigrid solver because it is faster in terms of overall CPU time. A CPU time comparison between the linear multigrid and MGPC-GMRES solvers is shown in Figures 4.21(a) through 4.21(d). These figures clearly demonstrate that the MGPC-GMRES solver is faster in terms of overall CPU time than the linear multigrid solver. On average the MGPC-GMRES solver is about 15% faster than the linear multigrid solver with the $p = 2$ case showing the most dramatic improvement. Figures 4.21(a) through 4.21(d) also illustrate the iterative convergence obtained from these cases when the line-implicit Jacobi smoother is replaced by the LEJ smoother in the linear multigrid solver. Clearly, not employing the line-implicit smoother yields a very slow and inefficient solver for DG discretizations on anisotropic meshes.

Robustness Enhancement

This case is also used to test the local order-reduction technique for enhanced robustness. This particular mesh demonstrates the under mesh-resolved phenomena discussion in Section 4.6. Near the stagnation point at the leading edge of main airfoil the mesh is too coarse to smoothly capture the high density and pressure gradients. As a result, for discretization orders $p = 3$ and $p = 4$ the extrema become non-smooth resulting in solver failure. To alleviate this problem the local order-reduction technique outlined in Section 4.6 is applied.



(a) $p = 1$ (b) $p = 2$

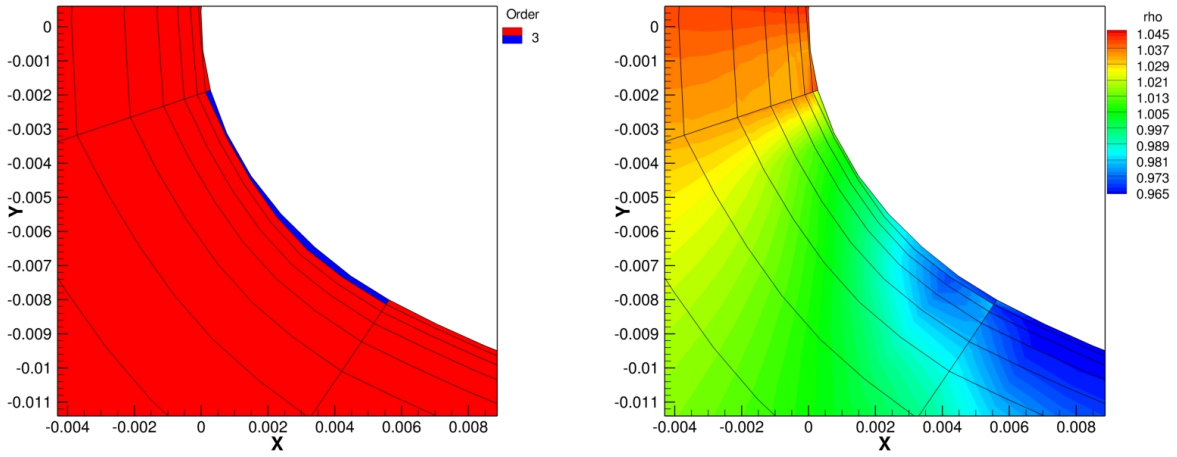
(c) $p = 3$ (d) $p = 4$

Figure 4.21: CPU time comparison between MGPC-GMRES (line-implicit Jacobi smoother), linear multigrid (line-implicit Jacobi smoother) and linear multigrid (LEJ smoother) for solution of the two-element airfoil case for using DG discretization orders $p = 1$ to $p = 4$

By applying this technique, the solver is able to generate solutions for discretization orders $p = 3$ and $p = 4$ where it was previously unable to do so. Figures 4.22(a) and 4.22(b) show the cells and density contours where the discretization order is reduced for the $p = 4$ case. The under mesh-resolved cell has large density variations across it and the density contours show some oscillations. One should make note of the rises in the residual convergence history for the $p = 3$ and $p = 4$ case. These rises correspond to the iterations where the discretization

order is being reduced. In these test cases density is the quantity used in the detector given by equation (2.7.2).

In conclusion, the MGPC-GMRES solver represents a significant improvement over linear multigrid methods for DG discretizations due to the faster overall CPU time. Figures 4.23(a) and 4.23(b) show the Mach contours and stream lines for a $p = 2$ solution and Figures 4.24(a) and 4.24(b) show the same for a $p = 4$ solution. Note that the wake is very well preserved due to the anisotropic mesh and high-order accurate solutions. This case also demonstrates the enhanced robustness given by the local order-reduction technique, allowing the $p = 3$ and $p = 4$ solutions to converge. While local order-reduction is not the most elegant technique it has proven sufficient this problem and warrants further study for resolving other more challenging phenomena.



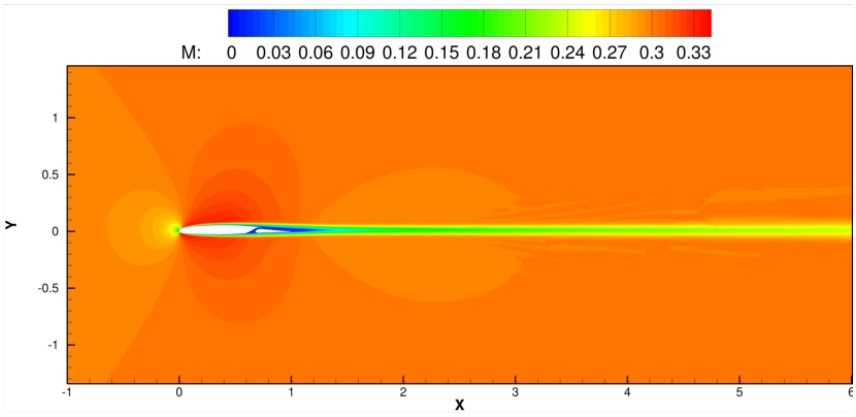
(a) Element discretization order

(b) Density contours

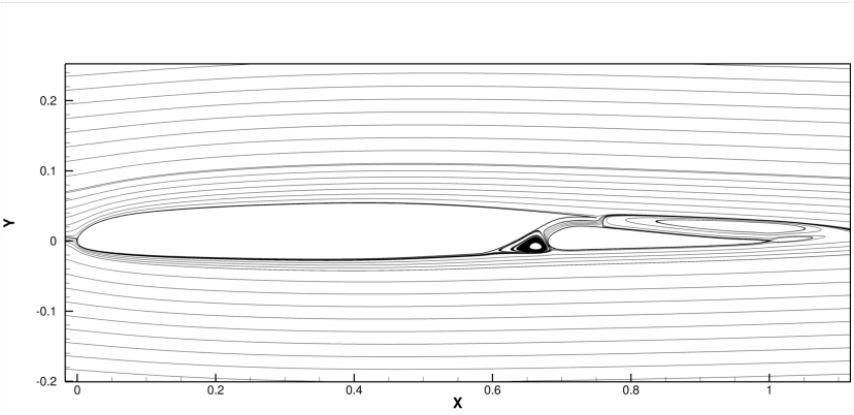
Figure 4.22: Close-up of the two-element airfoil under mesh-resolved leading edge showing the reduced-order cell and density contours for $p = 4$.

4.8 Summary

This work has investigated and developed efficient solution strategies for steady-state viscous flows using high-order DG discretizations in the presence of curved, hybrid-element



(a) Mach contours

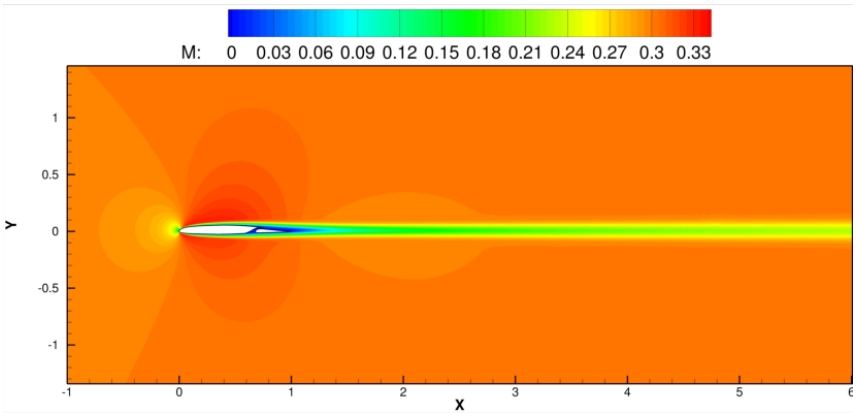


(b) Streamlines

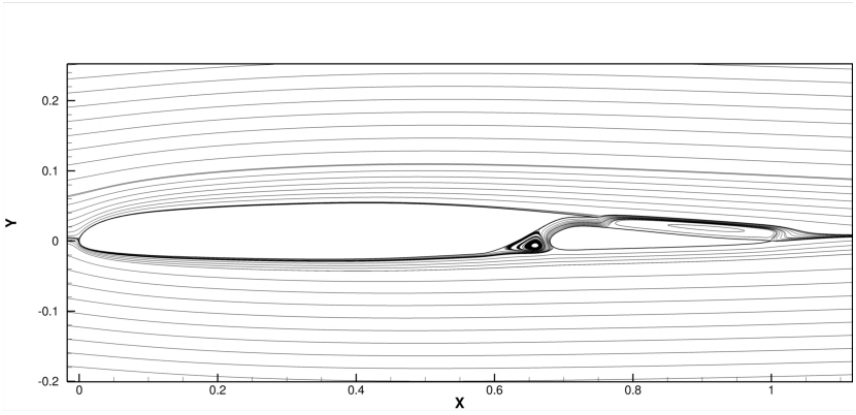
Figure 4.23: Illustration of computed solution using DG for the laminar viscous flow over the two-element airfoil at $M_\infty = .3$, $\alpha = 0^o$, and $Re = 5,000$ for $p = 2$.

anisotropic unstructured meshes. The solution strategy is based on the *hp*-multigrid approach previously developed for inviscid flows. A line creation algorithm and line-implicit Jacobi smoother were developed and implemented to enable efficient solution techniques on anisotropic meshes. Further improvement has been demonstrated through the use of a preconditioned Newton-Krylov technique.

Two-dimensional results are presented for a flat plate boundary layer, flow over a NACA0012 airfoil and a two-element airfoil. Current results demonstrate convergence rates which are nearly independent of the degree of anisotropy, discretization order p and level of mesh resolution (h). It was shown that the MGPC-GMRES algorithm outperforms standard multigrid techniques both in terms of CPU time and optimality. The MGPC-GMRES solver



(a) Mach contours



(b) Streamlines

Figure 4.24: Illustration of computed solution using DG for the laminar viscous flow over the two-element airfoil at $M_\infty = .3$, $\alpha = 0^o$, and $Re = 5,000$ for $p = 4$.

exhibits better h- and p-independence when the linear problem is fully converged as in the NACA0012 airfoil case. However, this might prove impractical for more realistic cases (*i.e* the two-element airfoil). A more practical scenario is to limit the number of multigrid cycles per Newton iteration. In this case MGPC-GMRES has been shown to be faster than linear multigrid in terms of CPU time for the same number of multigrid cycles.

One of the principal issues with high-order discretizations is the robustness of these methods in dealing with discontinuous, non-smooth, or under-resolved features. In this work, an element-wise order-reduction technique for addressing such robustness issues has been investigated. By detecting and reducing the discretization order of "troubled" cells for the two-element airfoil problem, the solver was able to overcome the under resolved

stagnation point. For this case, the issue stems from an under mesh-resolved leading edge where gradients in flow quantities are high. By under resolving the geometry at the leading edge, what should be smooth extrema become non-smooth, resulting in the solver creating overshoots that then cause failure. By locally reducing the discretization order, the increased dissipation allows the solver to handle the overshoot and converge with this under-resolved region. Based on these results, a robustness enhancement method that can increase the mesh resolution until the extrema again become smooth should be investigated. One possible strategy is apply an adaptive proceedure that will increase mesh resolution before applying high-order discretizations to under mesh resolved regions.

Chapter 5

Goal Oriented Adaptive Mesh Refinement

The use of goal oriented adaptive mesh refinement has become a prevalent technique in CFD [16–18, 20, 43, 45, 47, 71]. Rather than perform adaptive mesh refinement based on local indicators or flow features, goal-oriented mesh refinement targets a specified output of the simulation. However, goal oriented adaptive mesh refinement requires *a posteriori* error estimates in the output of interest. Herein the output error estimates will be derived as will the use of these estimates to drive mesh adaptation. The adaptive method employed is known as *hp*-adaptation and combines grid refinement (i.e. *h*-refinement) and order enrichment (i.e. *p*-enrichment) into a single adaptive method.

Due to the mixed-element meshes employed, *h*-refinement is more complicated than simplex(triangle) only refinement. Mixed-element mesh refinement involves hanging nodes which complicate surface integrations and interior element curvature methods. The combination of *h*-refinement and order or *p*-enrichment, denoted as *hp*-adaptation, will be introduced both as a method to enhance solver efficiency and also to enhance solver robustness. To demonstrate both properties, *hp*-adaptation will be applied to several test cases including laminar subsonic flows through supersonic viscous flows. Shock capturing will be accomplished using the piecewise constant artificial viscosity formulation, combined with *hp*-adaptation.

5.1 Motivation

A posteriori error estimation for functional outputs is becoming a mature technique for estimating the contribution of discretization error to simulation outputs [16–18, 44, 45]. These error estimates are based on solutions to the so called adjoint (dual) problem. Such error estimates provide a method to guide adaptive refinement techniques to optimally place degrees of freedom within a mesh. Recently these techniques have been used to perform adaptive mesh refinement (i.e. h-refinement only) in the context of DG discretizations of the RANS equations [20, 43, 46]. Reference [17] has used these techniques for the *hp*-adaptation of high speed shocked flows using a DG discretization of the compressible Euler equations.

In this work, *hp*-adaptation is used to adaptively enrich the discretization order and refine the mesh for DG discretizations of the Navier-Stokes equations on mixed-element meshes, i.e. meshes containing triangles and quadrilaterals. A discrete adjoint formulation is used to obtain the error estimates in the functional of interest. The formulation is based on a discrete adjoint approach using a fully dual (adjoint) consistent discretization. References [20,56,72] have shown numerically that using dual inconsistent discretizations can lead to sub-optimal convergence of the primal solution, whereas using dual consistent or asymptotically dual consistent discretizations leads to optimal convergence($O\left(h^{p+1}\right)$) of the primal problem while producing more accurate adjoint-based error estimates. Furthermore, Section 3.3 has shown that dual consistent discretizations result in the super convergence of the output or functional error.

Discontinuous Galerkin (DG) methods are capable of generating high-order accurate solutions to the Euler and Navier-Stokes equations. However, this is only attained if the solution is smooth. Unfortunately, for aerodynamic applications, solutions are rarely smooth. Non-smooth solutions can result from the expected discontinuities such as shock waves and contact discontinuities as well as from additional sources, which are not covered as thoroughly in the literature. For example, if the leading edge of an airfoil has been discretized with too few cells, oscillations can develop due to under-resolution of smooth phenomena and cause the solver to fail as shown in Section 4.7.3.

A unique property of DG discretizations is that the order of accuracy and the number

of degrees of freedom (DoFs) are coupled (hereafter referred to as resolution coupling). This is in contrast to traditional finite-volume or finite-difference techniques that instead rely on extended stencils to increase the discretization order. Resolution coupling is the reason that developing limiters for DG discretization is more difficult than for traditional CFD methods. Furthermore resolution coupling poses a rather serious drawback for computing discontinuous solutions with DG.

While for non-adaptive techniques the coupling between discretization order and the number of degrees of freedom poses a rather serious problem for limiting the solution as shown numerically in reference [15], the coupling is actually an advantage in the context of an adaptive method. This property of DG discretizations allows for a flexible procedure by which resolution can be added to a problem. This work examines the use of *hp*-adaptation for two purposes: the first of which is to place degrees of freedom within the domain as optimally as possible, the second of which is to improve solver robustness by avoiding the use of high-order polynomial approximations in regions of the mesh where it is not appropriate. In regions where the solution is smooth, *p*-enrichment is utilized, whereas in regions where the solution is not smooth, *h*-refinement is employed. Furthermore, the constant presence of discontinuous solutions in practical problems of interest motivates one to examine adaptation techniques that take this into account as robustly as possible.

While the application of standard limiting methods from the finite-volume literature have been attempted [35, 67, 75, 94, 95], these methods are either, not aware of the resolution coupling properties of DG or extend the stencil beyond the nearest neighbors. Either of these properties is sub-optimal in the author's opinion. Rather than attempt this type of limiting, *hp*-adaptation will be used to mimic the properties of a limiter. *hp*-adaptation can be viewed as a resolution coupling aware limiting approach, a so-called bottom up limiter that starts the solution at low discretization order and only increases the discretization order where appropriate, based on solution smoothness. In regions where the solution is discontinuous or non-smooth, *h*-refinement is invoked so that resolution is increased in a stable manner. In contrast, traditional slope limitation assumes that second or higher-order accuracy is appropriate everywhere in the domain and then reduces the order in non-smooth regions

without adding additional unknowns. This can be thought of as a top down approach to limiting. Thus *hp*-adaptation and slope limitation are very similar processes but operate in reverse directions of each other and when resolution coupling is present the bottom up approach of *hp*-adaptation is preferred.

5.2 Output Error Estimation

In this work the adaptation procedure is driven by local error estimates or by estimating the error in an output functional of interest. For output-based error estimation, the predicted error may also be used to give a correction to the functional value. In this section the derivation for output-based error estimation is presented along with the simplifications that lead to local error estimates.

5.2.1 Formulation

The following formulation is based on the approach described in reference [45]. Consider the functional of interest $\mathcal{L}(\mathbf{u})$ evaluated with the discrete flow-field variables, where the argument to the functional satisfies the following non-linear operator.

$$R(\mathbf{u}) = 0 \tag{5.2.1}$$

Furthermore, consider a coarse mesh $\mathcal{T}_H$ and flow solution $\mathbf{u}_H$ which satisfies the non-linear residual on the coarse mesh.

$$R_H(\mathbf{u}_H) = 0 \tag{5.2.2}$$

The solution $\mathbf{u}_H$ is used to evaluate the functional $\mathcal{L}_H(\mathbf{u}_H)$ on the coarse (i.e. current) mesh. Given this flow solution and functional, one seeks an estimate of the functional on a globally refined mesh $\mathcal{T}_h$, without computing the flow solution on the globally refined mesh. Therefore, the fine grid functional is expanded in a Taylor series about a solution projected from the coarse mesh to the fine mesh denoted by $\mathbf{u}_H^h$.

$$\mathcal{L}_h(\mathbf{u}_h) = \mathcal{L}_h(\mathbf{u}_H^h) + \left(\frac{\partial \mathcal{L}_h}{\partial \mathbf{u}_h}\right)_{\mathbf{u}_H^h} (\mathbf{u}_h - \mathbf{u}_H^h) + \cdots \tag{5.2.3}$$

where $\mathcal{L}_h(\mathbf{u}_H^h)$ is the fine mesh functional evaluated with the coarse mesh solution projected to the fine mesh. The vector $\left(\frac{\partial \mathcal{L}_h}{\partial \mathbf{u}_h}\right)_{\mathbf{u}_H^h}$ is the sensitivity of the functional with respect to the solution evaluated at the same projected state. To eliminate the term involving the solution on the fine mesh $\left(\mathbf{u}_h - \mathbf{u}_H^h\right)$, one appeals to the constraint equation. The residual defined by equation (5.2.1) evaluated on the fine mesh can also be expanded about the projected solution as:

$$\mathbf{R}_h(\mathbf{u}_h) = \mathbf{R}_h(\mathbf{u}_H^h) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]_{\mathbf{u}_H^h} (\mathbf{u}_h - \mathbf{u}_H^h) + \cdots \tag{5.2.4}$$

The fine level residual is constrained to be zero which allows one to re-arrange equation (5.2.4) to solve for the quantity involving the unknown fine solution as

$$(\mathbf{u}_h - \mathbf{u}_H^h) \approx -\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^{-1}_{\mathbf{u}_H^h} \mathbf{R}_h(\mathbf{u}_H^h) \tag{5.2.5}$$

Upon substitution of equation (5.2.5) into equation (5.2.3) one obtains the following expression for the estimate of the error in the functional

$$\mathcal{L}_h(\mathbf{u}_h) - \mathcal{L}_h(\mathbf{u}_H^h) \approx -\left(\frac{\partial \mathcal{L}_h}{\partial \mathbf{u}_h}\right)\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^{-1}_{\mathbf{u}_H^h} \mathbf{R}_h(\mathbf{u}_H^h) \tag{5.2.6}$$

where the flow residual on the fine mesh $\mathbf{R}_h(\mathbf{u}_H^h)$ is non-zero since the coarse mesh flow solution projected to the fine mesh does not satisfy the discrete equations on the fine mesh. Next the fine mesh adjoint variable $\mathbf{\Lambda}_h$ is defined as the variable satisfying

$$\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^T_{\mathbf{u}_H^h} \mathbf{\Lambda}_h = \left(\frac{\partial \mathcal{L}_h}{\partial \mathbf{u}_h}\right)^T \tag{5.2.7}$$

Therefore the functional error can now be defined in terms of the adjoint variable

$$\mathcal{L}_h(\mathbf{u}_h) - \mathcal{L}_h(\mathbf{u}_H^h) \approx -(\mathbf{\Lambda}_h)^T \mathbf{R}_h(\mathbf{u}_H^h) \tag{5.2.8}$$

The solution of the adjoint problem should be expected to cost as much as the flow solution and thus it is undesirable to compute the fine grid adjoint variable $\mathbf{\Lambda}_h$ directly. Therefore the coarse level adjoint solution is obtained via

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{u}_H}\right]^T_{\mathbf{u}_H} \mathbf{\Lambda}_H = \left(\frac{\partial \mathcal{L}_H}{\partial \mathbf{u}_H}\right)^T \tag{5.2.9}$$

which is solved on the coarse mesh. In this work, the fine mesh (level) employed for error estimation contains the same number of elements as the original mesh but employs a discretization order of $p+1$ where the coarse mesh employs a discretization order of p. Therefore the transition of mesh levels $H \rightarrow h$ is equivalent to the transition of discretization orders $p \rightarrow p+1$. The coarse adjoint solution is projected onto the fine mesh by injection. Injection is the process of initializing the fine mesh solution with the coarse mesh solution without performing any interpolation to obtain the fine mesh solution. Since the fine mesh employs a discretization order of $p+1$, the injection operator is defined by setting the modes from 1 to M_p of $\mathbf{\Lambda}_H^h = \mathbf{\Lambda}_H$ and setting the remaining high-order modes to zero, where M_p is the number of modes in a discretization of order p. The injection operator is followed by a small number ($<= 5$) of linear solution iterations on the fine mesh to generate the approximate fine mesh adjoint variable $\mathbf{\Lambda}_H^h$. Reference [17] has used a patch-wise least-squares method to reconstruct the adjoint solution on the fine mesh. However, the reconstruction procedure is more complicated in the current context, which involves mixed-element non-conforming meshes. Furthermore, several solution cycles on the fine mesh results in a relatively low cost operation and gives an approximate fine level adjoint solution, which is based on the discrete fine mesh equations. Introducing the approximate fine level adjoint solution results in the error estimate:

$$\mathcal{L}_h(\mathbf{u}_h) - \mathcal{L}_h(\mathbf{u}_H^h) \approx -\underbrace{(\mathbf{\Lambda}_H^h)^T \mathbf{R}_h(\mathbf{u}_H^h)}_{\epsilon_c} - \underbrace{(\mathbf{\Lambda}_h - \mathbf{\Lambda}_H^h)^T \mathbf{R}_h(\mathbf{u}_H^h)}_{\epsilon_a} \tag{5.2.10}$$

where ϵ_c is the computable error and ϵ_a is the error due to the approximate fine level adjoint. The magnitude of the contribution to the computable error from a particular element k is

$$\epsilon_{c_k} = \left| \left(\mathbf{\Lambda}_H^h\right)^T \mathbf{R}_h \left(\mathbf{u}_H^h\right) \right|_k \tag{5.2.11}$$

Additionally a so-called local discretization error estimate can be obtained by using the estimated fine level residual as an error indicator. This gives the local error estimate as

$$\epsilon_{l_k} = \left| \mathbf{R}_h \left(\mathbf{u}_H^h\right) \right|_k \tag{5.2.12}$$

Essentially, the local error is measure of how well the current solution satisfies the discrete equations of one order of accuracy higher than the current order of accuracy. The local error

estimate is not a goal oriented error estimation approach and does not target any output of the solver.

In this work the element-wise contributions of computable error ϵ_{c_k} or local error ϵ_{l_k} are used as the adaptation criteria. Following reference [18], the mesh is adapted by targeting for refinement elements that contribute a certain fraction of the total error in the mesh, usually $> 90\%$. For this work 99% of the total error is targeted. The process forms a sorted list of the elements according to the magnitude of their contribution to the total error, from the highest to the lowest values. A loop over the queue is performed and elements are flagged for refinement until the total amount of error processed exceeds the specified percentage of the total error. This ensures that only the elements with the highest contribution to the total error are refined for highly non-uniform error distributions. When the error has become more uniformly distributed near-uniform refinement will occur. Once the elements have been tagged for adaptation they are refined via either a p-enrichment or h-refinement procedure, depending on the measured local smoothness of the solution $\mathbf{u}_H$.

The computable error ϵ_c can be used to provide a more exact coarse level functional, by providing a correction to the coarse level functional $\mathcal{L}_h\left(\mathbf{u}_H^h\right)$ given by:

$$\mathcal{L}_h\left(\mathbf{u}_H^h\right)_{corr} = \mathcal{L}_h\left(\mathbf{u}_H^h\right) + \epsilon_c \tag{5.2.13}$$

$\mathcal{L}_h\left(\mathbf{u}_H^h\right)_{corr}$ is the so-called corrected functional. The corrected functional will be a good approximation of $\mathcal{L}_h\left(\mathbf{u}_h\right)$ provided the linear Taylor series approximation employed to estimate the functional error is valid. A linear Taylor series is a valid approximation if the behavior between coarse and fine level functionals is close to linear, which is not guaranteed.

5.3 hp-Adaptation

Discontinuous Galerkin methods increase the order of accuracy by adding additional modes to the expansion in equation (2.2.11), hence increasing the discretization order adds additional unknowns to the mesh. This gives DG methods additional flexibility with regard to the placement of the degrees of freedom by an adaptation strategy. In particular, DG methods have two paths by which to increase resolution: h-refinement and p-enrichment.

References [20, 46] have developed an unsteady mesh adaptation procedure within the context of high-order DG discretizations. However, the mesh adaptations are performed at a fixed discretization order and thus only exploit one method of adding resolution to the problem. References [17, 59] have developed an *hp*-adaptive approach for DG discretizations of the compressible Euler equations on purely triangular meshes and demonstrated the effectiveness of this approach for computing both purely smooth flows and flows with discontinuities. Herein the work of references [17,59] is extended to viscous flows on mixed-element meshes. Additionally, the discontinuity capturing ability of *hp*-adaptation is enhanced, using a similar combined *h*-refinement and *p*-enrichment approach and artificial diffusion for shock capturing. In what follows, each method of adaptation is described in isolation, and the techniques developed for combining *h*-refinement and *p*-enrichment methods to achieve *hp*-adaptation are presented.

5.3.1 *h*-Refinement

Reference [15] has shown that using quadrilateral elements in the highly stretched regions of the mesh is advantageous, as demonstrated and discussed in Chapter 4. As a result, the refinement process becomes more complicated than the simpler case of conforming triangular meshes [17, 59]. For meshes containing quadrilateral elements it is convenient to allow for non-conforming interfaces (i.e. hanging nodes) in the mesh. Therefore, triangles are now also refined such that triangles can have non-conforming interfaces. Refinement of both element types is done on a four-to-one basis with no more than a two-to-one discrepancy between the size of neighboring elements. Furthermore, while it is commonplace to smooth the refined meshes after they are generated, no mesh smoothing is applied in this work because mesh smoothing can corrupt the structure of the anisotropic boundary layer mesh.

The refinement pattern for triangles is depicted in Figure 5.1. The triangle is refined using mid-point subdivision where a node is inserted at the mid-point of each edge on the triangle. This results in four children (4:1) for each subdivided element. Quadrilaterals are refined in an analogous manner as depicted in Figure 5.2 with the exception that an additional node is placed at the center of the refined quadrilateral. The present method

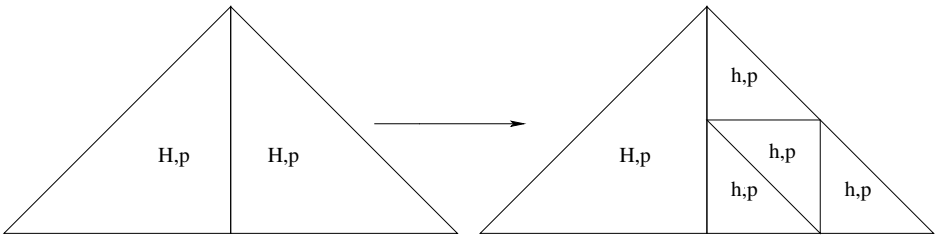


Figure 5.1: Illustration of the triangle h-refinement pattern.

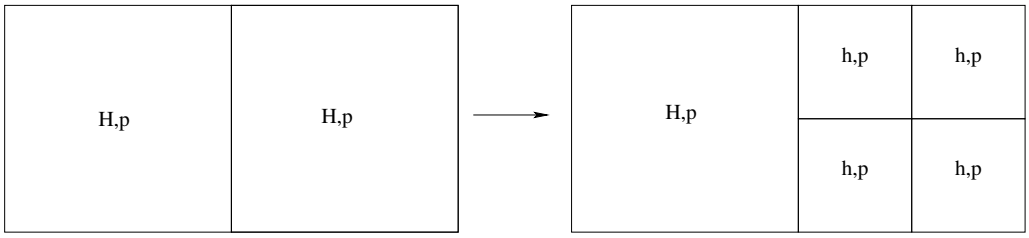


Figure 5.2: Illustration of the quadrilateral h-refinement pattern.

allows for the presence of hanging nodes for both triangular and quadrilateral elements as shown in Figures 5.1 and 5.2.

In an attempt to enforce a no more than 4:1 refinement rule, any element with all but one of its edges flagged for refinement will have its final edge refined. If an element has an edge marked for refinement and that edge is connected to a hanging node, then the element is flagged for a full refinement. While in this work de-refinement is not implemented, one could also apply de-refinement in this situation to enforce a no more than 4:1 refinement rule.

The presence of hanging nodes complicates the inter-element surface integral compared to a conforming mesh. In this approach an edge connected by any two nodes is defined as a unique edge in the mesh. Thus triangles with a hanging node actually have four edges (similarly quadrilaterals can have up to six edges). The surface integral between non-conforming elements where one has a hanging node is accomplished by computing each edge integral separately and then adding these individual edge fluxes back into the elements on each side of the edge. While for the element with the hanging node the two edges that surround the hanging node have unique identification numbers in physical space, they

have the same local edge number on the non-conforming element in the transformed space. Essentially, the edges surrounding a hanging node make up equal portions of a single edge in the transformed space, as depicted in Figure 5.3. The quadrature points for the non-

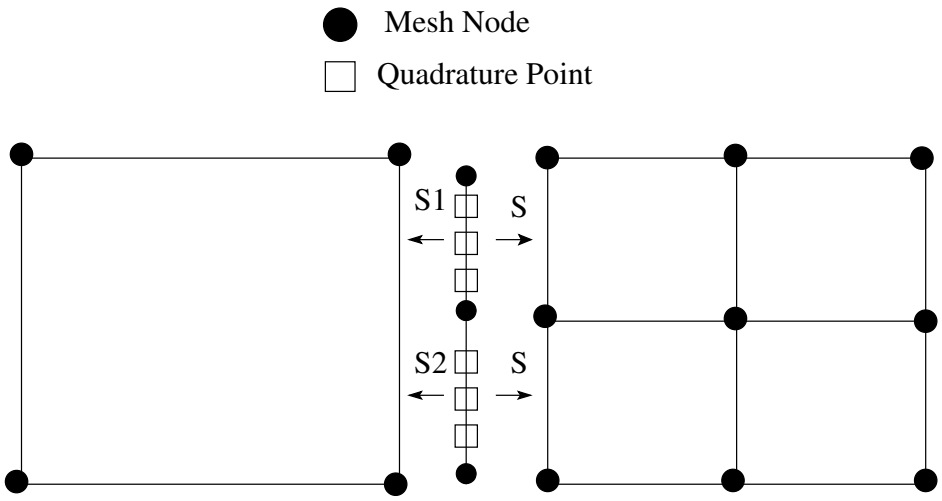


Figure 5.3: Diagram of the surface integral done at a non-conforming interface.

conforming side, i.e. the points viewed from the element on the left are linearly mapped to reside on half of a standard edge i.e. $S \to S_1$ or $S \to S_2$. These points are compressed and translated via the following formulas

$$\begin{aligned} S1 &= \frac{1}{2}S - S \\ S2 &= \frac{1}{2}S + S \end{aligned} \tag{5.3.1}$$

Thus for the element on the right side $S \in [-1, 1]$, however for the element on the left side S is split into $S1 \in [-1, 0]$ and $S2 \in [0, 1]$. Thus for conforming edges the edge integral is performed over the full edge length in transformed space($S \in [-1, 1]$), while the edge integral for the non-conforming element is split over two halves of the edge in transformed space, i.e $S1 \in [-1, 0]$ and $S2 \in [0, 1]$. The volume integrals are unaffected by the presence of hanging nodes.

5.3.2 Non-conforming Mesh Adaptation Mechanics

The presence of hanging nodes in the mesh necessitates setting some standard rules for the adaptation. When there are non-conforming interfaces the element surface integrals can

become complicated. In order to simplify the surface integrals, a rule of no-more than 2:1 in standard element edge length is permitted for non-conforming interfaces. In order to accomplish this, rules are placed on the mesh refinement strategy. Firstly, if all but one edge of an element is split by refinement then the element is flagged for refinement. This has two consequences. The first is to keep the number of non-conforming interfaces to a minimum for each element type (one for triangles and two for quadrilaterals). The second is to help provide smoother cell size distributions throughout the mesh by removing potential un-refined holes from the pattern of mesh refinement. Figure 5.4 depicts this process for triangular elements and Figure 5.5 for quadrilateral elements. Initially the two red elements have been flagged for refinement but the light blue one has not been flagged. The algorithm detects that all but one neighbor of the light blue element has been flagged for refinement and then tags the light blue element for refinement as well.

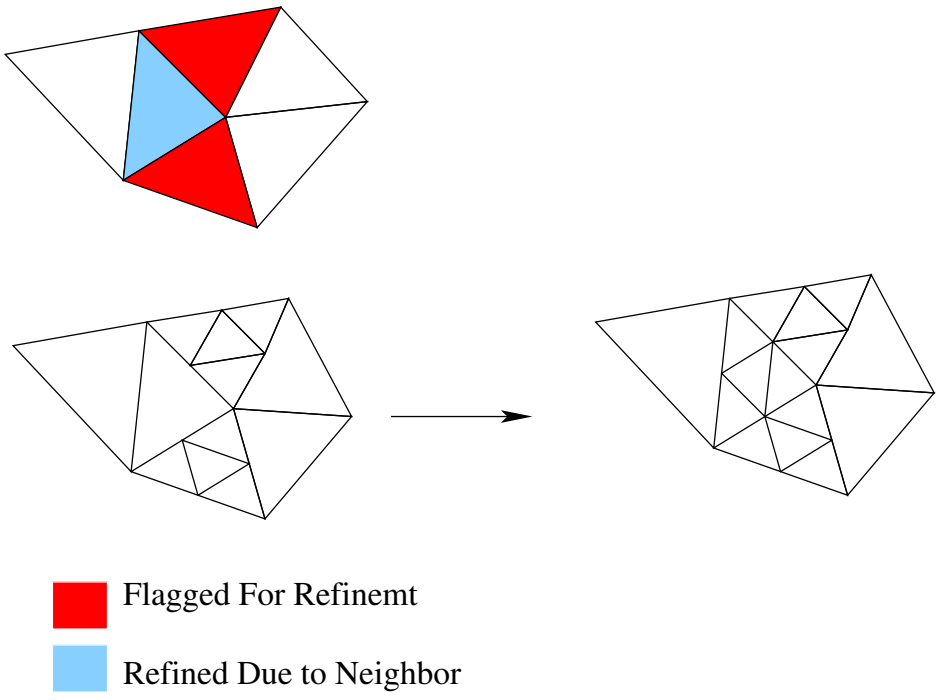


Figure 5.4: Refinement rule for triangles.

The second measure put in place prevents the subdivision of half length non-conforming edges. This keeps all non-conforming interfaces within a 2:1 edge length ratio, thus requiring only the simple non-conforming edge integration rule described previously. Non-conforming

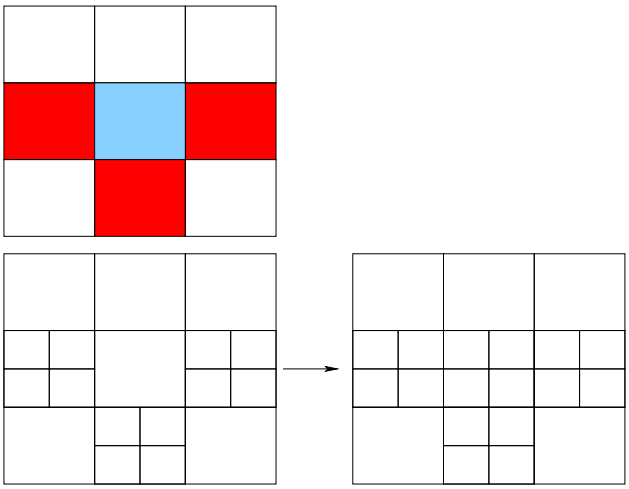


Figure 5.5: Refinement rule for quadrilaterals.

edges that are less than half length are prevented by ensuring, that if a half length edge is to be divided then the cell containing the non-conforming edge is also divided. Figure 5.6 shows a graphical example of this for triangles and Figure 5.7 shows a graphical example for quadrilaterals.

In this case the red elements are flagged for refinement but the green elements are not. Since refinement of the red element alone would result in an edge that makes up one quarter of a full edge for the non-conforming element, the green element is also refined, which ensures that all the non-conforming edge length ratios are 2:1. Enforcing this type of rule ensures that the quadrature rule illustrated in Figure 5.3 is performed correctly based on remapping the quadrature points according to equation (5.3.1). This measure also aids in generating a smooth element size distribution in the adapted mesh by ensuring that refinement does not proceed in too highly localized an area i.e. one element is continually refined while the element's neighbors never receive any refinement. It should be noted that large resolution discrepancies can induce artificial oscillations in the solution and as such efforts to provide smooth mesh resolution throughout the adaptive process are important.

5.3.3 Non-conforming Mesh Curvature

In order to attain optimal accuracy curved elements must be employed on the boundaries of the domain that are curved surfaces, such as the surface of an airfoil. Furthermore, in

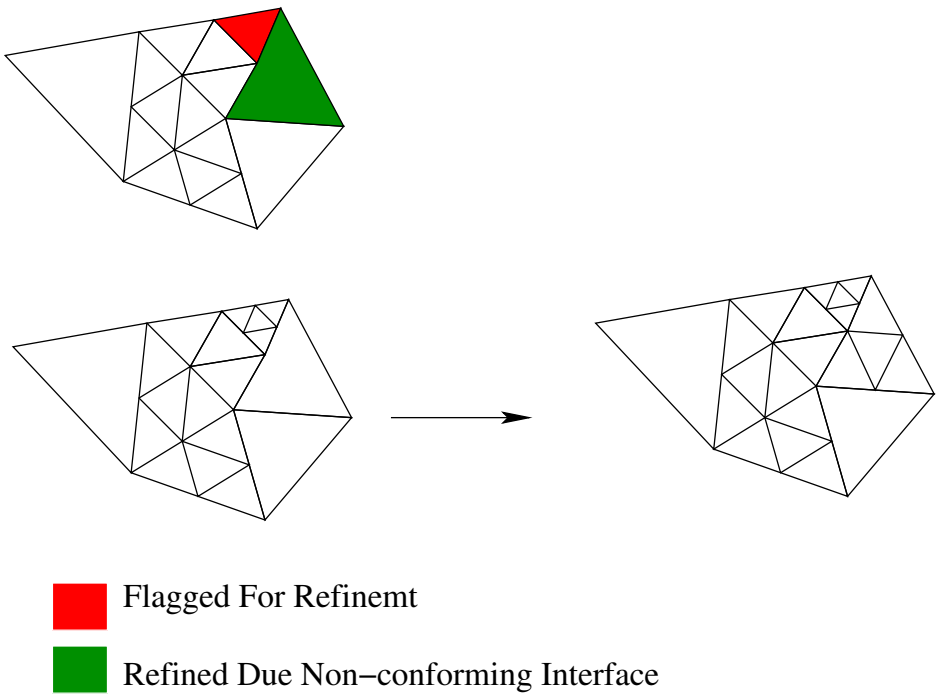


Figure 5.6: Non-conforming refinement rule for triangles.

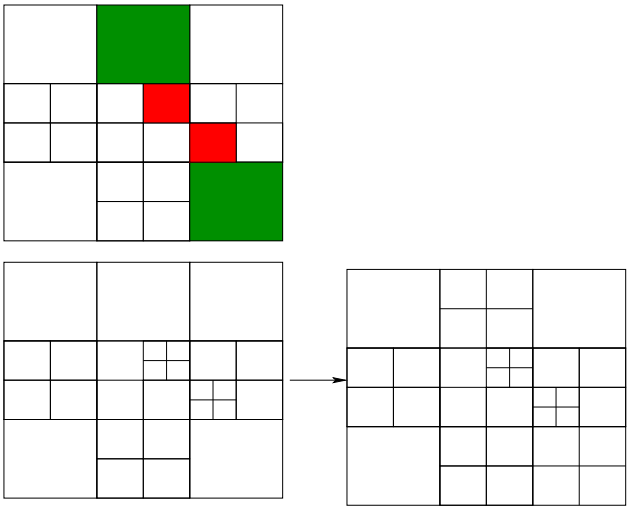


Figure 5.7: Non-conforming refinement rule for quadrilaterals.

order to maintain the integrity of anisotropic meshes, interior mesh elements must also be curved, which is a result of curving the elements on the physical boundary. The strategy used to curve the boundary elements is described in Section 2.4 and curving the interior mesh is discussed in Section 4.1.1. When considering non-conforming meshes, special attention must be paid to how the elements that have hanging nodes are curved. The presence of

non-conforming interfaces between elements means that there is a mismatch between the number of unknowns used to curve the elements on each side of the interface. However, this does not preclude using non-conforming meshes to refine curved elements.

There are two possible ways to curve an adaptively refined mesh. The first method takes the initial mesh, applies the desired refinement and then re-curves the mesh edges. For non-conforming meshes this can result in a situation such as the one depicted in Figure 5.8. In this case, one cannot guarantee that the non-conforming interfaces will be aligned after curving the mesh. This is due to the fact that the refined edges on a non-conforming interface

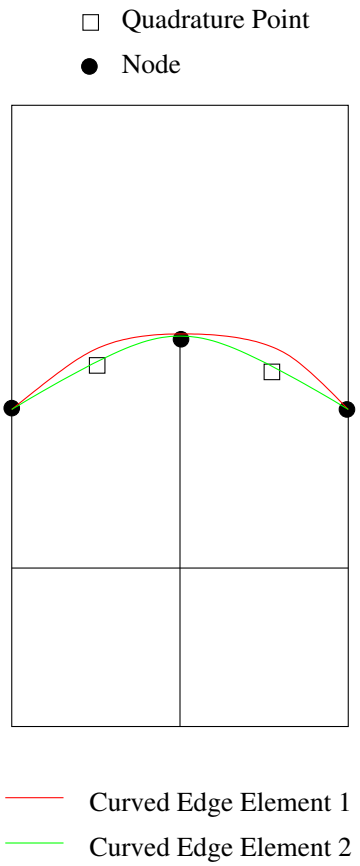


Figure 5.8: Example of unconstrained element curvature for non-conforming element interface. The red and green curves should be coincident at all points otherwise the mesh has a whole in it.

contain more mapping degrees of freedom, which if left unconstrained, will result in the situation depicted in Figure 5.8. However, this can be overcome by applying a constraint to ensure that the edges defined from both sides of the interface conform to the same mapping as shown in Figure 5.9. This mapping is generated by constraining the two quadrature

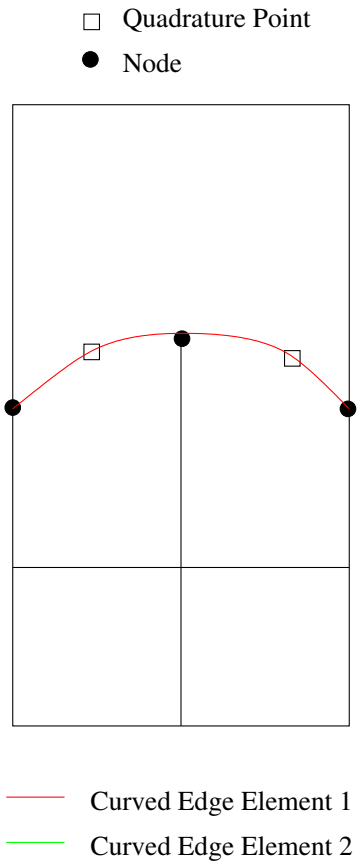
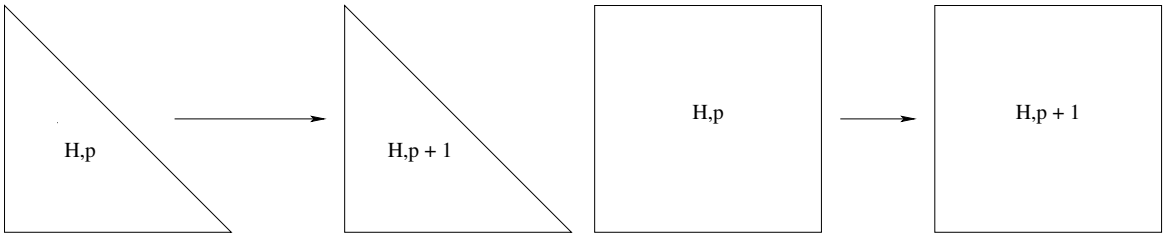


Figure 5.9: Example of constrained element curvature for non-conforming element interface. In this case the edges defined from both elements are coincident at all points, since one cannot see the green curve, which is under the red curve.

points(squares) to lie on the red curve in Figure 5.8, which essentially forces the two smaller curved edges shown as green curves to lie on the red curve in Figure 5.8. In practice, this is accomplished by curving the initial mesh in the adaptation to the maximum order required for all adaptive meshes. Then new nodes are placed on this curvature and rather than on the original geometry definition obtained from the mesh generator, which is where one would normally place additional surface mesh points. This establishes the required constraint in an indirect and easy to implement fashion. Using the mesh curvature definition to add new nodes during adaptation is acceptable since the initial mesh was curved using the geometry information from the mesh generator.

5.3.4 p-Enrichment

In contrast to h-refinement, p-enrichment refines the element in question by maintaining the current element size and connectivity. The p-enrichment procedure is much simpler than the h-refinement procedure and consists of simply increasing the discretization order from p to $p+1$ on the element flagged for refinement. p-enrichment is implemented for both element types as depicted in Figures 5.10(a) and 5.10(b), and a jump of no more than one discretization order is permitted between elements. This is enforced by looping over the mesh elements and checking to see if this rule is violated. If the rule is violated then the order of the offending element is raised as illustrated in Figure 5.11. This looping is performed iteratively until no more offending elements are found.



(a) Triangle (b) Quadrilateral

Figure 5.10: Illustration of p-enrichment on both triangles and quadrilaterals.

While p-enrichment induces no additional geometrical complexity, one does need to address how many quadrature points must be used to integrate the fluxes along the edges. In previous work the edge fluxes are integrated to $2p+1$ accuracy. When using a grid with variable discretization order it is necessary to use an integration rule that integrates the edge fluxes to $2(max(p^+, p^-))+1$ accuracy where p^+ and p^- denote the element order on each side of an edge. The solutions on either side of an edge are evaluated using the number of modes available from each individual element sharing the edge. The volume integrals remain unaffected by the variable discretization order throughout the mesh.

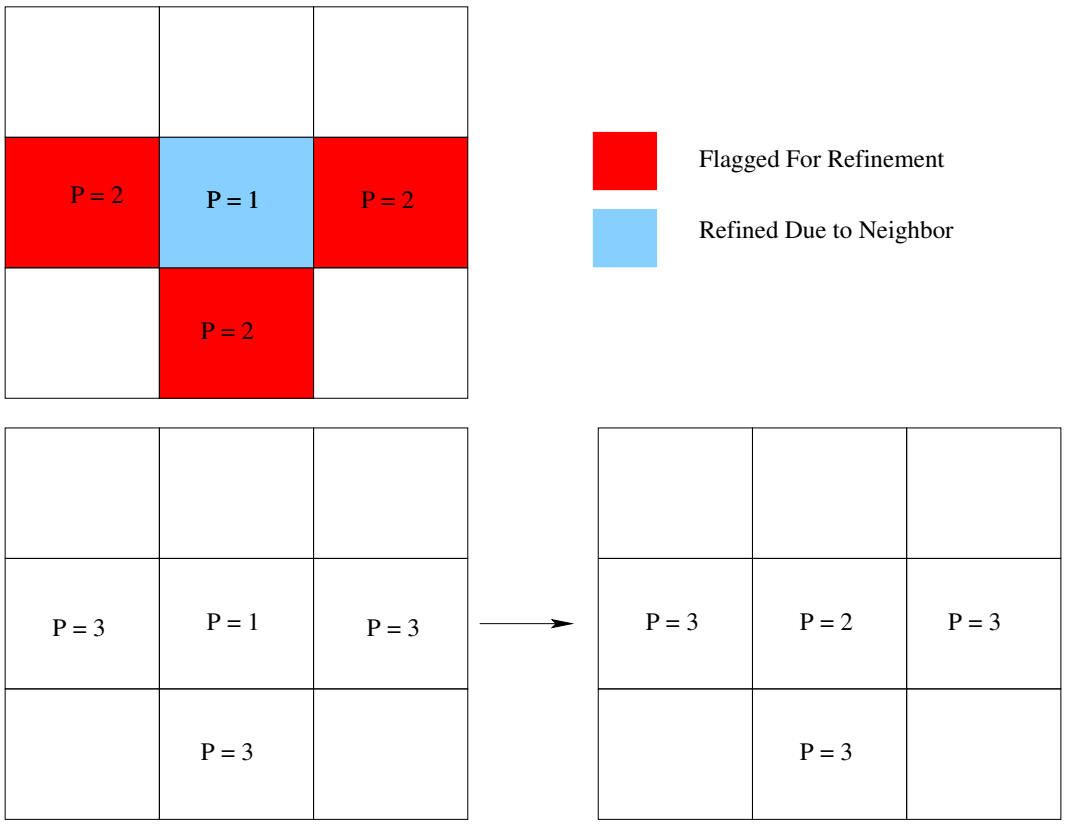


Figure 5.11: Order enrichment rule applied to quadrilaterals, triangles are treated exactly the same way.

5.3.5 *hp*-Adaptation

It is well known that using high-order polynomials in the vicinity of discontinuities results in unphysical numerical oscillations or Gibbs phenomena, which for the Navier-Stokes/Euler equations can cause solver failure. Hence a combination of h-refinement and p-enrichment is employed to account for the presence of discontinuities in the solution. This proceedure uses h-refinement near discontinuities and p-enrichment in smooth flow regions. The objective is to allow discontinuities to be captured using a low-order discretization while using high-order discretizations in smooth flow regions where the use of high-order discretizations is appropriate.

hp-adaptation is a hybridization of h-refinement and p-enrichment techniques. These two techniques are used in tandem such that if an element is to be refined, a decision must be made as to whether to use h-refinement or p-enrichment. The current implementation

examines the smoothness of the primal solution to determine which type of adaptation is used for each element. The smoothness is determined by examining the jump indicator of reference [67]. The value of the jump indicator for an element is given by

$$(\tilde{s}_k)_f = \frac{1}{|\partial\Omega_k|} \int_{\partial\Omega_k} \left| \frac{[\![q]\!] \cdot \vec{n}}{\{q\}} \right| ds \tag{5.3.2}$$

where q is taken as the pressure and the average($\{\}$) and jump($[\![]\!]$) operators are defined in equation (2.2.5) and equation (2.2.6) respectively. This indicator is essentially a summation of the inter-element jumps in pressure for each element. For a shock wave the jump indicator will return a value of $\mathcal{O}(1)$ because the jump of pressure and average pressure are of the same order of magnitude for a shock wave, while for smooth regions the jump in pressure is much smaller than the average. The choice between whether to refine an element with h-refinement or p-enrichment is made by

$$\begin{cases} \tilde{s_k} > \frac{1}{\mathcal{K}}, & h\text{-refinement} \\ \tilde{s_k} < \frac{1}{\mathcal{K}}, & p\text{-enrichment} \end{cases} \tag{5.3.3}$$

where $\mathcal{K} = 25$ is used throughout this work, as in reference [17]. In addition to selecting the refinement strategy based on the solution smoothness, a maximum discretization order p_{max} is also enforced. When an element reaches the prescribed maximum discretization order and further refinement is required, h-refinement is substituted for p-enrichment even if the solution within the cell is smooth.

Though hp-adaptation is designed to place degrees of freedom optimally for a given objective functional, hp-adaptation can also be viewed as a technique to enhance the robustness of the DG solver. In essence hp-adaptation seeks to design the mesh based on the solution, which for cases of under-resolved phenomena such as those encountered in reference [15] should ultimately result in a mesh of sufficient local resolution such that a high-order discretization can be used throughout. For flow features that will most likely remain under-resolved for the entire simulation (e.g. shock waves and contact discontinuities) the hp-adaptive scheme is capable of adding degrees of freedom while maintaining low discretization order in such a way as to avoid Gibbs phenomena, providing a natural way for the present DG solver to handle non-smooth solutions robustly.

5.4 Numerical Results

The proposed *hp*-adaptive method has been evaluated using four test cases. The first two test cases consist of the laminar viscous flow over a NACA0012 airfoil and a two-element airfoil. The first test case is presented to compare *hp*-adaptation with *h*-refinement at second-order($p = 1$) as well as with uniform *h*-refinement and uniform *p*-enrichment. The two-element airfoil case represents a practical application of the *hp*-adaptive method to a more complicated geometry. The third test case consists of the inviscid transonic flow over a NACA0012 airfoil, which is presented to demonstrate the accurate and robust shock capturing abilities of the *hp*-adaptive method. The results of the *hp*-adaptation are compared with uniform *p*-enrichment using the piecewise constant artificial viscosity method of Section 2.7.1. Additionally, the piecewise constant artificial viscosity is combined with *hp*-adaptation in order examine how these two robustness enhancement measures interact. The fourth and final test case consists of supersonic viscous flow over a half cylinder geometry. In this case *hp*-adaptation with integrated surface heating as the objective is employed. This case is a culmination of the previous test cases as the objective depends strongly on both the shock wave and boundary layer structures, e.g. on smooth and non-smooth flow physics.

Where appropriate the computational grid may contain both triangles and quadrilaterals within the same domain. Mixed-element meshes are addressed by the adaptive algorithm by allowing for non-conforming interfaces between elements of all types. For all test cases the adaptation is terminated when the functional of interest is grid converged i.e. the functional changes by less than .5% from one adaptation step to the next. The performance of the method is measured by considering the number of degrees of freedom (DoFs) required to yield a grid converged functional. The number of DoFs is determined as the total number of unknowns per equation in the domain. For example, a purely triangular mesh of 100 elements with a uniform discretization order of $p = 1$ would have 300 DoFs, which is 3 DoFs per triangle. The computational cost of generating these results is demonstrated by showing functional or functional error versus wall clock time (i.e. CPU-time or computational time).

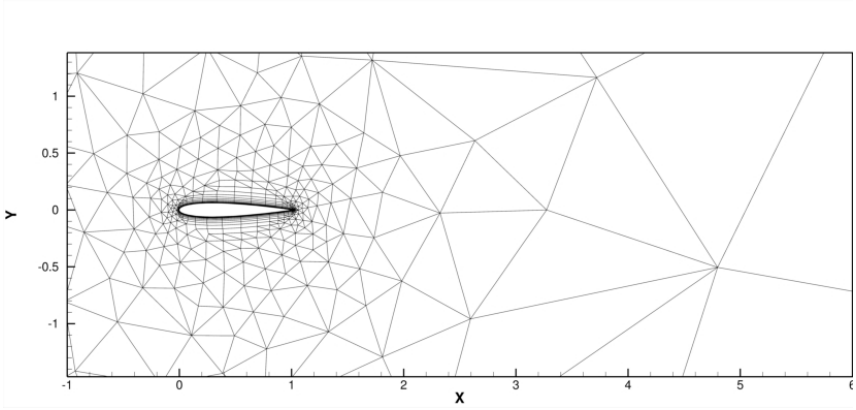
All results except the half cylinder have been computed using the Riemann solver of Roe [52] on the cell interfaces. All test cases are steady-state solutions and the flow and

adjoint equations have been converged such that the residuals have been reduced by 12 orders of magnitude at each stage of the adaptive process. In some sense this represents the worst case scenario for timing adaptive methods because one would probably only partially converge the intermediate steps before moving on to the next adaptive cycle.

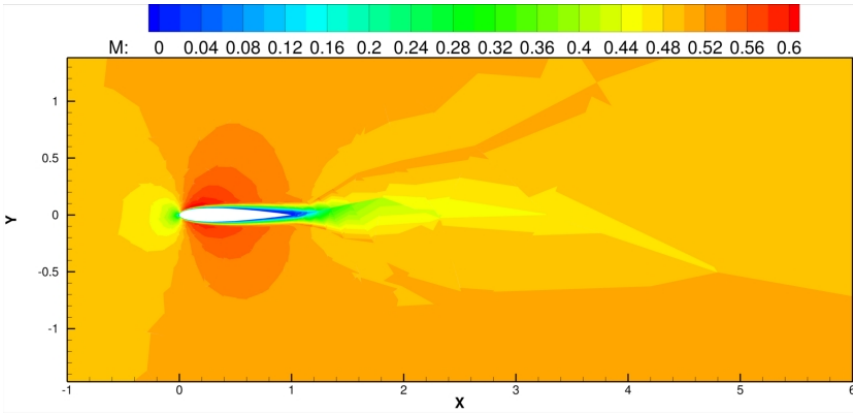
5.4.1 NACA0012 Airfoil: Drag-based Adaptation

The first test case consists of the laminar flow over a NACA0012 airfoil. The flow conditions are a free-stream Mach number $M_\infty = .5$, angle of attack $\alpha = 1^o$, and Reynolds number based on chord length $Re = 5,000$. Adjoint-based hp-adaptation and h-refinement with $p = 1$(second-order) are utilized for the adaptive mesh refinement of this flow. Additionally, uniform h-refinement and uniform p-enrichment are performed in order to draw comparisons between uniform and adjoint-based goal-oriented refinement strategies. All refinements are initialized on a mesh consisting of $N = 1,148$ elements, with a uniform discretization order of $p = 1$ resulting in 3,930 DoFs. When employing the hp-adaptive approach for this case, the maximum discretization order in the grid is set at $p_{max} = 5$(i.e. 6-th order accurate). Thus any cell that requires refinement and already has a discretization order of $p = 5$ will be subdivided using h-refinement regardless of how smooth the solution is within that cell. This test case is shown to illustrate the high efficiency of the hp-adaptive approach, i.e. hp-adaptation can produce very accurate functionals with respect to the reference solution using relatively few degrees of freedom when compared against uniform refinement/enrichment and/or h-refinement alone. Drag is chosen as the target functional for the adjoint-based goal oriented adaptations. The MGPC-GMRES solver described in Chapter 4 is utilized to converge both the flow and discrete adjoint equations for this test case.

Figures 5.12(a)-5.14(b) depict the initial and final grids using both adjoint h-refinement and adjoint hp-adaptation as well as computed Mach number contours on those grids. Note that the adjoint-based strategies target both the surface of the airfoil as well as the wake region downstream from the trailing edge. Refinement has also been applied upstream of the leading edge of the airfoil (recall the adjoint contours in Figure 3.2). Furthermore, note that the computed Mach contours resulting from h-refinement and hp-adaptation look



(a) Initial mesh: $N = 1,148$ elements, $p = 1$

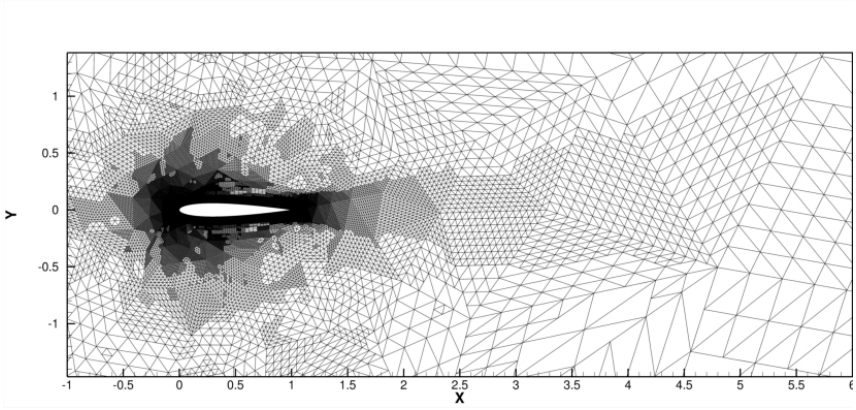


(b) Mach contours: initial mesh

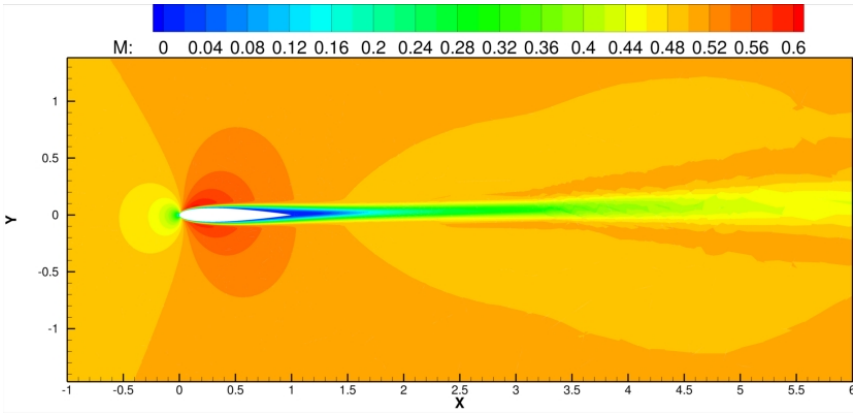
Figure 5.12: Initial mesh and Mach contours of the laminar flow over a NACA0012 airfoil with $p = 1$, $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$.

very similar at the final stage. However, examination of Figures 5.15 and 5.16 show that the *hp*-adaptation results contain approximately one third the number of DoFs compared with the *h*-refinement results. Figure 5.17 depicts the iterative convergence of the discrete flow equations using the MGPC-GMRES solver for the final adaptive step of the adjoint *h*-refinement and adjoint *hp*-adaptation. Note that the residuals are reduced by 12 orders of magnitude in both cases in order to eliminate any algebraic error that may contaminate the functional values.

Figure 5.15(a) depicts the computed drag versus the number of DoFs using adjoint-based *h*-refinement, adjoint-based *hp*-adaptation, uniform *h*-refinement and uniform *p*-enrichment. The reference value in Figure 5.15(a) was computed using the same DG solver with approx-



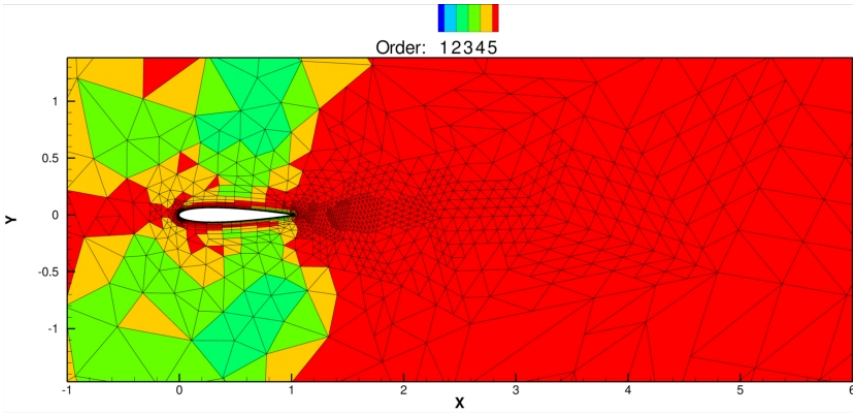
(a) Adjoint h-refinement: final mesh, $N = 174,145$, $p = 1$



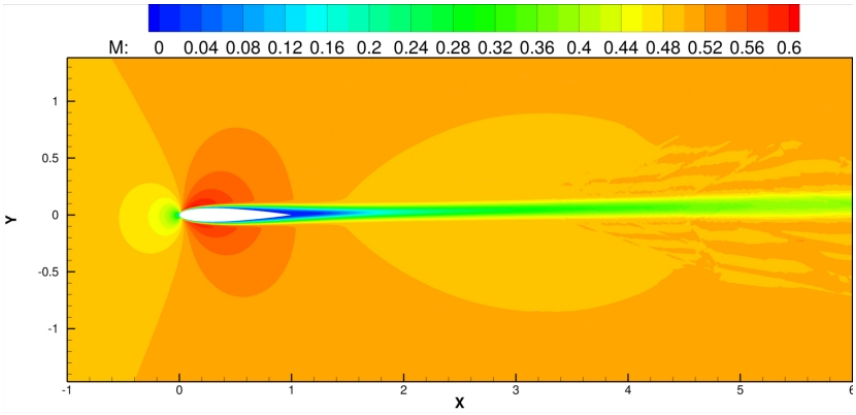
(b) Adjoint h-refinement: Mach number on the final mesh

Figure 5.13: Final mesh and Mach number contours of the laminar flow over a NACA0012 airfoil using adjoint h-adaptation with discretization order $p = 1$, $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$.

imately 250,000 DoFs at a uniform discretization order of $p = 4$ (i.e. 5-th order accuracy). Figure 5.15(a) clearly shows that the hp-adaptive method yields a grid converged drag result with the fewest number of degrees of freedom compared to any of the refinement methods presented. Comparison of the hp-adaptive approach with the h-refinement approach shows that the hp-adaptive approach yields a grid converged drag result with approximately one third the number of DoFs used in the h-refinement approach. Figure 5.15(b) depicts the drag versus the number of DoFs using the computable error predicted according to equation (5.2.10) to correct the coarse level drag. The arrows in Figure 5.15(b) point from the coarse level corrected drag to the fine level drag that the correction is estimating. The corrected drag value is computed via equation (5.2.13). While initially the corrected coarse level drag



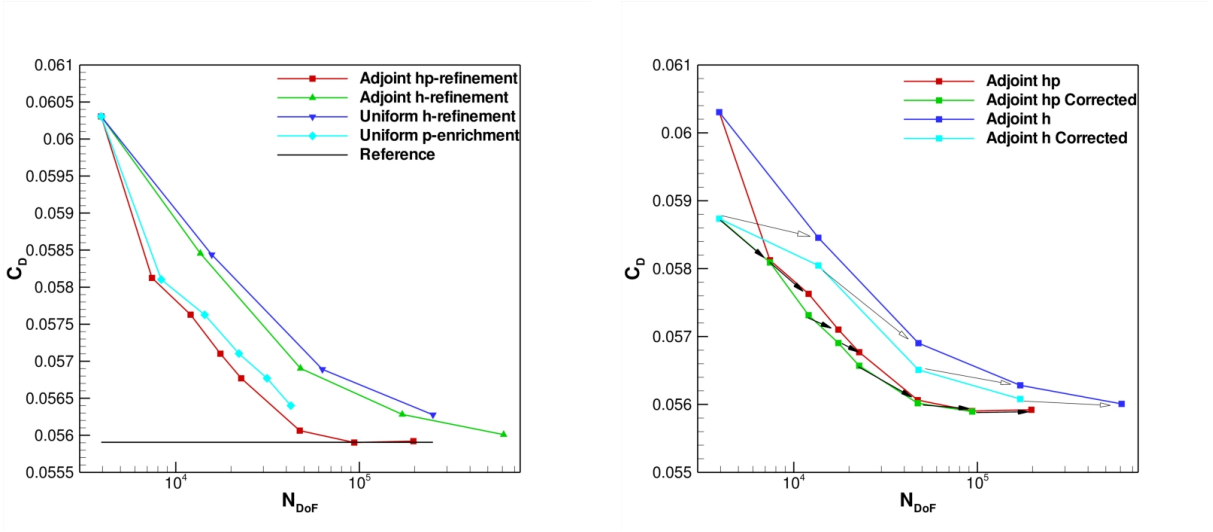
(a) Adjoint *hp*-adaptation: final mesh $N = 6,776$, $p = 1$ to $p = 5$



(b) Adjoint *hp*-adaption: Mach number on the final mesh

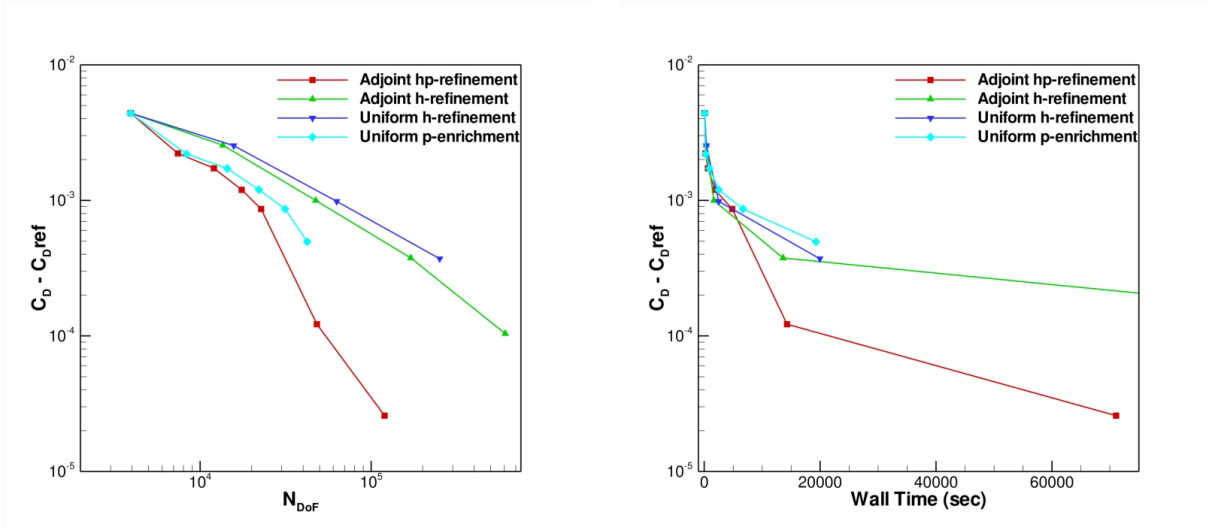
Figure 5.14: Final mesh and Mach number contours of the laminar flow over a NACA0012 airfoil using adjoint-based *hp*-adaptation with discretization order $p = 1$ to $p = 5$, $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$.

does not agree with the fine level computed drag, as the refinement process continues and the drag becomes closer to grid converged, the corrected coarse level drag values show improved agreement with the fine level computed drag values. For the last two refinement levels, the adjoint correction yields corrected coarse level drag values that closely match the corresponding fine level computed drag values. The increased effectiveness of the correction is explained by the fact that the error is predicted as a linear Taylor series expansion (equation (5.2.3)) about the coarse level solution and thus as the computed drag becomes closer to being grid converged, the linear Taylor series becomes a better approximation of the functional behavior between coarse and fine mesh levels. The adjoint *hp*-adaptive method clearly gives the most accurate drag result (i.e closest to the reference solution) for a given



(a) Drag vs. N_{DoF} (b) Adjoint corrected drag vs. N_{DoF}

Figure 5.15: Computed drag coefficient versus N_{DoF} for the laminar flow over a NACA0012 airfoil using various adaptation methods, with and without adjoint-based computable error correction.



(a) Drag error vs. N_{DoF} (b) Drag error vs. wall clock time

Figure 5.16: Computed drag error for the laminar flow over a NACA0012 airfoil using various refinement methods, including adjoint-based goal oriented *hp*-adaptation and *h*-refinement targeting drag.

number of DoFs. Note that the *h*-refinement computation was terminated early because the number of DoFs became impractically high and it was clear that the *hp*-adaptation result had become grid converged using far fewer DoFs.

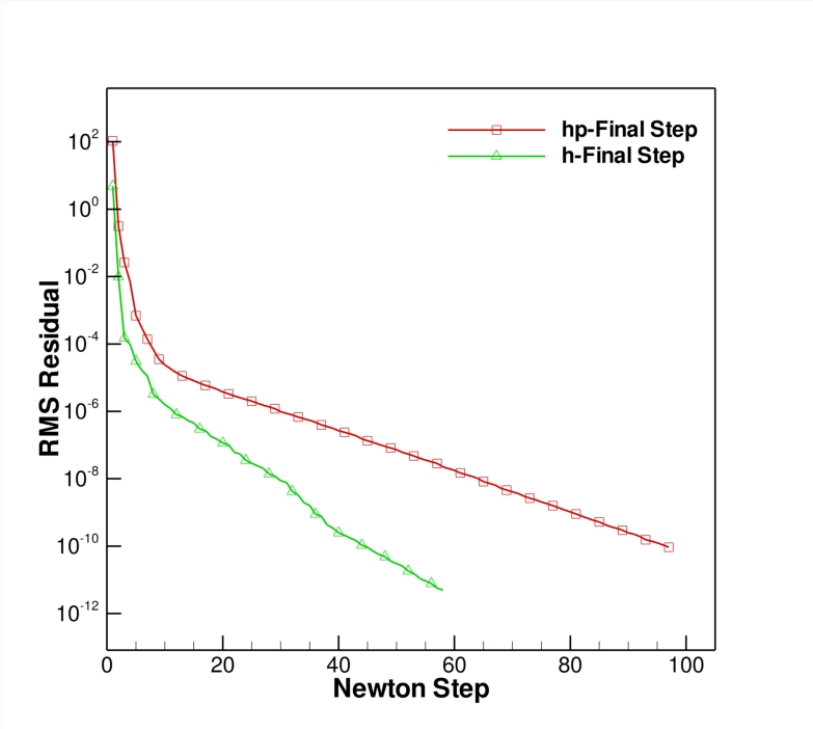


Figure 5.17: Flow solver iterative convergence using the MGPC-GMRES solver from Chapter 4 for the laminar flow over a NACA0012 airfoil.

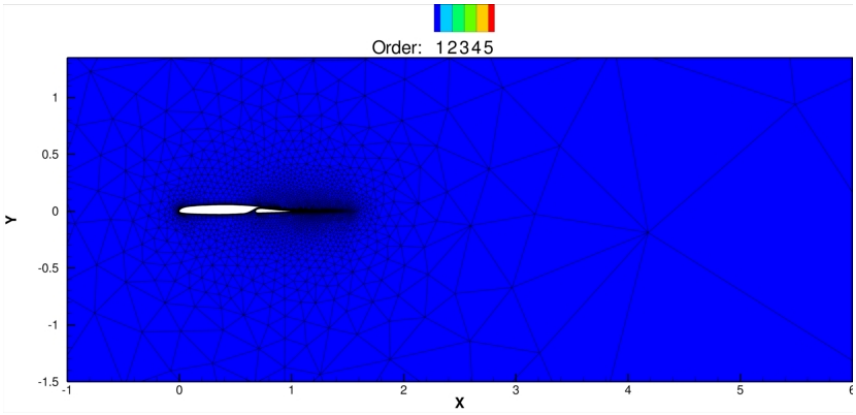
Figures 5.16(a) and 5.16(b) show the drag error versus the number of DoFs(N_{DoF}) and the wall clock time required for each computation. Figure 5.16(a) shows that the *hp*-adaptive method gives the lowest drag error per degree of freedom. The asymptotic slope of these error curves was computed for both the *h*-refinement and *hp*-adaptation computations. Theoretically for a dual consistent discretization using a uniform discretization order p, super-convergence of the functional error at a rate of order $O(h^{2p})$(where $h = \sqrt{N_{DoF}}$) should be observed [49]. This asymptotic functional error bound was proven in Chapter 3 and the final result is shown by equation (3.3.12). Computation of the asymptotic slope of the drag error versus h for the *h*-refinement computation yields the expected value of 2. If the scheme were dual inconsistent then the asymptotic slope of the drag error versus h curve would be 1 i.e. $O(h^p)$ (see Section 3.3). Computation of the asymptotic slope of the drag error versus h for the *hp*-adaptation computation results in a slope of 8.8, which is a striking result. Even though only a fraction of the grid contains cells with a $p = 5$ discretization *hp*-adaptation is able to obtain very close to the theoretical slope of 10. Figure 5.16(b) depicts the drag error

versus the wall clock time, which shows that *hp*-adaptation yields lower drag error using a fraction of the wall clock time compared to *h*-refinement at second-order accuracy. The comparison of functional error versus wall clock time is another metric that demonstrates the efficiency of *hp*-adaptation. Recall that the wall clock time shown for the adjoint based adaptation methods includes the fully converged adjoint and flow solutions at each refinement level. In practice one would probably partially converge the flow and adjoint solutions during the early stages of adaptive refinement.

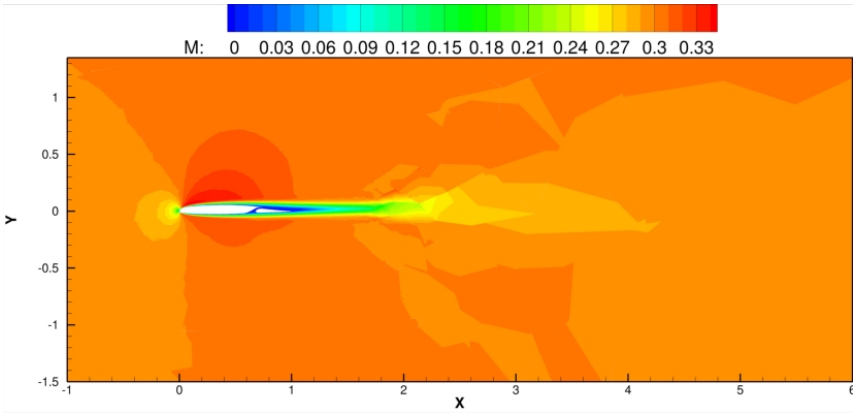
5.4.2 Two-element Airfoil: Drag-based Adaptation

The second test case considers the laminar viscous flow over a two-element airfoil with the following flow conditions, $M_\infty = .3$, $\alpha = 1^o$, and $Re = 5,000$. The initial mesh consists of $N = 6,921$ elements at a uniform discretization order of $p = 1$, which results in 22,401 DoFs. Based on the results of the previous test case, which have shown that *hp*-adaptation is the most efficient method for obtaining a grid converged output functional, this case is computed using only the *hp*-adaptive method where the maximum discretization order is fixed at $p = 5$, as with the NACA0012 airfoil case. As in the previous case, drag is chosen as the target functional for the adjoint-based adaptation.

Figures 5.18(a)-5.19(b) depict the initial and final meshes along with the computed Mach number contours on each mesh. Comparison of the computed Mach number contours in the wake region between the initial and final meshes shows that the adjoint-based *hp*-adaptation has increased the resolution in the wake, as seen by the increased distance over which the wake is captured in Figure 5.19(b). Also notice that *hp*-adaptive method did not subdivide any elements because the initial mesh is relatively fine and the smoothness indicator given by equation (5.3.2) did not detect any non-smooth phenomena, i.e. in this case changing the discretization order was sufficient to yield a grid converged result. The final mesh consists of the same number of elements as the original mesh, but with variable discretization orders ranging from $p = 1$ to $p = 5$ throughout the domain as depicted in Figure 5.19(a). Figure 5.20(a) shows the drag versus N_{DoF} for this case. Clearly a grid converged drag value has been obtained after four adaptive steps using approximately 80,000 DoFs. Figure 5.20(b)



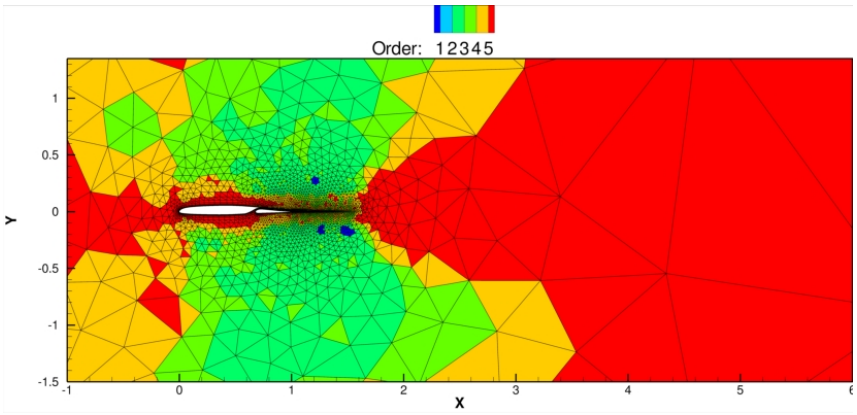
(a) Initial mesh: $N = 6,921$ elements, $p = 1$



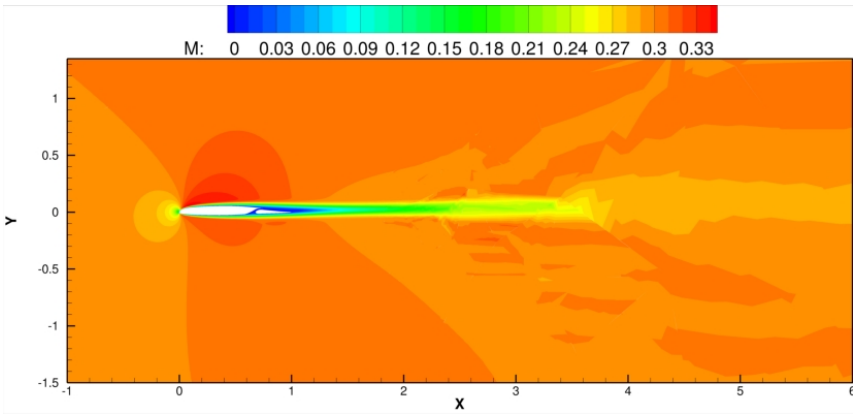
(b) Mach contours: initial mesh

Figure 5.18: Initial mesh and Mach number contours of the laminar flow over a two-element airfoil with $p = 1$, $M_\infty = .3$, $\alpha = 1^o$, and $Re = 5,000$.

shows the drag versus wall clock time for this case indicating that a grid converged drag value is generated in 53 min of computational time. Figure 5.21 depicts the convergence of the discrete flow equations using the MGPC-GMRES solver of Chapter 4 for all adaptation cycles.

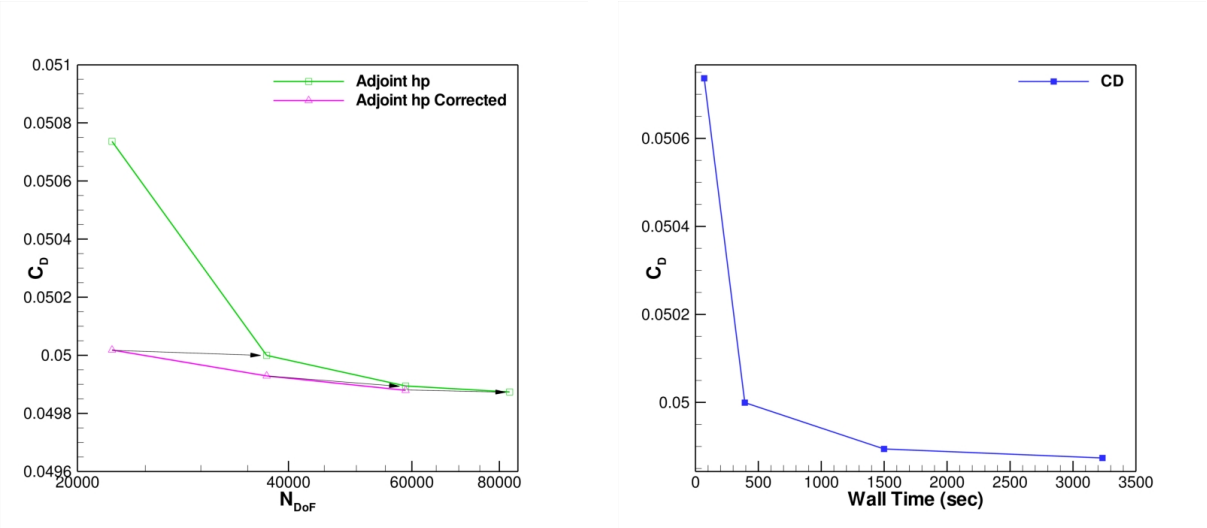


(a) Adjoint *hp*-adaptation: final mesh with $N = 6,921$, $p = 1$ to $p = 5$



(b) Adjoint *hp*-adaptation: Mach number on the final mesh

Figure 5.19: Final mesh and Mach number contours of the laminar flow over a two-element airfoil using adjoint *hp*-adaptation with $p = 1$ to $p = 5$, $M_\infty = .3$, $\alpha = 1^o$, and $Re = 5,000$.



(a) Adjoint *hp*-adaptation: drag vs. N_{DoF} (b) Adjoint *hp*-adaptation: drag vs wall clock time

Figure 5.20: Computed drag versus N_{DoF} and versus wall clock time for the laminar flow over a two-element airfoil using *hp*-adaptation.

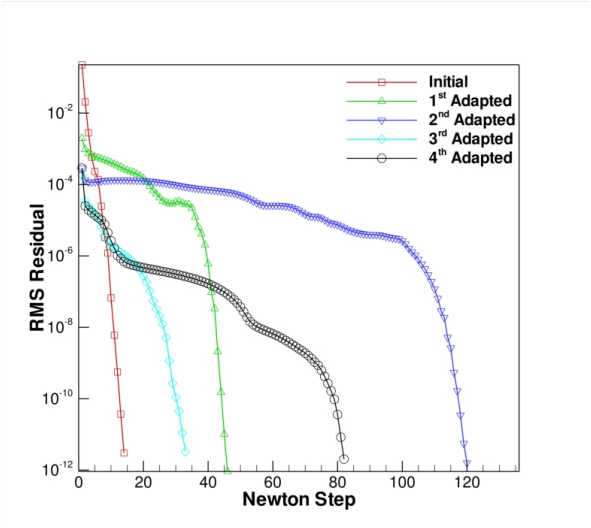


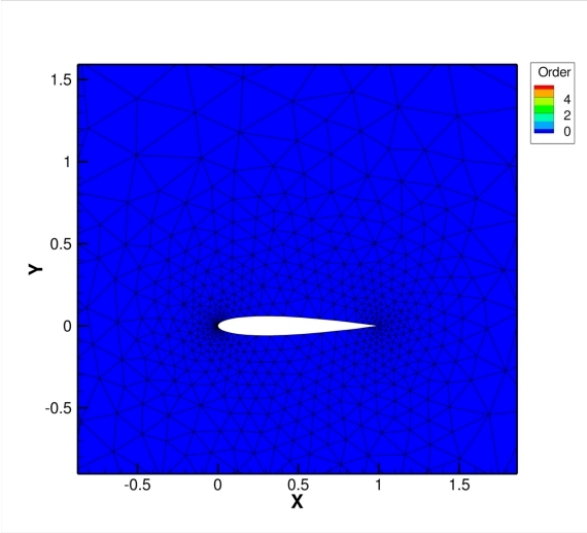
Figure 5.21: Flow solver iterative convergence for the laminar flow over a two-element airfoil.

For this test case the corrected drag coefficient values computed via equation (5.2.13) match the fine level computed drag values throughout the adaptive process. This agreement is obtained because the adaptive process was started on a much finer mesh than the NACA0012 airfoil case, hence the adaptive process was started much closer to grid convergence than the NACA0012 airfoil case. The reason that the initial mesh contains so many cells is that the small tolerances induced by the gap between the two airfoil elements forces the generation of relatively small cells in the initial mesh.

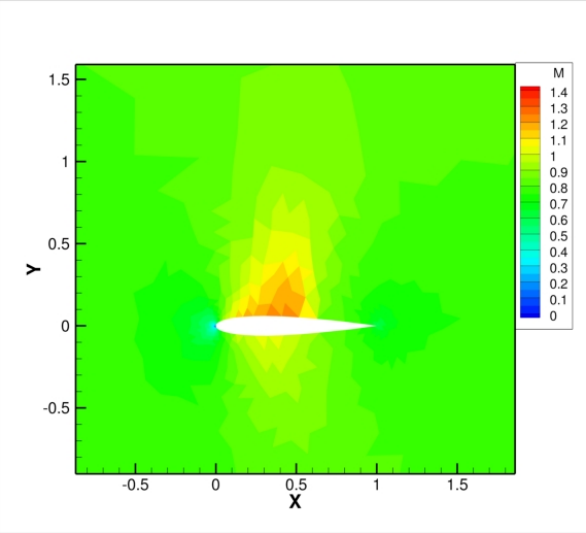
5.4.3 Inviscid Transonic NACA0012 Airfoil: Lift-based Adaptation

As an example of computing discontinuous solutions, the *hp*-adaptive method is applied to an inviscid transonic flow over a NACA0012 airfoil at $M_\infty = .8$ and $\alpha = 1.25^o$. While the previous two test cases contained smooth flow solutions, this case has both a strong and a weak shock wave. This case represents a scenario where the *hp*-adaptive approach not only yields high efficiency but also enhanced robustness. For comparative purposes three refinement scenarios are employed for this test case. In the first scenario the grid initially contains $N = 1,566$ triangles with a uniform discretization order of $p = 0$, resulting in $1,566$ DoFs, and is subsequently refined using *hp*-adaptation in the absence of any artificial diffusion. The second scenario uses uniform p-enrichment on a grid with $N = 3,086$ triangles and artificial diffusion to stabilize the high-order solutions in the presence of the shock waves. The final scenario employs *hp*-adaptation where the grid initially contains $N = 1,566$ triangles with a uniform discretization order of $p = 1$, which requires artificial diffusion to stabilize the $p = 1$ discretization in the vicinity of the shock wave. The objective function in all cases is the computed lift coefficient. A reference solution was computed using a second-order finite volume method with a $200,000$ element mesh (Courtesy Dr. Karthik Mani [19]). The goal of this case is to compare adaptation using $p = 0$ at the shock without artificial diffusion, adaptation using $p = 1$ at the shock with artificial diffusion, and uniform p-enrichment using artificial diffusion. It should be noted that using the piecewise constant artificial viscosity with high discretization orders can be difficult due to the need to change

the artificial viscosity settings from as the discretization order is varied, as mentioned in Section 2.7.4.

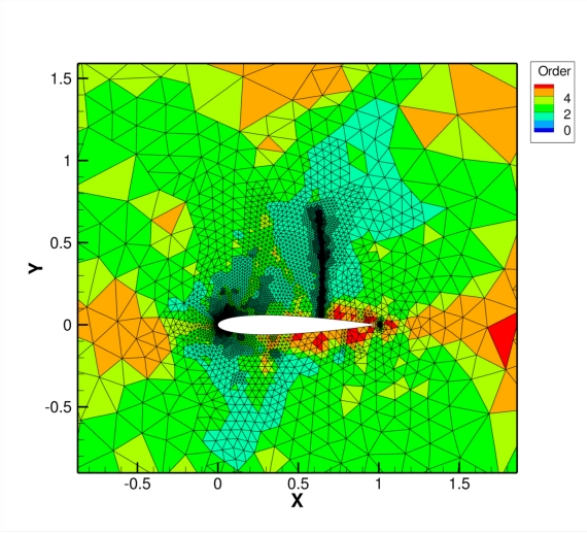


(a) Initial mesh: $N = 1,556$ elements, $p = 0$

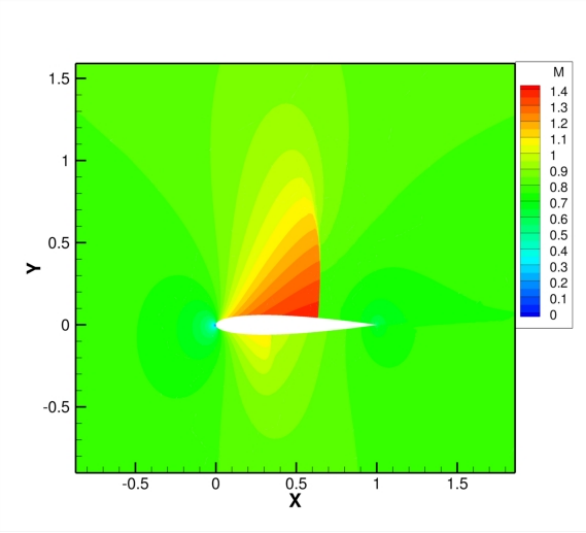


(b) Mach contours: initial mesh

Figure 5.22: Initial mesh and Mach number contours for the inviscid transonic flow over a NACA0012 airfoil with $p = 0$, $M_\infty = .8$ and $\alpha = 1.25^o$.

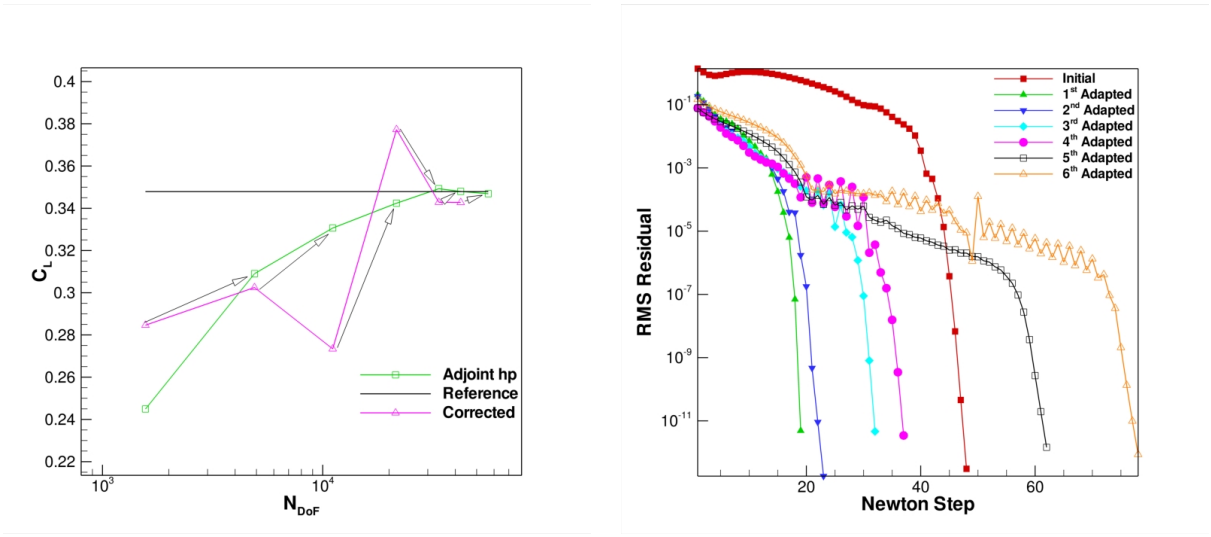


(a) Final mesh: $N = 26,407$, $p = 0$ to $p = 5$



(b) Final mesh: Mach number

Figure 5.23: Final mesh and Mach number contours on the final mesh for the inviscid transonic flow over a NACA0012 airfoil ($M_\infty = .8$ and $\alpha = 1.25^o$) using adjoint *hp*-adaptation with lift as the objective, the discretization order varies from $p = 0$ to $p = 5$.



(a) Adjoint *hp*-adaptation: lift vs. N_{DoF} (b) Iterative convergence

Figure 5.24: Computed lift coefficient versus N_{DoF} using using *hp*-adaptation without artificial viscosity and iterative convergence of the flow solver for inviscid transonic flow over a NACA0012 airfoil using the MGPC-GMRES solver.

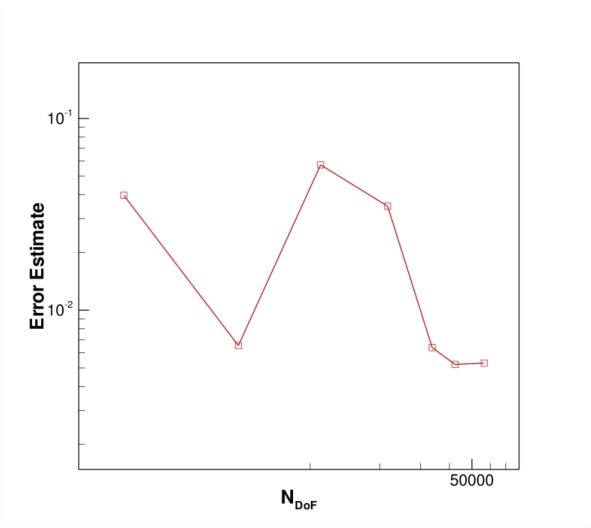


Figure 5.25: Error estimate in the computed lift coefficient over the *hp*-adaptation history employing $p = 0$ at the shock and no artificial viscosity.

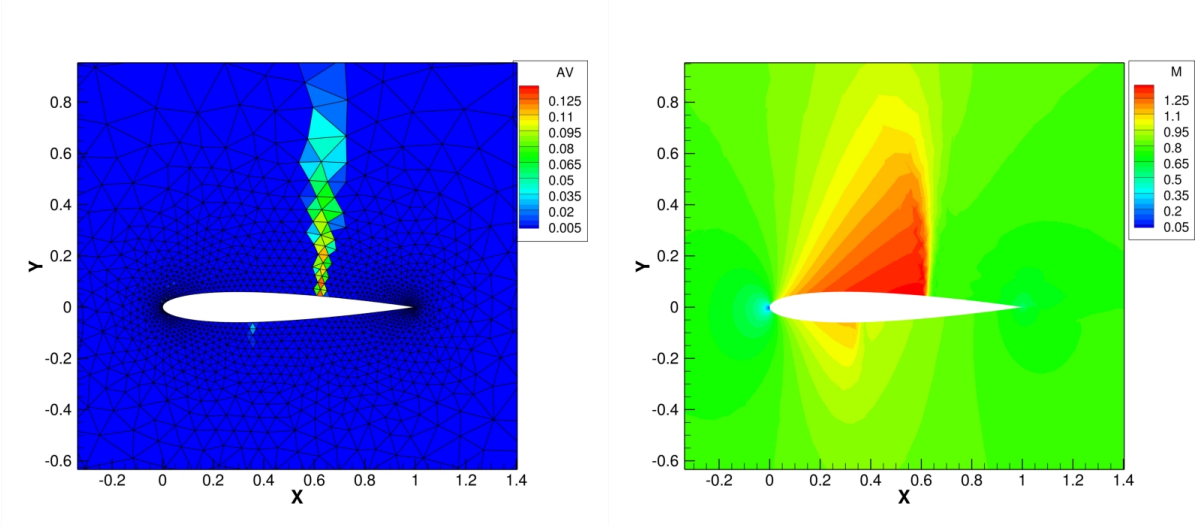
Figures 5.22(a)-5.23(b) depict the grids and computed Mach number contours for this case, at the initial and final stages of the adaptive process using *hp*-adaptation with $p = 0$ at the shock. The computed Mach number contours in Figure 5.23(b) show that the shock wave is very sharply resolved using the final *hp*-adapted mesh. Figure 5.24(a) depicts the

computed lift coefficient versus N_{DoF} and it is clear that a grid converged lift coefficient that closely matches the reference value is obtained using approximately 60,000 DoFs. The *hp*-adaptive algorithm has used less than one third the number of DoFs used to compute the reference solution. Figure 5.24(b) depicts the iterative convergence using the MGPC-GMRES solver for all adaptive cycles indicating that a fully converged solution is obtained at every stage of the adaptive process. Furthermore, this approach is relatively robust, requiring less than 100 Newton iterations for the flow solution at all adaptation cycles in Figure 5.24(b). Although there are $p = 0$ elements directly involved in the computation of the lift coefficient, a grid converged result is achieved efficiently.

While the computed the lift coefficient achieves grid convergence as shown in Figure 5.24(a), the corrected coarse level lift coefficient does not match the fine level computed lift coefficient at any point during the *hp*-adaptation proceedure. Examination of Figure 5.25, which depicts the adjoint error estimate ϵ_c in equation (5.2.10), shows that the computable error is not converging towards zero, as was the case with the previous laminar viscous test cases. This contradicts the computed lift coefficient result, which is trending toward a fixed value as seen in Figure 5.24(a). If the computed lift coefficient is trending towards a fixed value then the difference between the computed and exact lift coefficient must be decreasing as the mesh is adapted. Examination of equation (5.2.10) shows that for $|\epsilon_c|$ to decrease over the adaptation, the approximate fine level residual $\mathbf{R}_h\left(\mathbf{u}_H^h\right)$ must decrease, since the magnitude of the adjoint variable $\mathbf{\Lambda}_H^h$ cannot decrease, due the dual consistent discretization, as discussed in Chapter 3. Therefore, if $|\epsilon_c|$ is not decreasing then the residual $\mathbf{R}_h\left(\mathbf{u}_H^h\right)$ must not be decreasing. Examination of the residual norm $\parallel \mathbf{R}_h\left(\mathbf{u}_H^h\right) \parallel_2$ over the adaptation confirmed that the fine level residual estimate is not reduced during *hp*-adaptation. This is a result of using injection of the $p = 0$ solution into the $p = 1$ finite-element space to estimate the fine level solution $\mathbf{u}_H^h$, which induces Gibbs phenomena in the fine level solution estimate $\mathbf{u}_H^h$. The presence of Gibbs phenomena in $\mathbf{u}_H^h$ cause the fine level residual estimate $\mathbf{R}_h\left(\mathbf{u}_H^h\right)$ to behave irregularly as the mesh is refined. The method for removing the Gibbs phenomena from the fine level solution estimate $\mathbf{u}_H^h$ and thus correcting or regularizing the fine level residual estimate $\mathbf{R}_h\left(\mathbf{u}_H^h\right)$ is discussed subsequently.

While no explicit limiter has been used to generate these results, one can view the *hp*-adaptive approach as a form of limitation. Tradition slope limiters effectively reduce the order of accuracy locally. The idea behind slope limitation is to assume that a high-order discretization is appropriate everywhere in the grid and then to remedy those areas where a high-order discretization is not appropriate, corresponding to a top down approach. *hp*-adaptation can be viewed as a bottom up approach to limitation because *hp*-adaptation starts with a low-order discretization and moves towards a high-order discretization where appropriate. In the context of DG discretizations, the bottom up approach has an advantage because it takes the coupling between order of accuracy and resolution into account naturally, by applying *h*-refinement in the regions of the domain which are non-smooth.

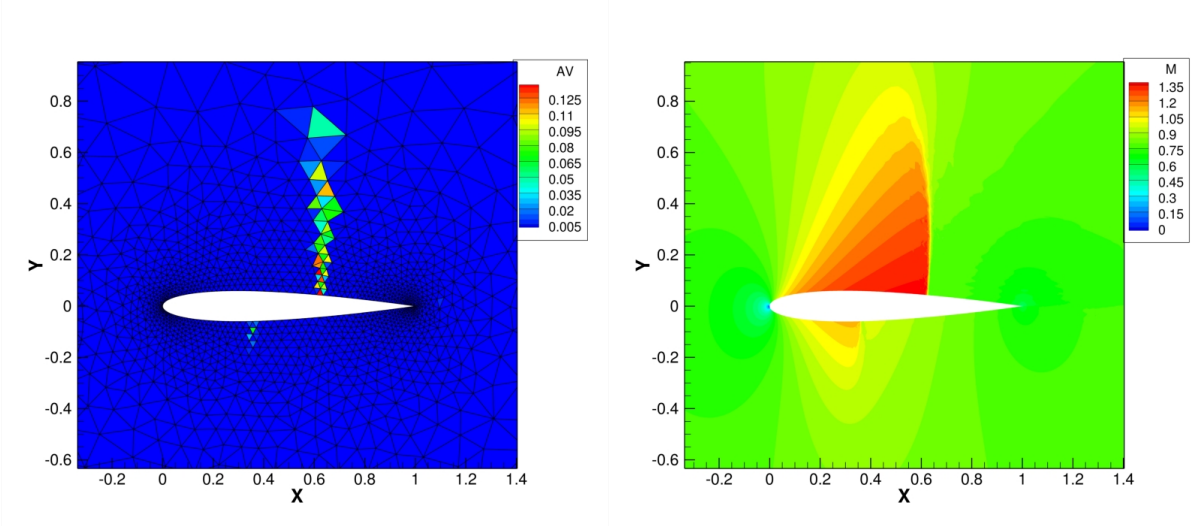
As a point of comparison, this flow is also computed using the piecewise constant artificial viscosity method from Section 2.7.1. The computations are performed using discretization orders $p = 1$ to $p = 4$ on a triangular mesh with $N = 3,189$ elements, which corresponds to approximately $10,0000$ to $50,000$ DoFs. The flow solutions are converged using a CGS preconditioned GMRES solver described in Section 4.3.3 and Section 4.5.



(a) Initial A.V. and mesh: $N = 3,086$, $p = 1$

(b) Initial mesh: Mach number

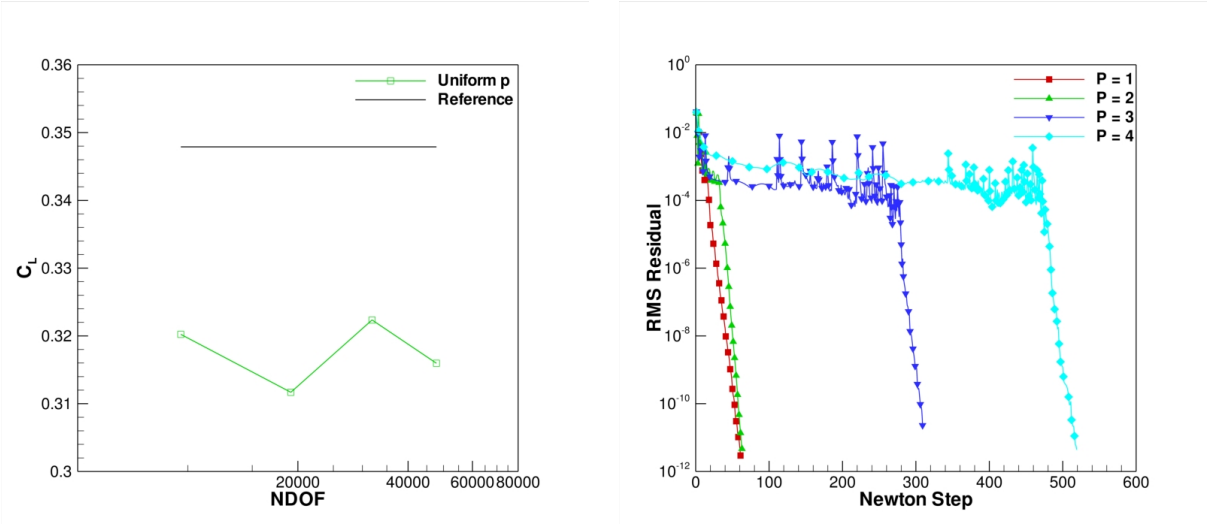
Figure 5.26: Initial artificial viscosity and Mach number contours for transonic flow over a NACA0012 airfoil with $p = 1$, $M_\infty = .8$ and $\alpha = 1.25^o$.



(a) Final A.V. and mesh: $N = 3,086$, $p = 4$

(b) Final mesh: Mach number

Figure 5.27: Final artificial viscosity and Mach number contours for transonic flow over a NACA0012 airfoil ($M_\infty = .8$ and $\alpha = 1.25^o$), using uniform p-enrichment with discretization order is $p = 4$.



(a) Lift vs. N_{DoF}

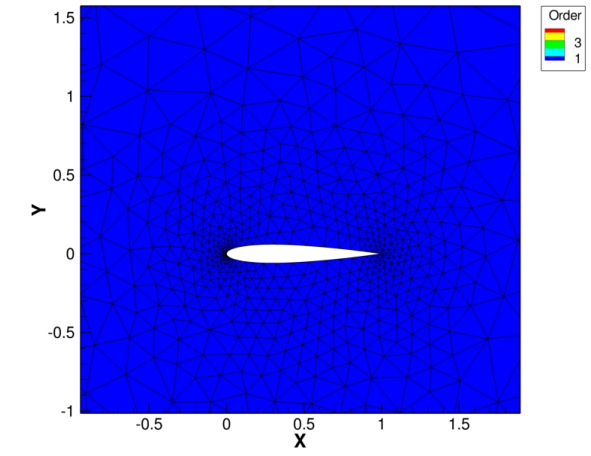
(b) Iterative convergence

Figure 5.28: Lift versus N_{DoF} for transonic flow over a NACA0012 airfoil using using artificial diffusion with $p = 1$ to $p = 4$ and iterative convergence of the flow solver using a CGS preconditioned GMRES solver.

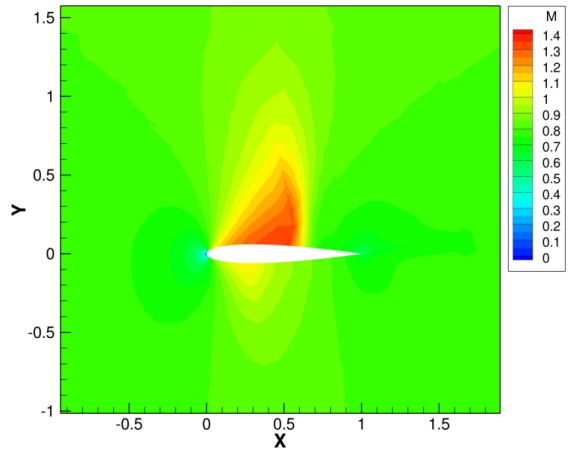
Figures 5.26(a)-5.27(b) depict the grid, artificial viscosity contours and Mach number contours for this flow, computed with uniform p-enrichment and using the piecewise constant artificial viscosity. Note that the shock wave sharpens when uniform p-enrichment is applied, indicating that increasing the discretization order p has increased the resolution of the flow field. Figure 5.28(a) depicts the lift versus N_{DoF}, illustrating that the computed lift coefficient does not converge to a fixed value as the discretization order p is increased. The conclusion is that while the resolution is certainly increased, the piecewise constant artificial viscosity has compromised the grid convergence of the higher-order result. Figure 5.28(b) depicts the iterative converge for this case using the CGS preconditioned GMRES solver. One can immediately see from the number of Newton steps required to converge the discrete flows equations(up to 500), that computing shock waves with this method can become quite expensive especially when compared with the adjoint hp-adaptation convergence history in Figure 5.24(b). Additionally, for each order of accuracy adjustments to the artificial viscosity parameters(up to a factor of 2) were required in order to obtain a convergent solution process . The variations in the artificial viscosity parameters are the root cause of the poor computed lift coefficient convergence behavior in Figure 5.28(a). This can be remedied by using a more robust artificial viscosity method as seen in Section 2.7.4.

As a third and final comparison, this test case is computed using adjoint hp-adaptation combined piecewise constant artificial viscosity and a minimum discretization order of $p = 1$. As shown in Section 4.7.1 a minimum discretization order of $p = 1$ is required for SIP based DG discretizations of the viscous terms. Therefore, it is of interest to examine the viability of employing a minimum discretization order of $p = 1$ for shocked flows. Furthermore, it is of interest to investigate if hp-adaptation can remedy the poor objective convergence observed(Figure 5.28(a)) when employing uniform p-enrichment with piecewise constant artificial viscosity for this flow.

Figures 5.29(a)-5.30(b) depict the grid and Mach number contours for this flow, computed with adjoint hp-adaptation and piecewise constant artificial viscosity. One clearly sees that h-refinement is applied in the region around the shock and p-enrichment is applied elsewhere. Figure 5.30(b) shows that the shock wave thickness is reduced dramatically during

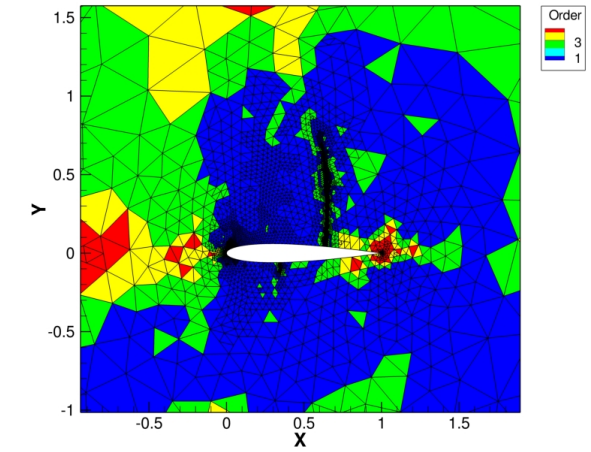


(a) Initial grid: 1,556 elements, $p = 1$

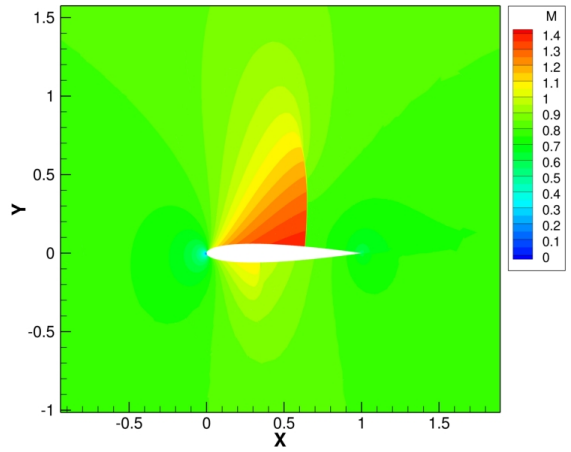


(b) Initial grid: Mach number

Figure 5.29: Initial mesh and Mach number contours for inviscid transonic flow over a NACA0012 airfoil with $p = 1$ and artificial diffusion, $M_\infty = .8$ and $\alpha = 1.25^o$.



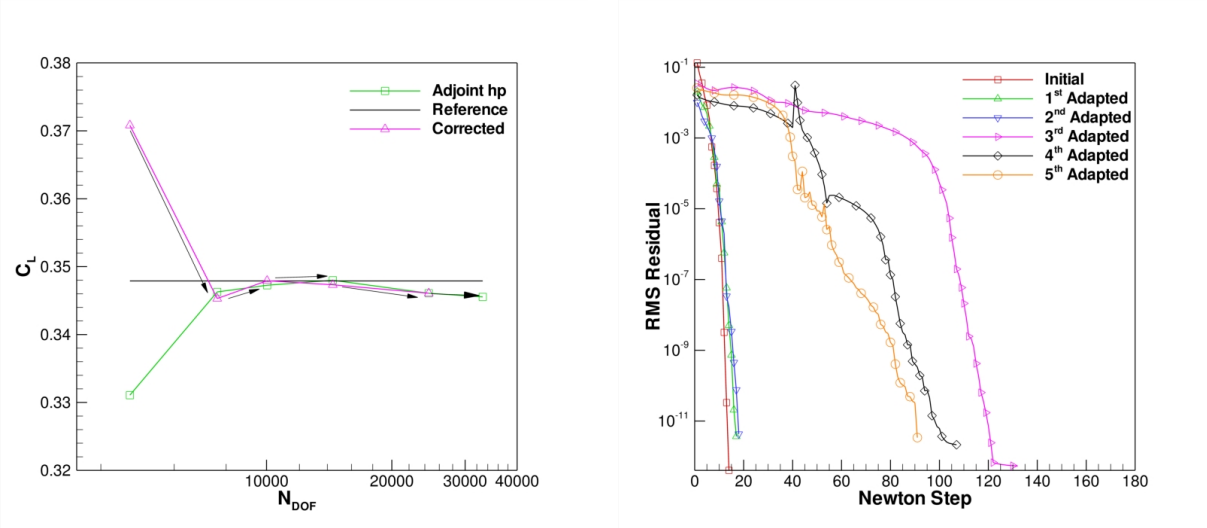
(a) Final mesh: with $N = 8,346$, $p = 1$ to $p = 4$



(b) Final mesh: Mach number

Figure 5.30: Final mesh and Mach number contours for inviscid transonic flow over a NACA0012 ($M_\infty = .8$ and $\alpha = 1.25^o$) airfoil using adjoint *hp*-adaptation and piecewise constant artificial viscosity, the discretization order varies from $p = 1$ to $p = 4$.

the adaptive process. Figure 5.31(a) shows the lift versus N_{DoF} for this case, where one can immediately notice that *hp*-adaptation remedies the functional convergence problems shown in Figure 5.28(a). Employing *h*-refinement rather than *p*-enrichment in regions where



(a) Adjoint *hp*-adaptation: lift vs. N_{DoF}

(b) Iterative convergence

Figure 5.31: Inviscid transonic NACA0012 airfoil: computed lift coefficient versus N_{DoF} using using *hp*-adaptation combined with piecewise constant artificial viscosity. Iterative convergence using the CGS preconditioned GMRES solver.

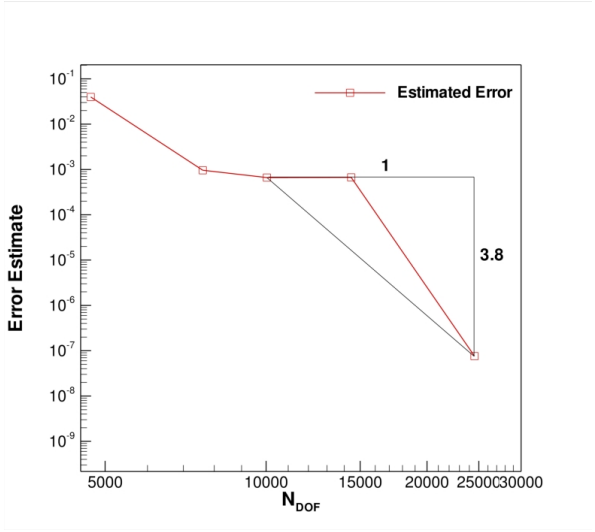


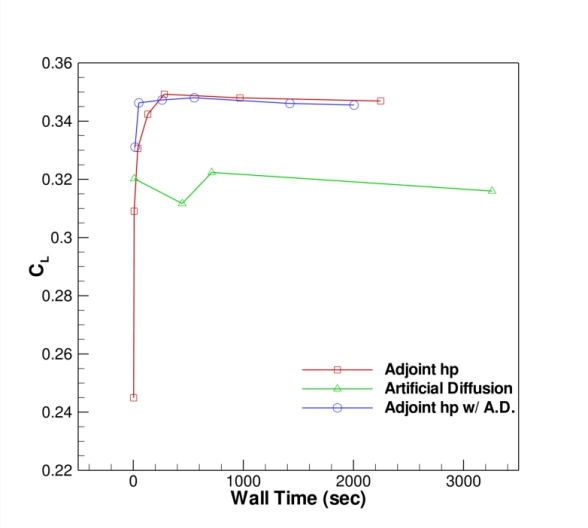
Figure 5.32: Error estimate in the computed lift coefficient over the *hp*-adaptation history employing a discretization order of $p = 1$ and piecewise constant artificial viscosity in the vicinity of the shock wave.

artificial viscosity is applied results this improved functional convergence behavior for this test case. The localized mesh refinement in the vicinity of the shock wave allows the artificial viscosity coefficients (Section 2.7.1) to remain fixed throughout the refinement process, which is significantly more effective at increasing functional convergence than *p*-enrichment, which

requires altering these coefficients as refinement is applied. Comparison of Figure 5.31(a) and Figure 5.24(a) shows that using *hp*-adaptation with a minimum discretization order of $p = 1$ combined with artificial viscosity uses significantly fewer degrees of freedom than *hp*-adaptation using $p = 0$ as the minimum discretization order. This improvement in efficiency is due to the uniform second-order accuracy in the regions of smooth flow features, resulting in a significantly better initial solution. Thus *hp*-adaptation with artificial diffusion is even more efficient than *hp*-adaptation using $p = 0$ to resolve the shock wave. Figure 5.31(b) depicts the iterative convergence of the flow solver using the CGS preconditioned GMRES solver. Note that the *hp*-adaptation results require far fewer Newton steps than the uniform *p*-enrichment results(Figure 5.28(b)), especially for the final refinement.

Furthermore, combining artificial viscosity with *hp*-adaptation has resulted in improved agreement between coarse level corrected lift coefficients (equation (5.2.13)) and the fine level computed lift coefficient as seen in Figure 5.31(a). Examination of the computed lift error estimate in Figure 5.32 shows that the error estimate is decreasing over the adaptation history. Comparing Figure 5.32 with Figure 5.25 shows that the addition of artificial viscosity has significantly improved the behavior of the error estimate. The addition of artificial viscosity eliminates the Gibbs phenomena in the fine level solution estimate $\mathbf{u}_H^h$, allowing the fine level residual estimate $\mathbf{R}_h\left(\mathbf{u}_H^h\right)$ in equation (5.2.10) to recover a decreasing trend as *hp*-adaptation is applied. Therefore artificial viscosity is a suitable regularization technique for the fine level solution estimate $\mathbf{u}_H^h$, allowing for accurate error estimation.

Figure 5.33(a) depicts the computed lift coefficient versus the wall clock time, which clearly shows that *hp*-adaptation generates more accurate lift coefficient values at a reduced cost compared to higher-order shock capturing with piecewise constant artificial viscosity. However, the most efficient method involves combining the piecewise constant artificial viscosity with *hp*-adaptation employing a minimum discretization order of $p = 1$. This combination produces a grid converged functional in the least amount of computational time as shown in Figure 5.33(a). Furthermore, *hp*-adaptation has remedied the poor functional convergence observed for the piecewise constant artificial viscosity with uniform *p*-enrichment computations.



(a) Lift vs. wall clock time

Figure 5.33: Comparison of the computed lift versus wall clock time for inviscid flow over a NACA0012 airfoil using using piecewise constant artificial viscosity with uniform p-enrichment and hp-adaptation.

The combination of piecewise constant artificial viscosity and hp-adaptation has proven to be more robust than either of these methods in isolation. Employing hp-adaptation without artificial viscosity requires that the shock wave not enter a high-order element during the solution process. However, the combination of hp-adaptation and piecewise constant artificial viscosity allows this constraint to be relaxed because if a shock wave were to move into a higher-order element during the flow solution process, the artificial viscosity will become active and stabilize the element. Therefore, when considering the robustness of combining hp-adaptation and artificial viscosity, no special care must be taken to avoid shock waves entering high-order elements. However, the results indicate that additional accuracy is achieved if h-refinement is employed in the vicinity of the shock wave. Therefore when the combination of hp-adaptation and piecewise constant artificial viscosity is employed, the decision between h-refinement and p-enrichment in equation (5.3.3) should be made such that elements with non-zero artificial viscosity values are targeted with h-refinement. This can be accomplished if the resolution indicator of equation (2.7.2) is used as the hp-adaptation smoothness indicator and $\frac{1}{\mathcal{K}}$ in equation (5.3.3) is set as $\frac{1}{\mathcal{K}} = s_0 - \kappa$, which will target all elements with non-zero artificial viscosity using h-refinement.

While the *hp*-adaptive approach is not the most elegant shock capturing method for DG discretizations, *hp*-adaptation has some significant advantages; *hp*-adaptation gives robust and fast iterative convergence, and with each refinement the functional accuracy improves and eventually grid converge of the functional is achieved. The results indicate that the piecewise constant artificial viscosity should be combined with at least h-refinement at a minimum discretization order of $p = 1$, in order to achieve grid converged functional values. In particular, this work has shown *hp*-adaptation to be a very effective choice when used in combination with artificial viscosity. Based on these results, the recommended strategy for shock capturing is to combine *hp*-adaptation with artificial viscosity and to maintain the discretization order at $p = 1$ in the vicinity of the shock wave.

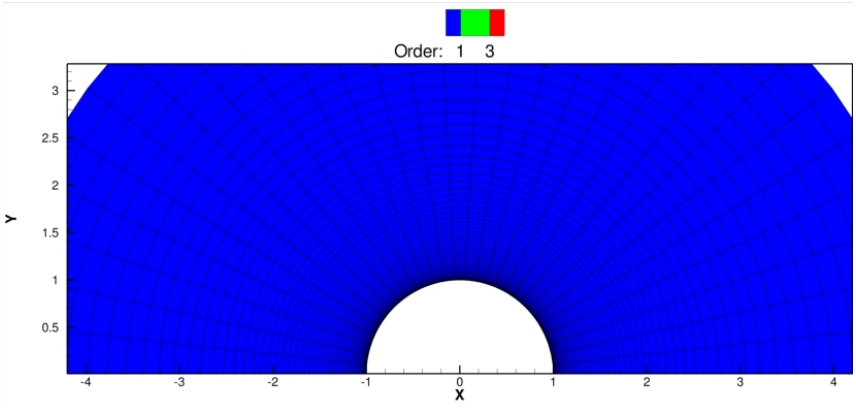
5.4.4 Supersonic Viscous Cylinder: Surface Heating Based Adaptation

The fourth and final test case considers supersonic viscous flow over a half cylinder geometry. The flow conditions are $M_\infty = 3.0$, $\alpha = 270^o$, and $Re = 10,000$. The initial mesh consists of $N = 1,711$ quadrilateral elements and is initialized to a uniform discretization order of $p = 1$, which results in 6,844 DoFs. Since the end goal of this work is to develop a robust and accurate high-order flow solver, this case is designed to test the *hp*-adaptation strategy for viscous shocked flows. In this case the piecewise constant artificial viscosity of Section 2.7.1 is employed as the shock capturing method. As previous test cases have shown, using high discretization order(p) combined with piecewise constant artificial viscosity has proven to be ineffective at reducing functional error. Hence adjoint based *hp*-adaptation is employed to simultaneously increase solver robustness and accuracy. In this case the target of the adjoint-based adaptation is the surface heating coefficient C_H defined as:

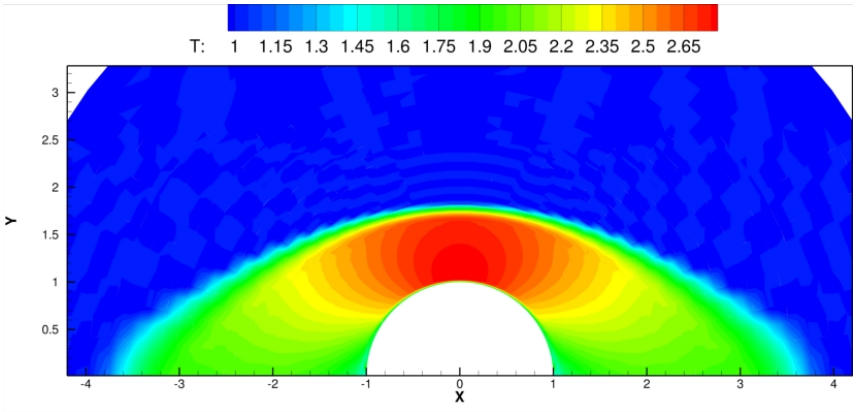
$$\begin{aligned} C_H &= \frac{\mu_b}{P_r} \frac{\nabla T_b \cdot \vec{n}}{\frac{1}{2}\rho_b U_\infty^3} \\ U_\infty &= \sqrt{u_\infty^2 + v_\infty^2} \end{aligned} \tag{5.4.1}$$

where $P_r = .72$ is the Prandtl number, μ_b is the viscosity at the surface, T_b is the surface temperature, $\vec{n}$ is the surface normal vector, ρ_b is the surface density, u_∞ is the free-stream

u-velocity, and v_∞ is the free-stream v-velocity. Furthermore, $p = 3$ is set as the maximum allowable discretization order for this case.



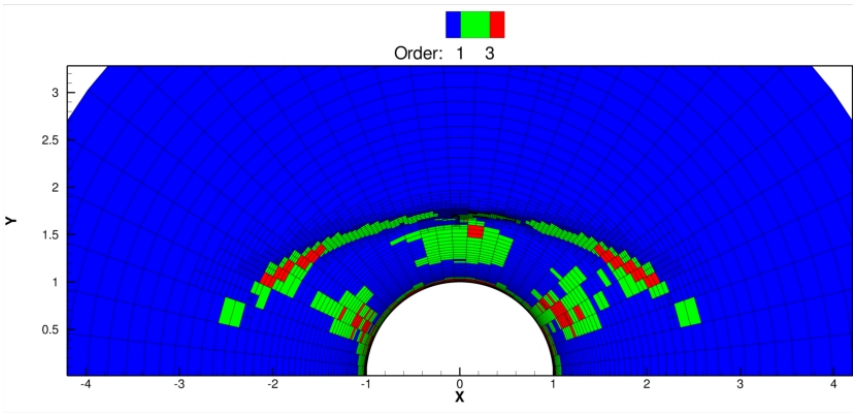
(a) Initial mesh: $N = 1,711$ elements, $p = 1$



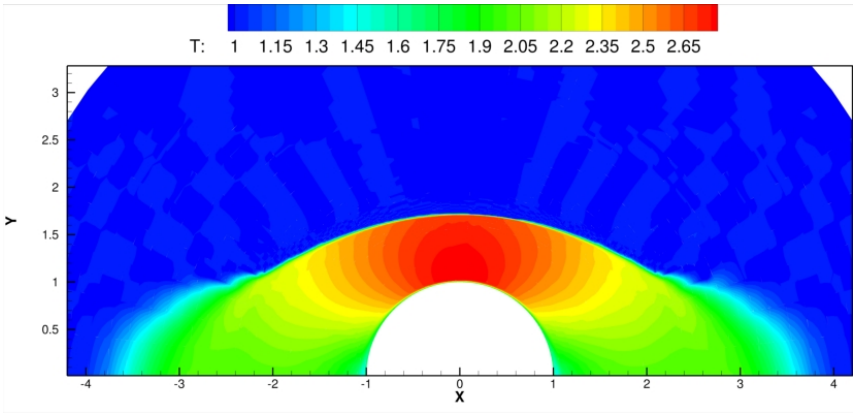
(b) Initial mesh: temperature contours

Figure 5.34: Initial mesh and temperature contours for supersonic viscous flow over a half cylinder using a uniform discretization order of $p = 1$.

Figure 5.34(a) through Figure 5.35(b) depict the meshes and temperature distributions at the initial and final stages of the *hp*-adaptive process, which show that a substantially more resolved shock wave is obtained on the final *hp*-adapted mesh. The surface heating objective has targeted only a portion of the shock wave, which is refined using h-refinement as shown in Figure 5.35(a). Similarly, a portion of the region behind the shock wave, known as the shock layer, is targeted for refinement using p-enrichment due to the smooth flow features in this region. Lastly, the boundary layer along the surface of the cylinder is also targeted using p-enrichment since this is also a smooth flow feature. The portion of the shock wave



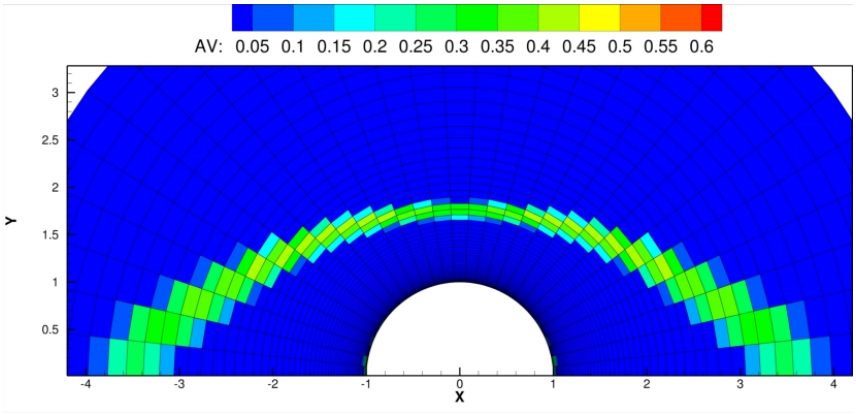
(a) Adjoint *hp*-adaptation final mesh: $N = 6,253$, $p = 1$ to $p = 3$



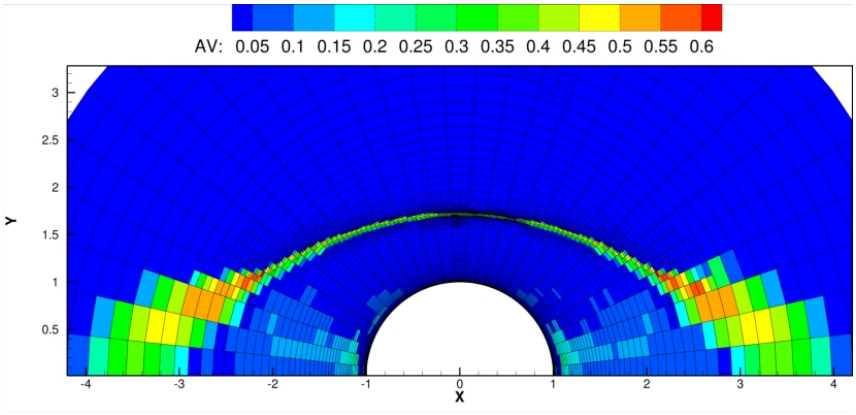
(b) Final mesh: temperature contours

Figure 5.35: Final mesh and temperature contours for supersonic viscous flow over a half cylinder, the discretization order varies from $p = 1$ to $p = 3$.

that is relevant to surface heating is captured very sharply and robustly by *hp*-adaptation. Figure 5.37 depicts the temperature extracted along the stagnation streamline on the intial and final *hp*-adapted meshes and shows that *hp*-adaptation has increased the resolution of the shock wave significantly on the final mesh. Figure 5.36(a) depicts the artificial viscosity distribution on the initial mesh and Figure 5.36(b) depicts the artificial viscosity distribution on the final mesh. On the initial mesh the artificial viscosity targeted only elements in the vicinity of the shock wave and vanished in regions of smooth flow features. Figure 5.36(b) shows that on the final adapted mesh the artificial viscosity has been activated in the vicinity of the shock wave as well as in the shock layer near the $y = 0$ lines. Comparing Figure 5.36(a) and Figure 5.36(b) illustrates that as *h*-refinement is applied in the vicinity of the $x = 0$



(a) Initial mesh: artificial viscosity



(b) Final mesh: artificial viscosity

Figure 5.36: Artificial viscosity on initial and final meshes for supersonic viscous flow over a half cylinder.

line, the artificial viscosity is confined to a thinner region than on the initial mesh.

Figure 5.38(a) depicts the integrated surface heating over the adaptation history, which becomes grid converged after 5 adaptive cycles. The corrected coarse level surface heating given by equation (5.2.13) correctly predicts the fine level surface heating for all but the first two adaptive cycles. Furthermore, the application of *hp*-adaptation has substantially reduced the adjoint error estimate over the adaptation process as shown in Figure 5.38(b). Figure 5.38(b) depicts the computed functional error estimate versus N_{DoF}(i.e. h^2). The convergence rate of the function error versus $h = \sqrt{N_{DoF}}$ is 6.2 and is very close to the optimal value of 6.0. This combination of *hp*-adaptation and piecewise-constant artificial viscosity is both robust and accurate and clearly demonstrates the advantages of using high-

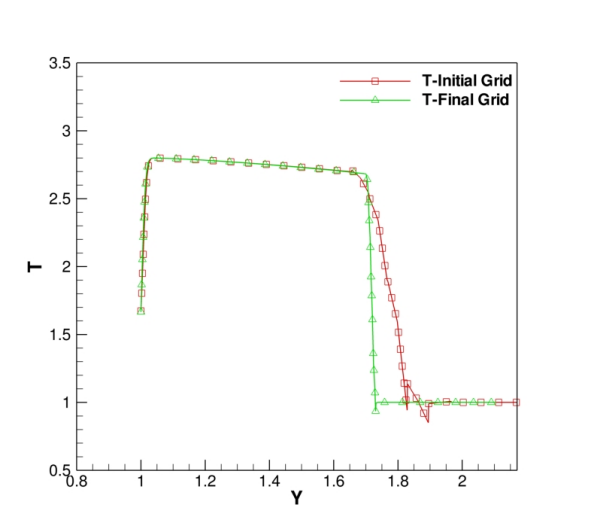
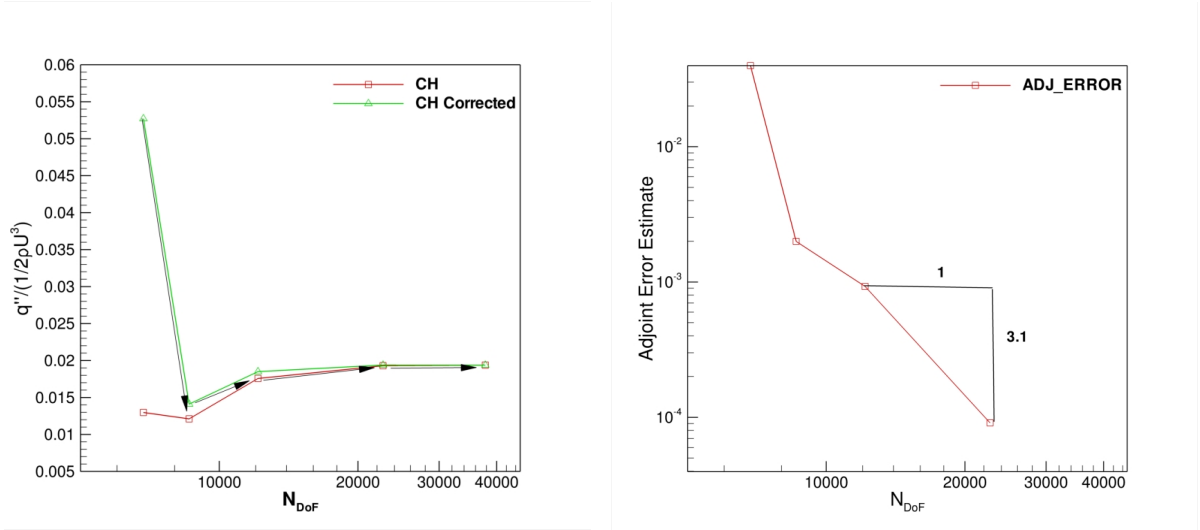


Figure 5.37: Temperature profile extracted along the stagnation streamline on the initial and final *hp*-adapted meshes.



(a) Computed surface heating: C_H (b) Computed surface heating error estimate

Figure 5.38: Computed surface heating and adjoint error estimate of computed surface over the *hp*-adaptation history.

order methods in this fashion, where as one achieves the desired functional error convergence properties without using high-order elements in the vicinity of the shock wave.

5.5 Summary

An *hp*-adaptive high-order discontinuous Galerkin solver for the Navier-Stokes equations has been developed and applied to four test cases. The adaptive method presented is driven by a goal-oriented approach which makes use of adjoint-based error estimation and is capable of adapting both the grid and discretization order locally. The solver adapts the grid non-conformally to allow for *h*-refinement of mixed-element meshes such as those shown in the numerical results. The use of *hp*-adaptation has demonstrated high efficiency by computing high accuracy functionals using fewer degrees of freedom than the reference solution or uniform refinement solutions. Furthermore, applications of *hp*-adaptation to transonic and supersonic flows has demonstrated the robustness of this method for shock capturing. The use of *hp*-adaptation is essential for obtaining grid convergence of functionals for flows with shock waves when the piecewise constant artificial viscosity method is employed.

While *hp*-adaptation alone is a form of limitation, the best overall results are shown when both *hp*-adaptation and piecewise constant artificial viscosity are combined. A supersonic test case demonstrates that high accuracy surface heating can be obtained with shock waves captured using $p = 1$ elements. While these results employ the piecewise constant artificial viscosity, the remainder of this work employs the PDE-based artificial viscosity of Section 2.7.2. The PDE-based artificial viscosity is considered for the remainder of this work because it has proven to be more robust for Mach numbers higher than $M_\infty = 3$. The PDE-based method is more robust because this method spreads the artificial viscosity distribution over a wider region than the piecewise constant method and also results in a smooth artificial viscosity distribution. Numerical experiments have shown that a smooth artificial viscosity distribution is critical to robust shock capturing as also pointed out in references [37, 69].

Chapter 6

Application of DG to Turbulent Flows using the RANS Equations

In this work, a robust discontinuous Galerkin (DG) solver for turbulent aerodynamic flows using the turbulence model of Spalart and Allmaras(SA) [41] is developed. The SA turbulence model equation, which governs the so-called turbulence model working variable $\tilde{\nu}$, is given in equation (2.1.2). Initial attempts to solve the SA turbulence model equation using a high-order DG discretization resulted in solver failure due to robustness issues. The most pressing robustness issue is related to an artificial sharp interface or discontinuity in the turbulence model working variable. As such, this work focused on improving the robustness and efficiency of the discontinuous Galerkin solver for turbulent flows governed by the Reynolds Averaged Navier-Stokes(RANS) equations coupled to the one equation turbulence model of Spalart and Allmaras. Herein a first-order finite-volume discretization is implemented for the turbulence model convection term, which is a standard practice in the finite-volume methods context. Section 2.6 describes the finite-volume discretization of the SA turbulence model equation. Computational results will show that, despite the first-order discretization of the turbulence model convection term, there is still benefit to employing high-order DG discretizations for the mean flow equations, and, at the very least, high-order accurate discontinuous Galerkin(DG) solutions to the RANS equations are obtained robustly. The combination of a high-order DG discretization for the RANS (mean

flow) equations and the first-order finite-volume discretization of the SA turbulence model equation will be denoted as a hybrid discretization in this work. The hybrid discretization is applied to realistic aerodynamic flows including a subsonic turbulent airfoil flow and two different high-lift multi-element airfoil configurations at high angles of attack.

6.1 Issues Facing RANS and High-order Methods

Non-smooth solutions present a significant challenge to discontinuous Galerkin discretizations. The most pressing source of non-smooth behavior is associated with the turbulence models used to close the Reynolds Averaged Navier-Stokes (RANS) equations. Recent work has shown that higher than first-order accurate discretizations of the convection term of the turbulence model of Spalart and Allmaras(SA)(equation (2.1.2)) produce a non-smooth behavior or a discontinuity in the turbulence model working variable $\tilde{\nu}$. The working variable discontinuity is located near the edge of boundary layers and wakes. This same phenomena has also been observed by Oliver and Darmofal in references [20, 46, 74]. The discontinuity results in oscillations of the high-order solution of the turbulence model equation, which leads to negative values of the working variable that can easily cause the presented DG solver to fail (i.e diverge). Numerical experiments with the presented DG solver have shown that some flows are more susceptible to solver failure than others. For example, solver failure is more likely to occur to for high-lift configurations, where the negative values of the working variable have particularly high magnitudes than for simple flat-plat boundary layer flows. In general, the higher the magnitude of the negative working variable values, the more likely the solver is to fail due to the turbulence model working variable discontinuity. However, results from the AIAA drag prediction workshops have shown that discretization error negatively impacts the state-of-the-art of second-order finite-volume CFD solvers [13, 14]. However, it is well known that one of the most effective methods for removing discretization error is to increase the order of accuracy [17, 20, 29, 47, 56, 96]. Therefore competing interests exist for the computation of turbulent flows. On one hand additional accuracy is required and on the other hand the turbulence model equations have so far proven difficult to solve robustly

using higher than a first-order discretization of the convection term. Hence the primary goal of this work is to examine whether increasing the mean flow discretization order, while maintaining a first-order discretization for the turbulence model equation, constitutes a viable strategy. Additionally, the robustness of the current solver will be demonstrated by applying the solver to challenging test cases that are relevant to aerodynamics. The DG solver in this work is capable of discretizing the SA model equation using both a finite-volume discretization with a first-order accurate convection term discretization and an arbitrarily high-order DG discretization, enabling comparisons between these two discretizations. The first-order finite-volume discretization is described in Section 2.6 and the DG discretization, which is also applied to the mean flow equations is described in Section 2.2.

While there are a several examples of successful high-order DG RANS solutions [15, 20, 42, 43, 70], significant robustness issues remain. For example, the flow over a flat-plate can be computed successfully using a high-order DG discretization of the SA turbulence model equation as in Section 6.2.1. However, the present DG solver has never been able to solve the flow over a high-lift multi-element airfoil configuration, such as the three-element airfoil configuration considered in Section 6.5.2, using a high-order DG discretization of the SA turbulence model equation. Furthermore, the current solver using a DG discretization of the SA turbulence model equation is able to replicate nearly all the results of references [15,20,42, 43,70]. However, high-order DG discretizations of the SA turbulence model equation are not robust enough for high-lift calculations on arbitrary grids. Many production level solvers such as, CFL3D [66], FUN3D [65], and NSU3D [64] employ finite-volume discretizations of the turbulence model equations with first-order accurate convection terms for all implemented turbulence models. Furthermore, Spalart and Allmaras employ a first-order convection term discretization of the SA turbulence model equation in reference [41]. Borrowing from this idea, the presented DG solver is modified to use the same discretization of the turbulence model equation as references [5, 64]. High-order DG discretizations are implemented for the mean flow equations in order to remove as much discretization error from the mean flow equations as possible.

There have been a few attempts to stabilize high-order solutions of the SA turbulence

model equations [15, 46, 70]. Unfortunately, these stabilization methods have proven unsatisfactory in one way or another: either the stabilization method allows for negative values of the turbulence model working variable $\tilde{\nu}$ or it adversely impacts the solution accuracy of turbulence model. In particular, one effect is the under-production of the turbulence model working variable, which results in eddy viscosity values that are too low to utilize in the Boussinesq approximation, which is hereby referred to as inadequately modeling turbulent flow physics. Numerical experiments using the present DG solver and the turbulence model stabilization methods of references [15, 46, 70] have shown that these stabilization methods suffer from the effects described above.

In order to determine whether the discontinuity in the the SA working variable is a purely numerical artifact resulting from DG discretizations, tests using a finite-volume solver are conducted in order to determine whether a higher than first-order accurate discretization of the turbulence model may be employed in the finite-volume context. Using the finite-volume solver as a base-line, the turbulence model discretization options and the manner in which these options affect the possibility of discretizing the SA turbulence model equation with high-order DG methods are discussed. Specifically, the choice of the convective numerical flux discretization is analyzed in detail to determine the most appropriate convective numerical flux for the hybrid discretization.

Just as perplexing is the determination of the optimum strategy for the coupling of the turbulence model and mean flow equations. If the turbulence model is solved using a decoupled flow Jacobian, there are some techniques [41] that can be used to help with the SA working variable discontinuity. However, using a decoupled flow Jacobian may prevent the solver from being able to fully converge the discrete equations as seen in this work and reference [97]. Fully converging the discrete equations is a very important issue for higher-order discretizations because the discrete equations must be converged to tight tolerances(usually 9 or more orders of magnitude), in order to ensure that the solver outputs such as computed lift coefficient are computed to the desired error tolerances. For example, the first AIAA international high-order methods workshop has specified an error tolerance of .01 counts ($1.0e^{-6}$) in the computed lift, drag, or moment coefficients for most test cases.

6.2 DG Discretizations of the Mean Flow and Turbulence Model

Although solving the turbulence model equation using high-order DG discretization is very difficult due to the presence of the turbulence model working variable non-smooth behavior, it is possible to obtain a solution in isolated incidences. In particular, it is possible to obtain high-order DG solutions to the turbulence model equation for simple flows such as a flat plate or an airfoil at low angles of attack. Numerical experience with computing high-order discretization turbulence model solutions has shown that, as the angle of attack, Mach number and Reynolds number are increased, the turbulence model discontinuity becomes stronger, resulting in higher magnitude negative values of the working variable. Higher magnitude negative values of the turbulence model working variable more readily cause solver failure. This section details the preliminary results of applying high-order DG discretizations of the mean flow and turbulence model equations to flow problems where a solution can be obtained. Additionally, the local-order reduction technique of Section 4.6 is applied to asses the merits of using this technique to enhance the robustness of the DG discretization of the turbulence model equation. The results presented are computed such that the turbulence model and mean flow equations all have the same discretization order, as well as the same convective numerical flux function for an element. This is the most rigorous option for discretizing the total RANS-SA system.

6.2.1 Turbulent Flat-Plate

The first test case consists of the the incompressible zero pressure gradient turbulent flow over a semi-infinite flat plate at $M_\infty = .1$, $\alpha = 0^o$, and $Re = 10,000,000$. In this case the RANS equations are coupled to the one-equation turbulence model of Spalart and Allmaras (SA model). The computational mesh is made up of $N = 540$ quadrilateral elements, as shown in Figure 6.1 and employs discretization orders $p = 1$ to $p = 4$. The computed results are compared with experimental data [98], which represents a verification of the RANS implementation for discretization orders $p = 1$ to $p = 4$. The MGPC-GMRES solver is used

to solve the discrete flow equations.

The u^+-velocity profiles versus y^+ for all discretization orders are plotted along with the experimental u^+-velocity data from reference [98] in Figure 6.2(a). The non-dimensional velocity u^+ and the non-dimensional coordinate perpendicular to the wall y^+ are given by:

$$\begin{aligned} y^+ &= \frac{u_\tau y}{\frac{\mu}{\rho}} \\ u^+ &= \frac{u}{u_\tau} \\ u_\tau &= \sqrt{\frac{\left(\frac{\mu}{\rho} + \frac{\mu_T}{\rho}\right)\frac{\partial u}{\partial y}}{\rho}}\Bigg|_{wall} \end{aligned} \tag{6.2.1}$$

where the term u_τ is the so-called friction velocity defined via the total wall shear stress. Additionally, the computed skin friction coefficient as well as experimental skin friction coefficient data versus x/c are plotted in Figure 6.2(b). The results show that good agreement between computed and experimental results is obtained for both the u^+-velocity profile at the mid-chord of the plate $x/c = .5$ and the skin friction coefficient along the plate length (i.e. versus x/c). Figure 6.3 shows the turbulence model working variable plotted versus y/c at the mid-chord of the plate $x/c = .5$ for discretization orders $p = 1$ and $p = 4$. Note the oscillations at the boundary layer edge, indicating the presence of non-smooth behavior, with more oscillatory behavior for the $p = 4$ result. References [20, 74] have also noted that high-order discretizations of the SA turbulence model equation yield oscillations at the boundary layer edge.

The convergence history for a discretization order of $p = 2$ is depicted in Figure 6.4. The MGPC-GMRES solver achieves convergence in under 80 Newton iterations for both the flow and turbulence model equations. Note the very sharp rise in the turbulence model residual around the 50^{th} Newton iteration. This is the point in the solution convergence history when the model begins to develop negative working variable values and the source term modifications given in Section B.1 become active. If not properly damped, this secondary transient will cause solver failure. Despite employing the modifications to the source terms in Section B.1, negative working variable values are still present in the final converged solution.

Furthermore, the secondary transient has proven difficult to overcome for more complex flow problems.

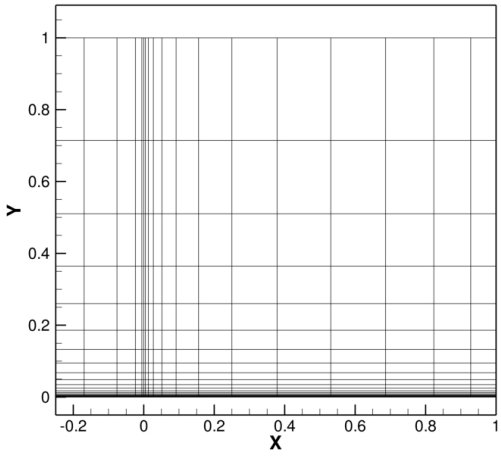
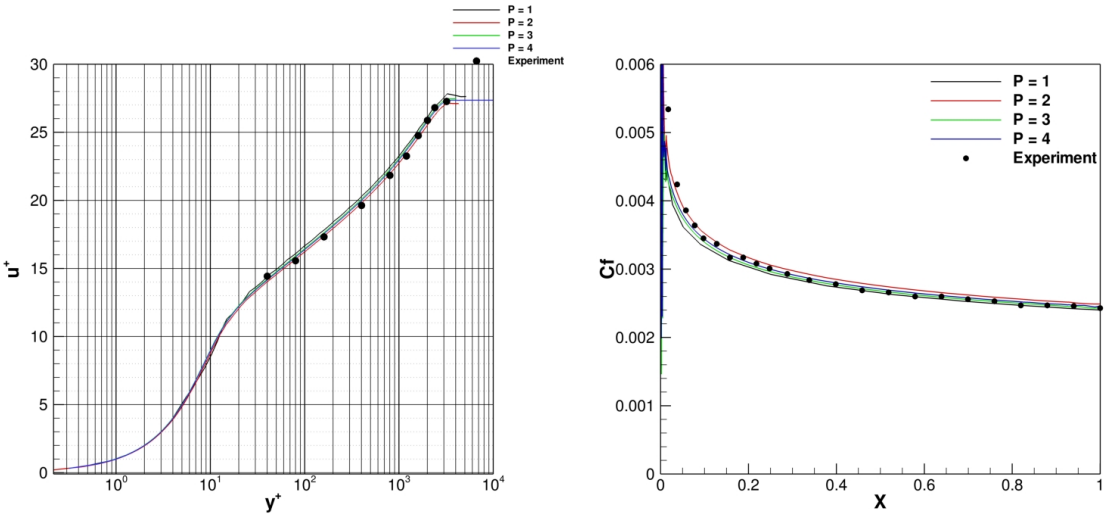


Figure 6.1: Computation mesh used for computing turbulent flow over a flat plate with the Spalart-Allmaras turbulence model consisting of $N = 540$ quadrilaterals.



(a) u-velocity profile

(b) skin-friction coefficient

Figure 6.2: Comparison of computed solution using a DG discretization with the Spalart-Allmaras turbulence model for a flat plate boundary layer compared with experimental data using $p = 1$ up to $p = 4$.

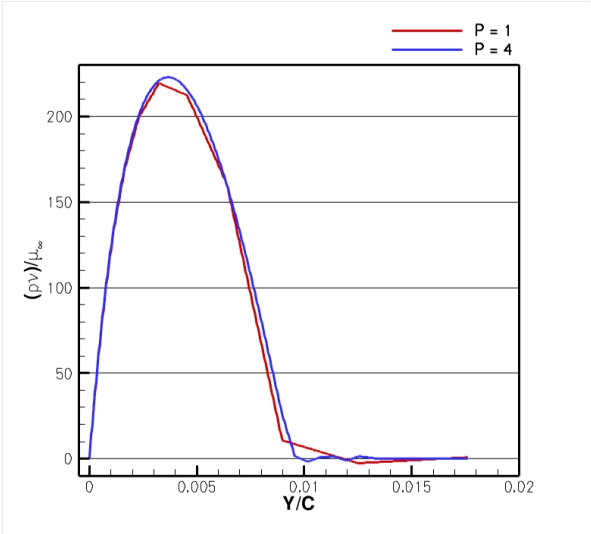


Figure 6.3: Computed profile of Spalart Allmaras working variable for turbulent flow over a flat plate at $x/c = .5$ (plate mid-chord) illustrating non-smooth behavior at the boundary layer edge for $p = 1$ and $p = 4$. Note that solution is more oscillatory when employing a discretization order of $p = 4$

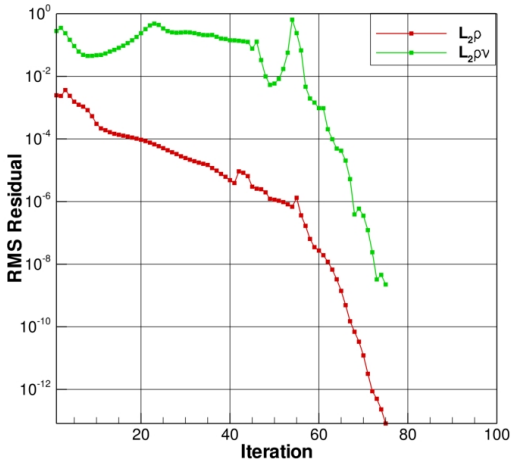


Figure 6.4: Convergence history of the density and turbulence model variable for turbulent flow over a flat plate using the Spalart-Allmaras turbulence model for a $p = 2$ DG discretization using MGPC-GMRES solver.

6.2.2 Turbulent NACA0012 Airfoil

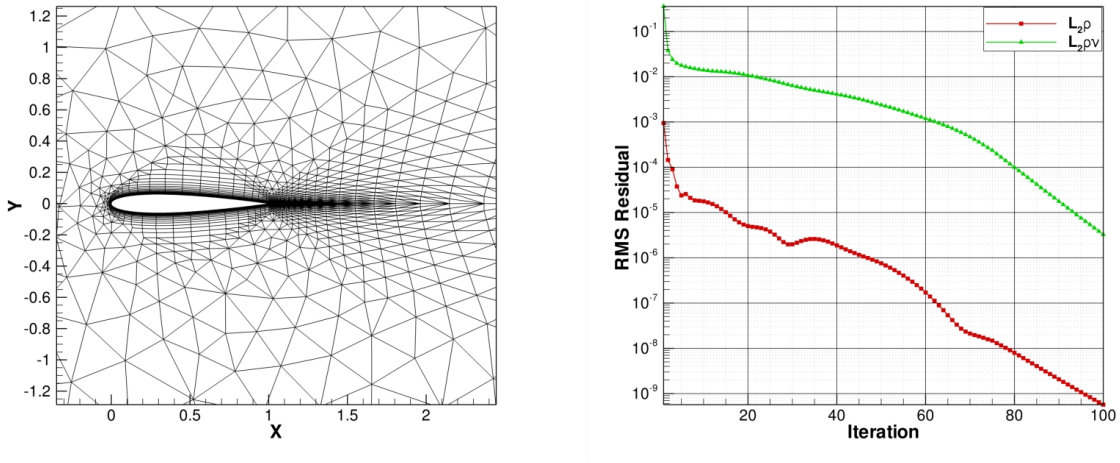
The second test case consists of the turbulent flow over a NACA0012 airfoil using the Spalart-Allmaras turbulence model. The flow conditions are $M_\infty = .25$, $\alpha = 0^o$, and $Re = 1,685,000$ and discretization orders $p = 1$ to $p = 3$ are employed on a mesh with $N = 3,579$ elements

(1,302 quadrilaterals and 2,277 triangles). The maximum aspect ratio of any element in the mesh is 4650:1. The mesh for this test case is shown in Figure 6.5(a). The solution is converged using the MGPC-GMRES solver and each higher p solution is initialized with a fully converged solution of order $p-1$. This flow is significantly more challenging to solve than the flow over the flat-plate as the magnitude of the negative values of the turbulence model working variable become substantially larger. As with the flat plate, this case demonstrates the oscillations in the turbulence model working variable at the edges of boundary layers. Additionally, this flow demonstrates that the edge of the wake contains the same oscillations, which are more severe than at the edge of the boundary layer and generate higher magnitude negative working variable values. This test case also demonstrates a need for additional smoothness and/or limiting of the turbulence model solution to increase the robustness of the DG discretization.

The convergence history for the $p = 3$ solution is depicted in Figure 6.5(b), which clearly shows that the convergence rate is slower than that of both laminar cases in Chapters 4 and 5 and the flat-plate case previously presented. The slow convergence rate in the initial part of the convergence history is due to transients induced by the model source terms. However, the slow convergence rate during the later portion of the convergence history is due to Newton damping requirements imposed by the presence of negative turbulence model variable values. In fact for the $p = 3$ DG discretization of this test case, the maximum CFL number was set to $CFL_{max} = 1000$, which is at least 4 orders of magnitude lower than any other presented test case in this work. Figures 6.6(a) and 6.6(b) show the computed Mach number and turbulent viscosity contours for a $p = 3$ discretization. Figure 6.8 illustrates the surface pressure distribution for orders $p = 1$ to $p = 3$. Figure 6.8 clearly shows an increase in solution quality and smoothness as the discretization order is increased. Figure 6.7 depicts the $\tilde{\nu}$ contours in the negative value regime, which demonstrates that the negative $\tilde{\nu}$ increase in magnitude in the wake region of the domain.

As a preliminary attempt at increasing robustness of the DG RANS discretization the local-order reduction technique of Section 4.6 is applied to the $p = 2$ solution of this problem. In order to apply local-order reduction, the indicator given in equation (2.7.2) was used with

the turbulence model working variable as the quantity of interest. While the quantity used to trigger local-order reduction is based on the turbulence model working variable $\tilde{\nu}$, the discretization order p is reduced for all equations in an element that is flagged for local-order reduction. Figure 6.9(a) shows the cells in which the discretization order p is reduced, which are all at the edge of the boundary and wake, (i.e. turbulent and non-turbulent interfaces) confirming that this region exhibits non-smooth behavior. Although the application of local-order reduction increased the robustness of the solver, local-order reduction also degraded the quality of the solution. Figure 6.9(b) shows the surface pressure distributions for this test case employing a discretization order of $p = 2$ with and without local-order reduction. The $p = 2$ solution with local-order reduction results in a surface pressure profile that is significantly less smooth than the $p = 2$ without local-order reduction. In fact, comparison of Figure 6.8 and Figure 6.9(b) shows that the local-order reduction surface pressure result more closely resembles the $p = 1$ surface pressure result than the $p = 2$ result without local-order reduction. By using $p = 1$ elements at the edge of the boundary layer, the pressure (which is approximately constant through the boundary layer) at airfoil surface has been compromised, indicating that the accuracy of quantities such as lift and drag can be degraded using this approach. Applying local-order reduction to all the equations has shown that the resolution and smoothness requirements of the mean flow and SA turbulence model equations are at odds with one another. Clearly, a robustness enhancement technique that isolates the turbulence model equation from the mean flow equations is required in order to yield the requisite smoothness of the model variable, while simultaneously avoiding adverse effects on the mean flow equations, since the mean flow equations are not the cause of solver failure. Furthermore, this case demonstrates that the mean flow resolution requirements are at odds with the SA turbulence model resolution requirements. Regions where the mean flow equations can benefit from high p discretization order cannot be discretized with high p discretization order due to turbulence model equation robustness problems.



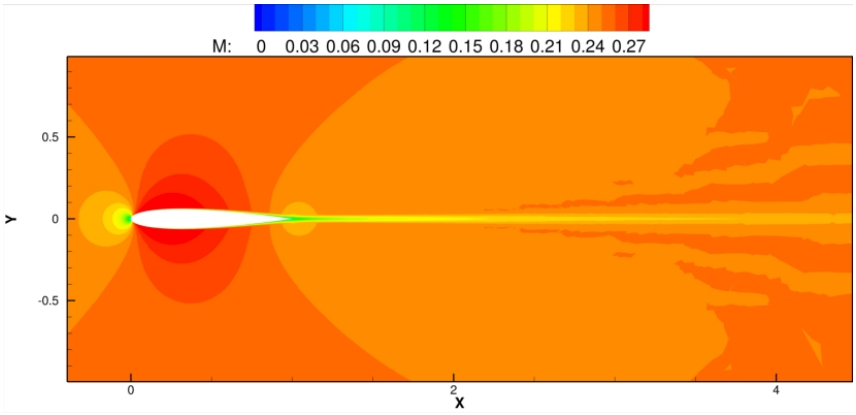
(a) Mesh: $N = 3,579$

(b) Convergence history: $p = 3$

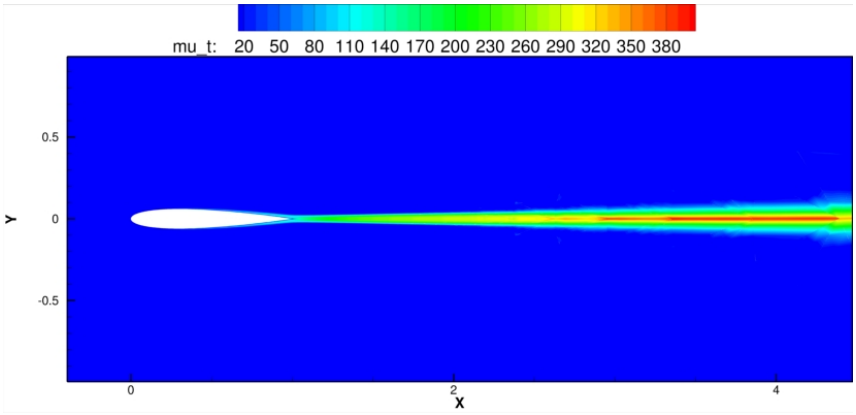
Figure 6.5: Mesh and convergence history for turbulent flow over a NACA0012 airfoil with a DG discretization of both RANS and the Spalart-Allmaras turbulence model equations.

6.3 *hp*-Adaptation with High-order DG Discretization of the Spalart Allmaras Turbulence Model

The results of applying local-order reduction have shown that there is a discrepancy in the type of resolution required by the turbulence model compared to the type of resolution required by the mean flow. Thus *hp*-adaptation is performed to enhance the robustness of solving turbulent flows using high-order DG by accounting for this resolution discrepancy. The idea is simple; anywhere the turbulence model working variable is non-smooth h-refinement is applied, while p-enrichment is applied otherwise. Since a uniform second-order accurate discretization is required globally, this results in some turbulence model working variable negative values. However, provided the initial solution is attainable then *hp*-adaptation should be able to increase turbulent flow resolution and accuracy, enabling high-order methods to attain grid converged functionals and exhibit more robust behavior.



(a) Mach



(b) $\frac{\mu_T}{\mu_\infty}$

Figure 6.6: Computed Mach number and turbulent viscosity contours for turbulent flow over a NACA0012 airfoil with a DG discretization of both RANS and the Spalart-Allmaras turbulence model equations.

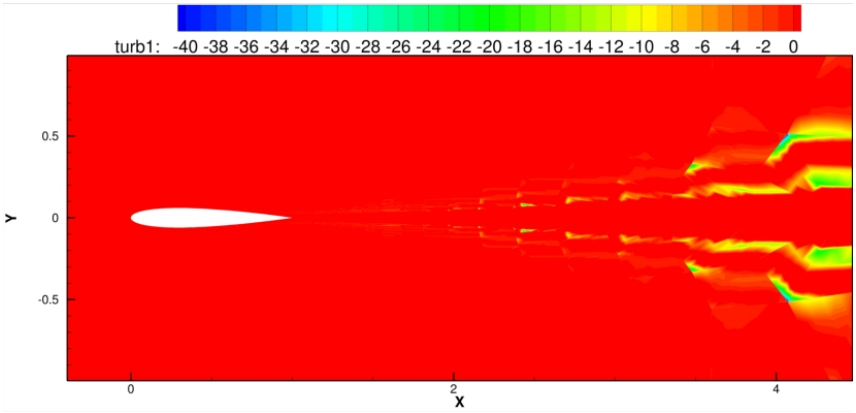


Figure 6.7: Computed $\frac{\tilde{\nu}}{\nu_\infty}$ contours, levels are bounded from 0 to -40 to show the negative turbulence model working variable values.

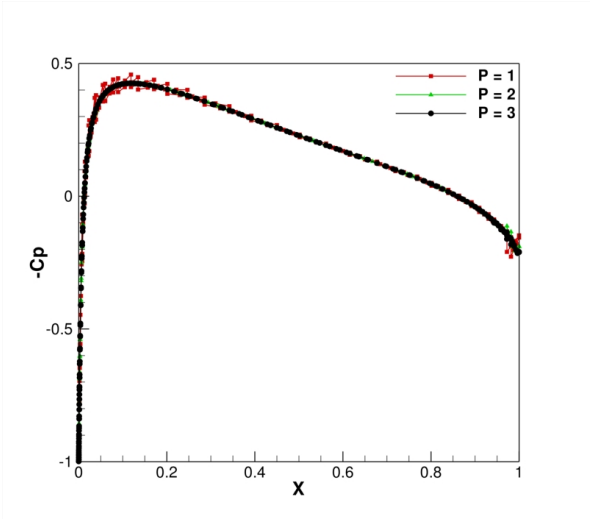
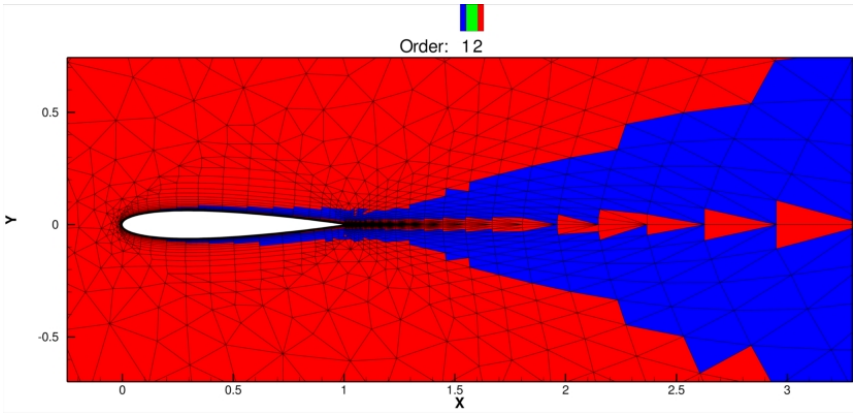


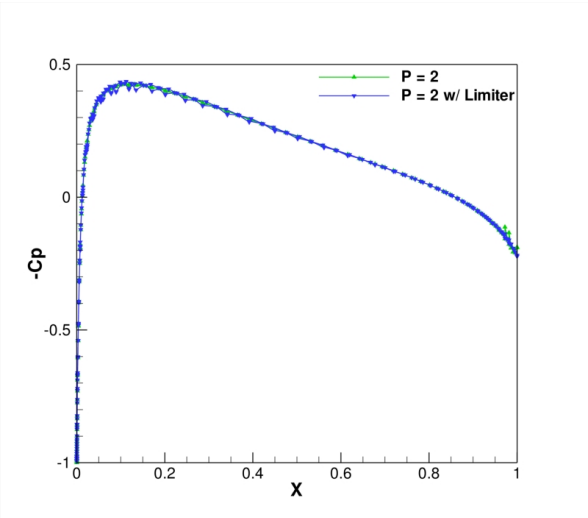
Figure 6.8: Computed surface pressure coefficient of a NACA0012 using RANS coupled to the Spalart-Allmaras turbulence model with orders $p = 1$ to $p = 3$.

6.3.1 Turbulence Model Grid Resolution Requirements

In order to improve the computational efficiency of the solver, the *hp*-adaptation algorithm detailed in Section 5.3 will be applied using both high-order DG and first-order finite-volume discretizations of the SA turbulence model equation. When applying *hp*-adaptation to high-order DG discretizations, the *hp*-adaptation algorithm chooses between *h*-refinement or *p*-enrichment by examining the smoothness of the solution within a cell. Applying this method to turbulent flows requires examining both the smoothness of the pressure and the turbulence model working variable $\tilde{\nu}$. However, experiments applying the smoothness detector of equation (5.3.2) to the turbulence model working variable for turbulent flows have shown that the turbulence model is non-smooth in a very large portion of the domain including at the wall and at the edges of boundary layers and wakes. Figures 6.10(a) and 6.10(b), which are from the computation of a turbulent flat-plate flow at the same flow conditions used in Section 6.4.1, show the smoothness indicator values of equation (5.3.2) for the mean flow and turbulence model respectively, where the cells that have color are deemed as non-smooth. Notice, that while the mean flow is sufficiently resolved in the region adjacent to the wall, the smoothness detector has determined that the turbulence model working variable is non-smooth in this region. While the non-smooth behavior at the edges of boundary



(a) Discretization order

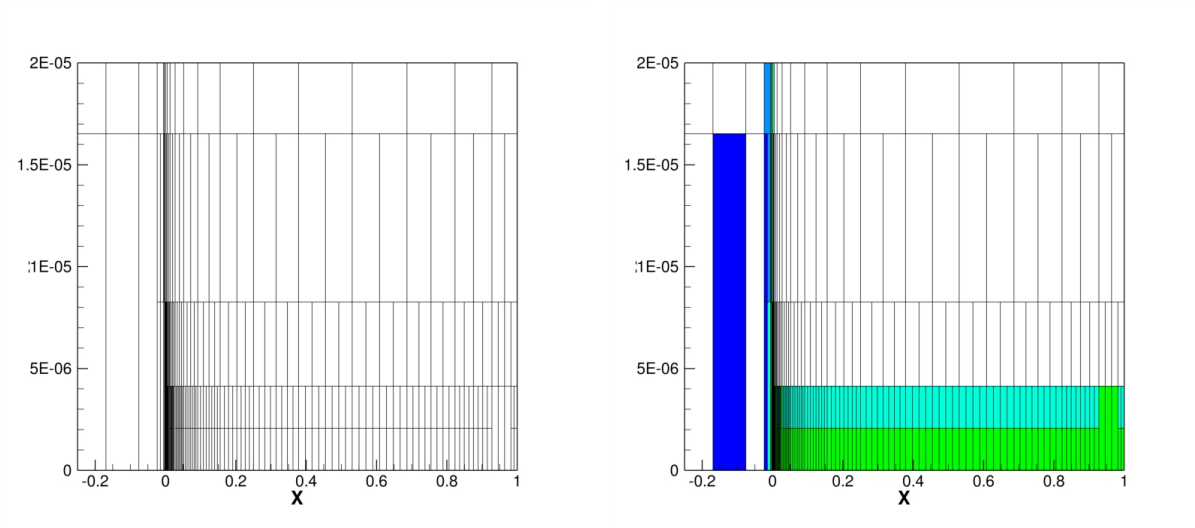


(b) Computed surface pressure coefficient

Figure 6.9: Results of applying local-order reduction to the DG discretization of RANS equations for turbulent flow over a NACA0012 airfoil using the Spalart-Allmaras turbulence model.

layers and wakes is be expected, based on the results in this work as well as the results of references [15, 20, 46, 70, 74], the detection of non-smooth behavior at the wall is unexpected. The results shown in Figure 6.10(a) and Figure 6.10(b) indicate that the SA model requires higher grid resolution at the wall than the mean flow equations.

Unfortunately the region adjacent to the wall is an area where the mean flow equations would benefit form high-order polynomial representation, in order to resolve the smooth high gradient mean flow field in this region. In order to isolate the region adjacent to the wall from the smoothness detection required for the non-smooth behavior at the edge of the boundary



(a) Mean flow smoothness indicator

(b) Turbulence model smoothness indicator

Figure 6.10: Close-up of near wall cells smoothness indicator for the turbulent flow over a flat-plate at $M_\infty = .1$, $Re = 10,000,000$. White cells are detected as smooth and colored cells as non-smooth. DG discretization is employed for the turbulence model with Roe approximate Riemann solver for the convective numerical flux

layer, various indicator quantities for turbulence model smoothness were examined. This work was unable to find an indicator quantity that was able to simultaneously avoid the region adjacent to the wall and also detect the edge of the boundary layer. Furthermore, a search of the literature found that only two references [46, 70] have considered turbulence model smoothness detection and neither of the detection strategies in these references was able to overcome this problem.

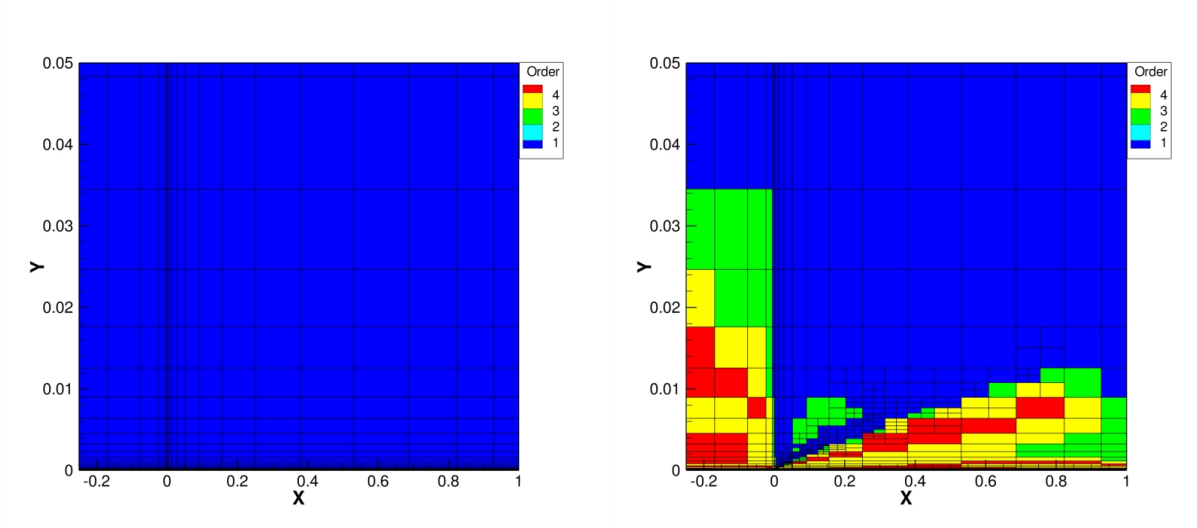
This work also makes use of a hybrid discretization approach where the mean flow equations are discretized using a high-order DG discretization and the turbulence model is discretized using a first-order finite-volume discretization. Since this discretization removes the non-smooth behavior of the turbulence model, turbulence model smoothness becomes a non-issue for this hybrid discretization approach. However, use of a first-order finite-volume discretization of the turbulence model means that increasing the discretization order p does not increase the resolution of the turbulence model. In order to effectively utilize the hp-adaptation strategy, modifications to the decision process that chooses between p-enrichment and h-refinement are required. Rather than basing the decision between p-enrichment and

h-refinement based on turbulence model solution smoothness alone, the contribution from the turbulence model equation to the functional error estimate for a given element (equation (5.2.11)) is quantified, and used as a metric to determine if an element should be refined via h-refinement or p-enrichment. If the contribution of turbulence model equation to the functional error estimate is large enough relative to the total functional error estimate for a given cell, h-refinement is chosen for that cell, which allows for increased resolution of the turbulence model. The smoothness of the pressure is still used to detect non-smooth cells for the mean flow equations. In practice, the smoothness of the mean flow equations is determined for each element and then the contribution from the turbulence model to the output functional error estimate is determined and used to override the mean flow smoothness detector in elements where this contribution is high enough. For this work, if the turbulence model contributes more than 50% to the functional error estimate in a given cell, then the cell is refined via h-refinement, regardless of the smoothness of the mean flow field.

6.3.2 Flat-plate: hp-adaptation

The first hp-adaptation test case consists of the turbulent flow over a semi-infinite flat-plate at $M_\infty = .1$, $\alpha = 0.0^o$, and $Re = 10,000,000$. For this test case, the turbulence model is discretized using a DG discretization with the same discretization order as the mean flow equations. Adjoint based hp-adaptation is performed with drag as the objective. Due to the discontinuity of the turbulence model working variable, the functional convergence is not guaranteed to be regular. Figure 6.11(a) shows a close up of the initial mesh, which contains $N = 540$ elements and employs a discretization order of $p = 1$.

Three cycles of hp-adaptation using computed drag coefficient as the objective are performed, at which point the computed drag coefficient is deemed grid converged. The final hp-adapted mesh is shown in Figure 6.11(b) which has $N = 1,116$ elements employing variable discretization orders of $p = 1$ to $p = 4$. Figure 6.12(a) shows the computed drag coefficient values over the adaptive history. Clearly the drag becomes grid converged over the adaptive history since on the last refinement step the drag value changes by less than .5%. Figure 6.12(b) shows the adjoint error estimate of drag over the adaptation history.

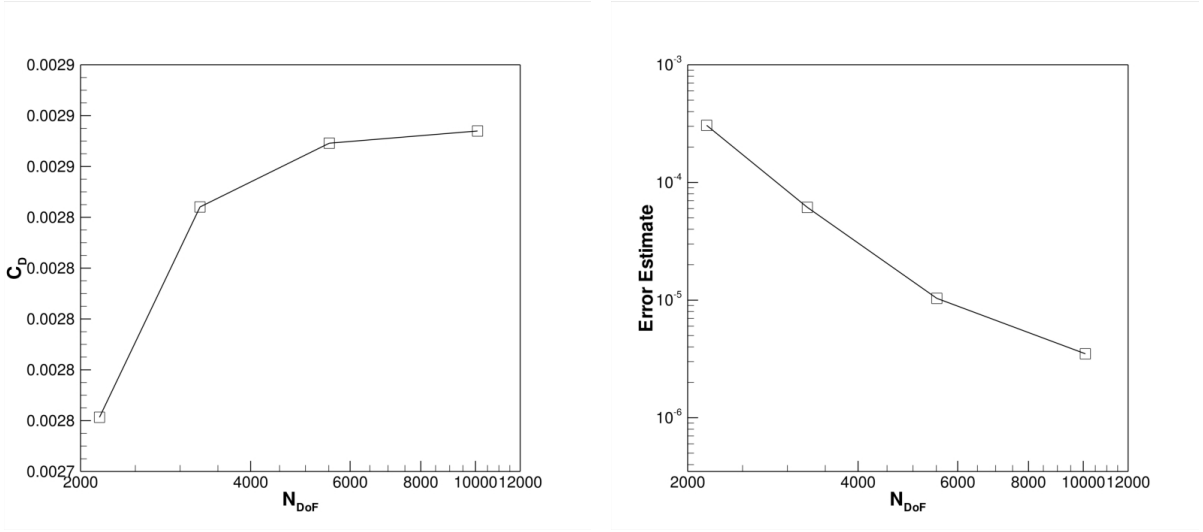


(a) Initial mesh $N = 540$ elements, $p = 1$

(b) Final hp-adapted mesh $N = 1,116$ elements, $p = 1$ to $p = 4$

Figure 6.11: Initial and final meshes for drag driven hp-adaptation of the turbulent flow over a flat-plate. Note the expanded y-axis scale for clarity.

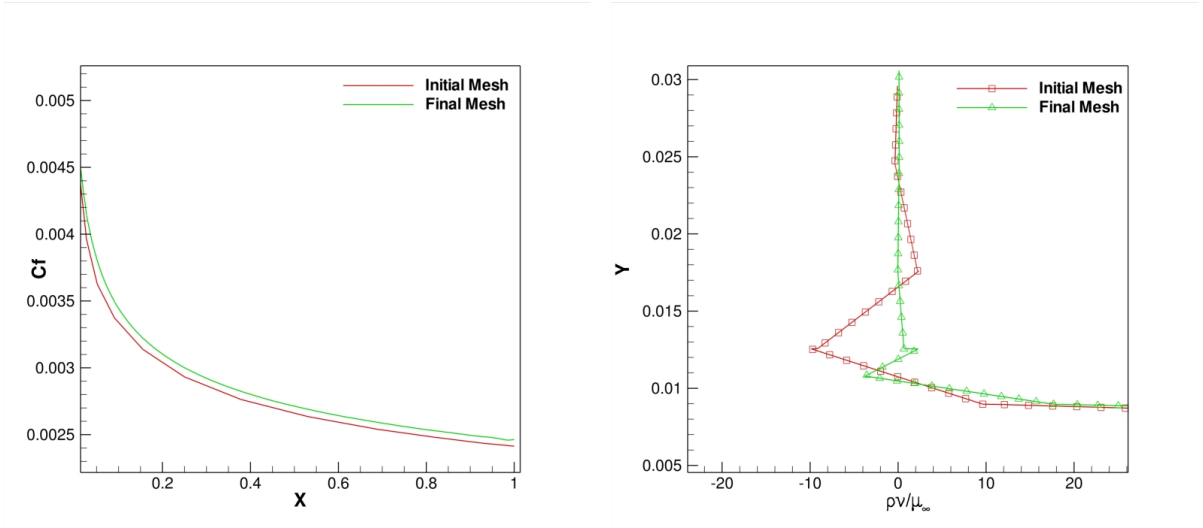
In this case, the drag error estimate continues to drop over the entire adaptation history and reaches a value .03 counts (3.0e-6) at the final adaptation step, indicating that grid convergence of drag comes at very low error levels at least for the viscous drag component, which is the only drag component present in this flow. Furthermore, the average slope of the drag error estimate versus $h(h = \sqrt{N_{DoF}})$ over the adaptation history is 8. Figure 6.13(a) shows the computed skin friction distribution on the plate for the initial and final meshes, illustrating a significant increase in skin friction at all points along the plate. Since the drag has become grid converged, it is interesting to examine the distribution of $\rho\tilde{\nu}$ at the boundary layer edge as shown in Figure 6.13(b). Figure 6.13(b) shows that, despite the significant increase in resolution, the profile of $\rho\tilde{\nu}$ at the edge of the boundary layer has not become smooth. While the final hp-adapted mesh shows a drop in negative value magnitude, the overall behavior is still oscillatory and the model equation is still producing negative values of $\rho\tilde{\nu}$. The fact that the boundary layer edge has not become smooth leads one to question whether this artifact will vanish with increasing mesh resolution. This subject will be the consideration of future work as the goal of the present work to determine if turbulent flows can be solved on reasonable meshes i.e. meshes that are not designed around turbulence



(a) Drag vs. N_{DoF}

(b) Adjoint estimate of drag error

Figure 6.12: Computed drag and drag error estimate vs. N_{DoF} for the *hp*-adaptation of turbulent flow over a flat-plate.



(a) Skin fiction coefficient

(b) Boundary layer edge close-up

Figure 6.13: Computed Skin friction coefficient on initial and final mesh and close up of boundary layer edge working variable at $x = .5$.

model discontinuities.

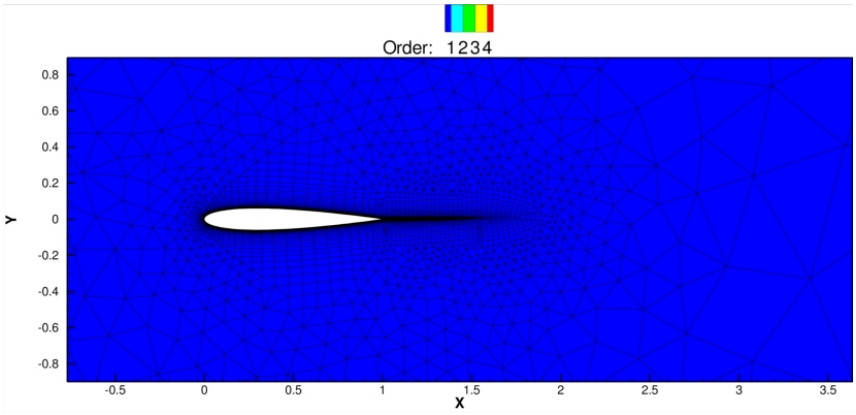
6.3.3 NACA0012 Airfoil: *hp*-adaptation

The second *hp*-adaptation test case consists of the turbulent flow over a NACA0012 airfoil. This test case investigates the robustness enhancement properties of *hp*-adaptation for a more challenging test case with larger magnitude negative turbulence model working variable values. Furthermore, this test case also investigates the the grid convergence of lift, which is a more challenging functional due to the strong dependence of lift on inviscid flow phenomena.

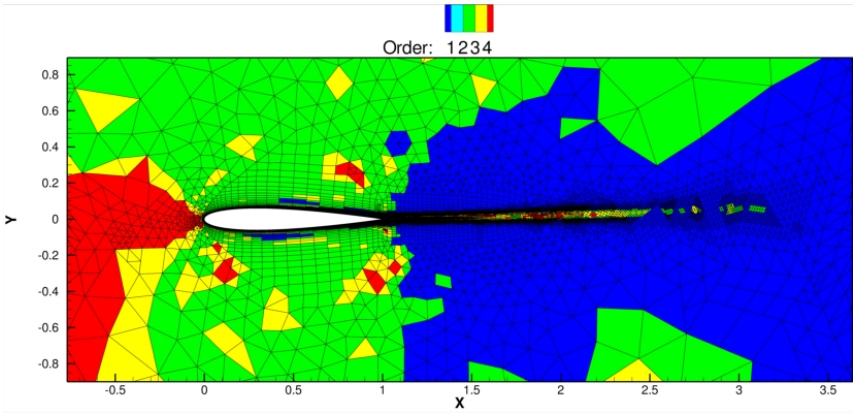
The flow conditions for this test case are $M_\infty = .25$, $\alpha = 2.0^o$, and $Re = 1,685,000$. Six cycles of *hp*-adaptation are performed with lift as the objective and $p = 4$ is set as the maximum allowable discretization order. The turbulence model equation is discretized to the same order as the mean flow equations using a high-order DG discretization. The approximate Riemann solver of Roe is used for the convective numerical flux of both the mean flow and turbulence model equations. Additionally, the flow is solved using a fully coupled flow Jacobian with the GMRES method preconditioned by the CGS algorithm. The initial mesh shown in Figure 6.14(a) contains $N = 5,082$ elements at a discretization order $p = 1$, yielding $18,706$ DoFs.

Figure 6.14(b) shows the final *hp*-adapted mesh after 6 cycles of *hp*-adaptation with the computed lift coefficient as the objective. Note the highly refined wake and boundary layer regions. The edge of the wake where the turbulence model is non-smooth is targeted exclusively with *h*-refinement while the core of the wake is targeted with *p*-enrichment. Furthermore, *p*-enrichment is applied upstream and around the airfoil outside of the boundary layer where the turbulence model is smooth and has little influence on the solution. The only reason that *h*-refinement is applied is due to turbulence model non-smoothness or because an element has been *p*-enriched to the maximum allowable discretization order, which for this case is set to a discretization order of $p = 4$.

Figure 6.15(a) through Figure 6.16(b) show the computed Mach number and eddy viscosity contours on the intial and final meshes. Notice that the wake is significantly more resolved on the final mesh, as is the inviscid flow region above and below the airfoil. Figure



(a) Initial mesh: $N = 5,082$ elements, $p = 1$

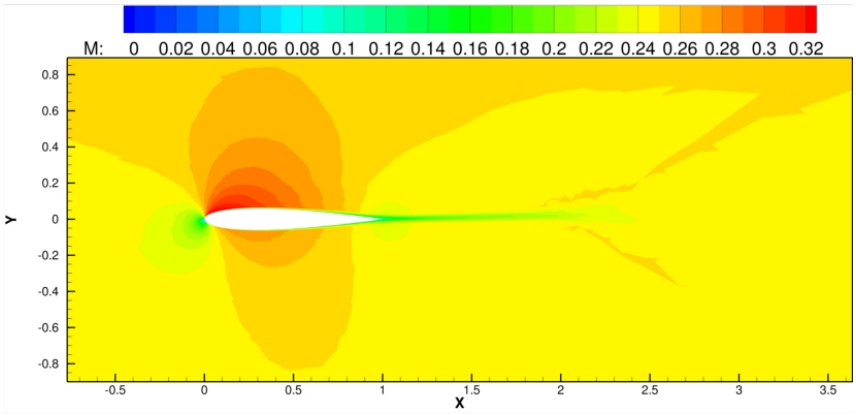


(b) Final *hp*-adapted mesh: $N = 92,358$, $p = 1$ to $p = 4$

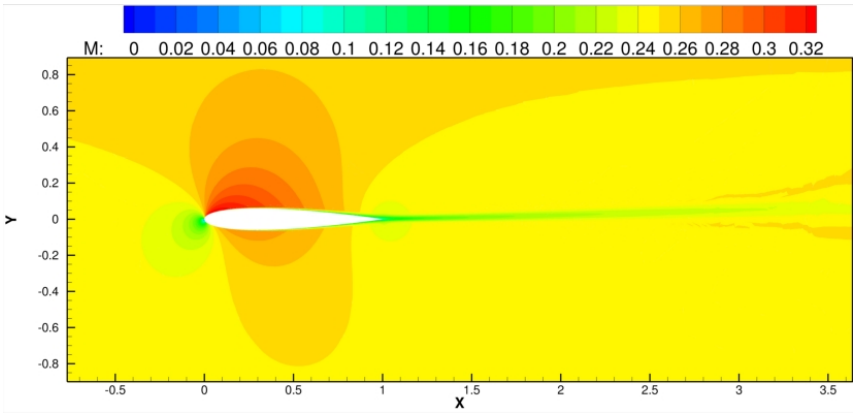
Figure 6.14: Initial and final meshes for lift-driven adjoint-based *hp*-adaptation of turbulent flow over a NACA0012 airfoil.

6.17(a) and Figure 6.17(b) depict the pressure and skin friction distributions on the final *hp*-adapted mesh respectively. Smooth pressure and skin friction distributions are obtained at the final adaptive step.

Figure 6.18(a) shows the computed lift coefficient versus N_{DoF} over the adaptation history. Examination of this figure shows that the computed lift coefficient does not become grid converged despite using over $600,000$ DoFs for this relatively simple flow. Furthermore, Figure 6.18(a) shows that the coarse level corrected lift coefficient does not match the fine level computed lift coefficient it is designed to predict at any point during the *hp*-adaptation history. The poor correction of the coarse level lift coefficient is not surprising based on the adjoint error estimate behavior shown in Figure 6.18(b). Figure 6.18(b) shows that the



(a) Initial mesh: Mach number contours

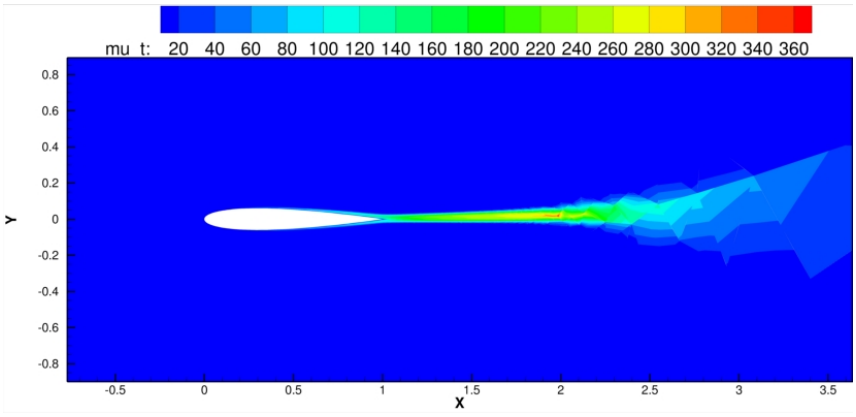


(b) Final mesh: Mach number contours

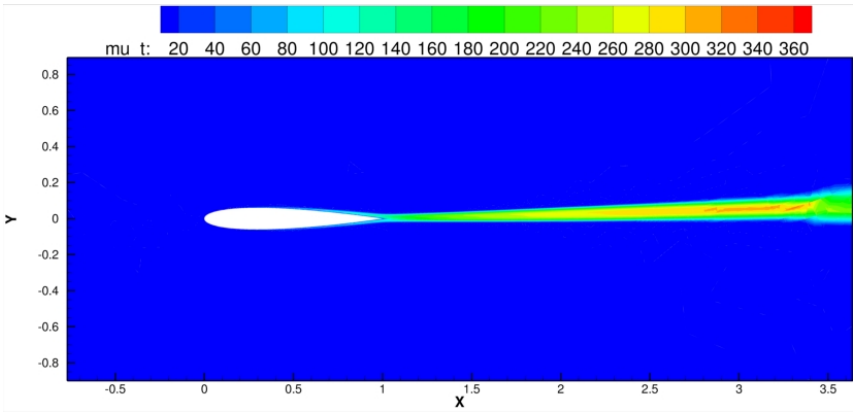
Figure 6.15: Computed Mach number contours on the initial and final meshes of the *hp*-adaptation of turbulent flow over a NACA0012 airfoil.

adjoint error estimate is not reduced by performing *hp*-adaptation. The error estimate is not converging to zero because of an effect similar to the one discussed in Section 5.4.3. However, for this test case, it is not the mean flow equations that are corrupting the fine level residual estimate but rather the turbulence model equation due to the discontinuity of the turbulence model working variable and the dual inconsistency discussed in Chapter 3. The discontinuity in the turbulence model working variable causes Gibbs phenomena, which as discussed in Section 5.4.3 impacts the fine level residual estimate $\mathbf{R}_h\left(\mathbf{u}_H^h\right)$ used to obtain the computable error ϵ_c in equation (5.2.10).

These results lead to several questions: why is the error estimate so poor? What is affecting the functional convergence? Why was this not the case with the flat-plate? The



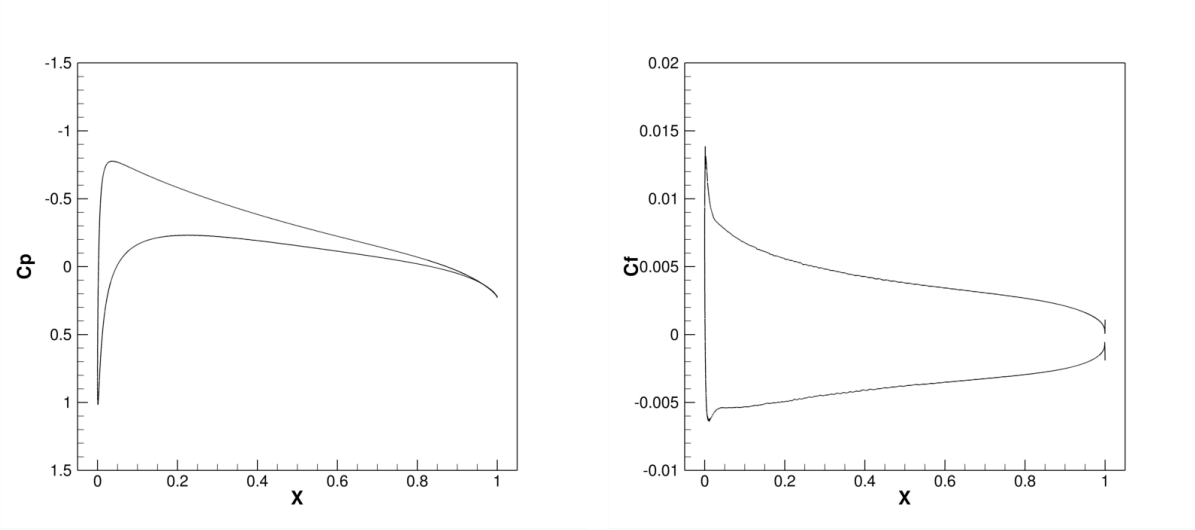
(a) Initial mesh: $\frac{\mu_T}{\mu_\infty}$ contours



(b) Final mesh: $\frac{\mu_T}{\mu_\infty}$ contours

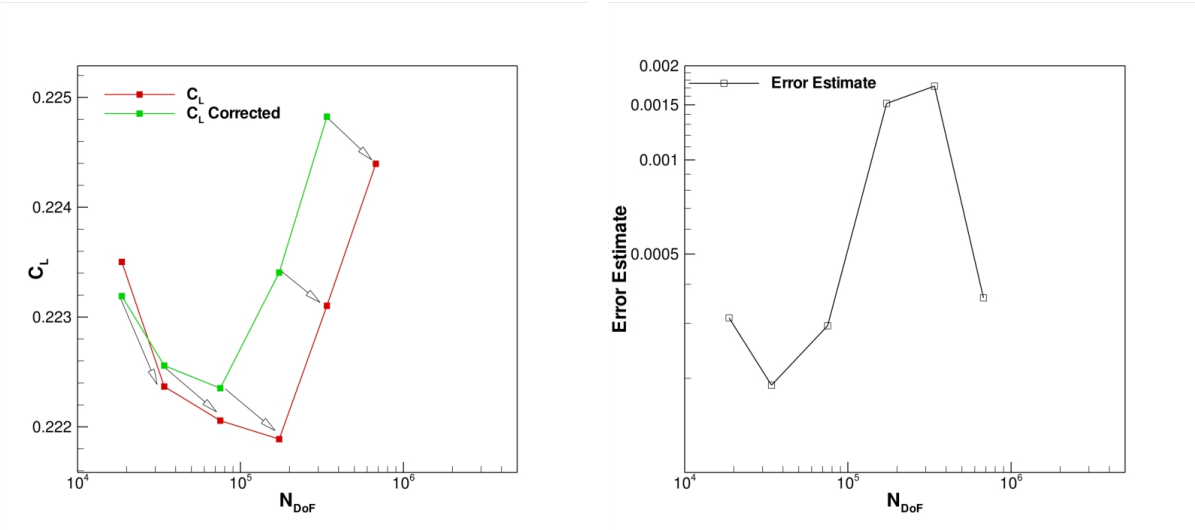
Figure 6.16: Computed eddy viscosity contours on the initial and final meshes resulting from the *hp*-adaptation of turbulent flow over a NACA0012 airfoil.

answer to all of these questions is related to the turbulence model working variable discontinuity. The turbulence model working variable discontinuity adversely impacts functional converge, due to significant changes in the turbulence model working variable, which do not converge to fixed values as adaptation is performed. The turbulence model working variable discontinuity causes inaccurate error estimation in the same fashion as a $p = 0$ discretization at the shock wave does in the *hp*-adaptation case presented in Section 5.4.3. Essentially, the fine level residual estimate is corrupted due to the Gibbs phenomena that result from the turbulence model working variable discontinuity. In the case of the transonic flow in Section 5.4.3, artificial viscosity is able to regularize fine level solution estimate $\mathbf{u}_H^h$ and hence correct the Gibbs phenomena.



(a) C_p (b) C_f

Figure 6.17: Computed coefficient of friction and surface pressure coefficient on the final mesh for the *hp*-adaptation of the turbulent flow over a NACA0012 airfoil.



(a) Lift vs. N_{DoF} (b) Lift adjoint error estimate vs. N_{DoF}

Figure 6.18: Computed lift coefficient and lift coefficient error estimate over the *hp*-adaptation history for turbulent flow over a NACA0012 airfoil.

Artificial viscosity was also added in both piecewise constant and PDE-based forms to the turbulence model equation in order remove the Gibbs phenomena. However, application of the artificial viscosity resulted in failure either through causing solver failure (lack of robustness) or by altering the eddy viscosity profile so that the eddy viscosity was no

longer sufficient to model the turbulent mixing. Furthermore, the dual inconsistency of the turbulence model discretization can contribute to inaccurate error estimation. Attempts to remedy the dual inconsistency were made, and as mentioned in Chapter 3, these dual inconsistency remedies were also ineffective. The dual consistency modifications employed for the turbulence model increased the computational time by a factor of two and also adversely impacted the robustness of the solver (see Chapter 3 for details).

Finally corrupt error estimates were not obtained in the flat-plate case because the functional in that case did not depend on the turbulence model solution in the discontinuous regions. Recall that the flat-plate was adapted based on drag. In that case, viscous drag is the only component of the drag that is non-zero. Viscous drag accuracy is influenced by the region of the boundary layer very close to the wall where the SA model equation is relatively well behaved. In contrast, the airfoil lift has a strong dependence on flow curvature, which generates pressure based forces. These pressure based forces are primarily responsible for generating lift in this fully attached flow. Flow curvature is a largely inviscid phenomena, but there is a strong influence from the edge of the boundary layer on the computed lift coefficient. The obvious nature of this statement is seen in the final *hp*-adapted mesh shown in Figure 6.14(b) where the edges of the boundary and wake regions as well as the outer region of the flow have been very heavily refined.

In conclusion, the presence of the SA turbulence model working variable discontinuity in the actual solution has caused poor functional convergence while the presence of the discontinuity in the fine level solution estimate causes inaccurate error estimation. If left unchecked, this discontinuity will continue to plague turbulent flow solutions, adversely impacting robustness, functional convergence, and functional error estimation. Since artificial viscosity has proven to be successful as a shock capturing method, attempts were made to utilize artificial viscosity for the turbulence model working variable discontinuity, but artificial viscosity was unable to remove the Gibbs phenomena without adversely impacting turbulence model solution accuracy. In fact, Section 6.4.1 will show that employing a limiter for the second-order discretization of the turbulence model equation convection term does not perform in a satisfactory manner. Section 6.4.1 will also show that only a first-order dis-

cretization employing the upwind convective numerical flux alleviates the Gibbs phenomena effectively at least for the solution of the turbulence model equation.

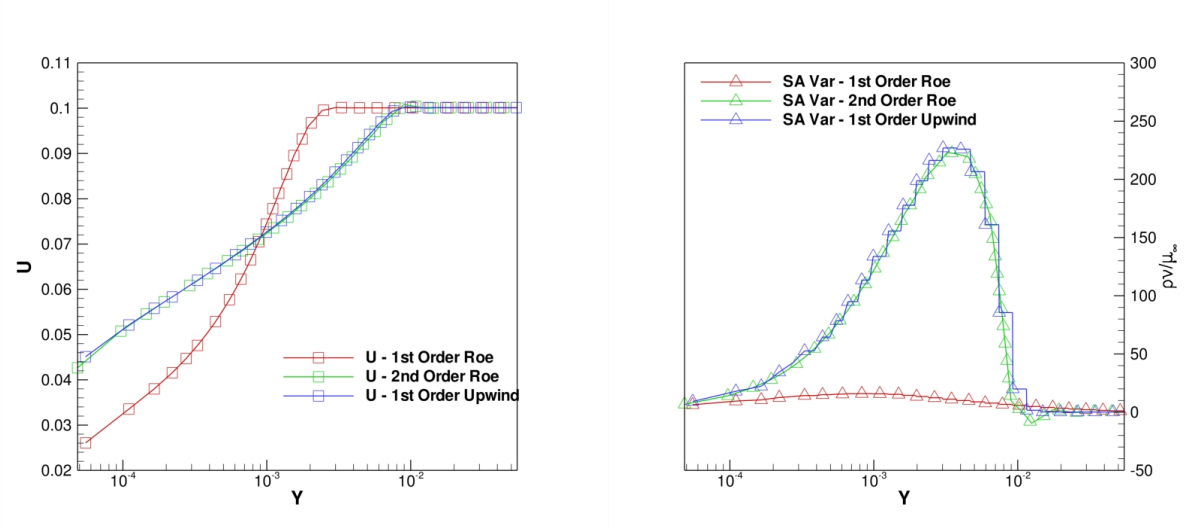
6.4 Effects of Numerical Methods on the Spalart Allmaras Turbulence Model Solution

In this section, the effects of numerical methods on the turbulence model solution are examined. In particular, the choice of convective numerical flux as well as the coupling of the mean flow and turbulence model equations are discussed. For simplicity the discussion of these effects is restricted to a single numerical example, which is the turbulent flow over a flat-plate at $M_\infty = .1$ and $Re = 10,000,000$. The discussion of the effects of numerical methods on the turbulence model solution is restricted to this simple test case in order to avoid unsteady flow solutions in the presence of low eddy viscosity values.

6.4.1 Convective Flux Discretization

Through numerical experimentation, it has been found that the SA turbulence model is sensitive to the amount of artificial diffusion introduced by the convective flux discretization. This sensitivity is demonstrated by either the generation of negative values of the turbulence model working variable or low eddy viscosity production, which causes low eddy viscosity values that are insufficient to model turbulent flow physics. On the one hand, high artificial diffusion values cause the eddy viscosity values to become inappropriately low. Artificially low values of eddy viscosity result in an inadequate model of the turbulent physics. On the other hand, low artificial diffusion values result in the generation of negative turbulence model working variable $\tilde{\nu}$ values. The negative values of $\tilde{\nu}$ the result of the aforementioned non-smooth behavior or discontinuity in the turbulence model working variable. The presence of negative values of $\tilde{\nu}$ often result in solver failure.

Consider the turbulent flow over a semi-infinite flat-plate at the aforementioned flow conditions. Two basic discretizations are compared for this flow. The first discretization is a $p = 1$ DG discretization that employs the approximate Riemann solver of Roe [52] for

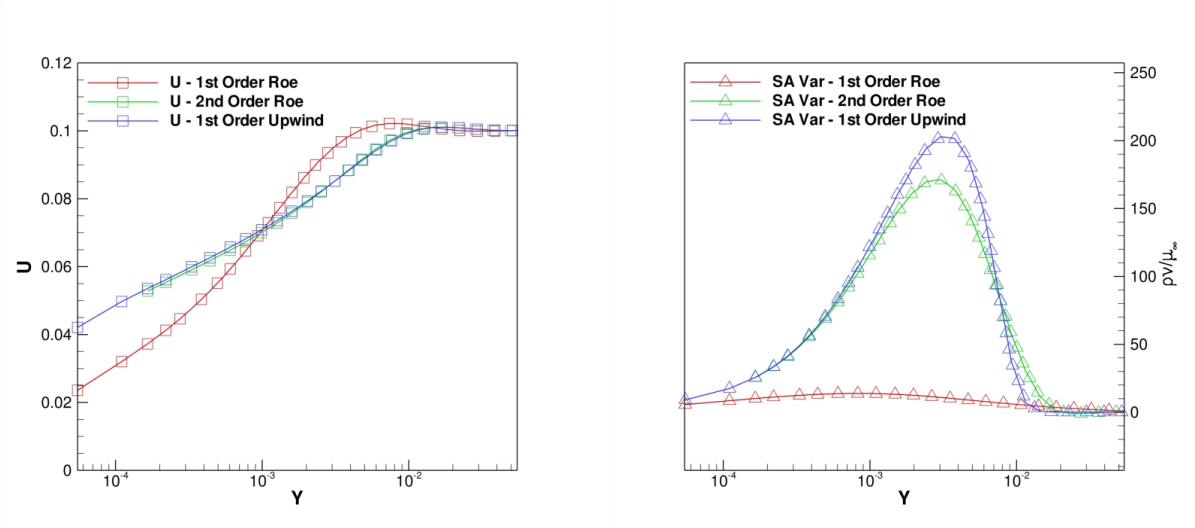


(a) u-velocity versus y/c

(b) $\rho\tilde{\nu}/\mu_\infty$ versus y/c

Figure 6.19: Mid-chord profiles of u-velocity and working variable versus y/c for flow over flat-plate with $M_\infty = .1$, $Re = 10,000,000$, and $p = 1$ using a DG solver for the mean flow and various discretizations and convective numerical flux formulations for the turbulence model .

the convection terms of both the turbulence model equation and the mean flow equations. The second discretization is a $p = 1$ DG discretization for the mean flow combined with a finite-volume discretization employing first-order accurate convection terms, for the turbulence model, which is described in Section 2.6. Two convective numerical fluxes for the first-order discretization of the turbulence model are examined: an upwind flux derived for the turbulence model alone (Section 2.6), where the artificial diffusion has no acoustic component, and a Roe approximate Riemann solver which fully couples the artificial diffusion for the mean flow and turbulence model equations and therefore contains an acoustic component in the turbulence model convective numerical flux. Essentially, the upwind flux treats the model equation as through it is decoupled from the mean flow equations. However, the Roe approximate Riemann solver treats the turbulence model equation by adding it as an additional governing PDE to the system of equations in equation (2.1.1) and re-derives the approximate Riemann solver for this new system of equations. Figures 6.19(a) and 6.19(b) depict the u-velocity and $\rho\tilde{\nu}$ profiles plotted versus y/c using all the turbulence model discretization options. These profiles are extracted at the mid-chord of the plate $x/c = .5$ where c is the chord length of the plate.



(a) u-velocity (b) $\rho\tilde{\nu}/\mu_\infty$

Figure 6.20: Mid-chord profiles of u-velocity and working variable versus y/c for flow over flat-plate with $M_\infty = .1$, $Re = 10,000,000$ using a second-order finite-volume solver for the mean flow and various discretizations and convective numerical flux formulations for the turbulence model.

One immediately notices that using a $p = 1$ DG discretization of the turbulence model equation results in oscillations at the boundary layer edge, which causes negative SA working variable values. Contrarily, employing a first-order finite-volume turbulence model convection discretization using a Roe approximate Riemann solver for the convective numerical flux, results in under production of the eddy viscosity. The only presented turbulence model discretization option that provides sufficient eddy viscosity levels and remains positive throughout the domain is the first-order finite-volume discretization of the turbulence model convection term employing an upwind flux(equation (2.6.3)). One should ask why the upwind flux is not used in combination with a $p = 1$ DG discretization of the turbulence model, and the answer is that the solver is unable to converge the turbulence model equation with this combination. The artificial diffusion supplied by this combination is insufficient to generate a stable discretization i.e. the negative SA working variable $\tilde{\nu}$ values have large magnitudes and cause solver failure. Furthermore, as illustrated in Figure 6.19(a) a highly inaccurate mean flow u-velocity profile results from the use of a first-order turbulence model convection term discretization, employing a Roe approximate Riemann solver.

Figures 6.20(a) and 6.20(b) depict the same flow conditions computed using a second-

order finite-volume solver. The finite-volume solver is capable of discretizing the mean flow equations using a second-order accurate finite-volume method based on a weighted least-square gradient reconstruction [81]. This discretization has also been implemented for the turbulence model equation in this finite-volume solver. Additionally, the finite-volume solver is also able to discretize the turbulence model equation using the first-order convection term discretization described in Section 2.6. In this case, the second-order finite-volume discretization of the turbulence model employs a min-mod limiter, the application of which was found to be necessary in order to obtain a converged solution. Although it is more difficult to see in Figure 6.20(b), the second-order finite-volume turbulence model convection discretization produces an oscillation despite the use of the min-mod limiter in these computations and the oscillation results in negative values of the SA working variable locally. Furthermore, the presence of the limiter has affected the peak value of the eddy viscosity, which is 16% lower than the first-order finite-volume discretization employing the upwind flux. Thus the second-order turbulence model convection discretization is just as unsatisfactory in the finite-volume context as in the DG context, even with the use of the min-mod limiter. The second-order finite-volume turbulence model discretization results show the same trend as the $p = 1$ DG turbulence model discretization results. Hence, these issues appear in both DG and finite-volume discretization methods on unstructured grids. The results of the finite-volume solver illustrate that the non-smooth behavior is not restricted to DG discretizations.

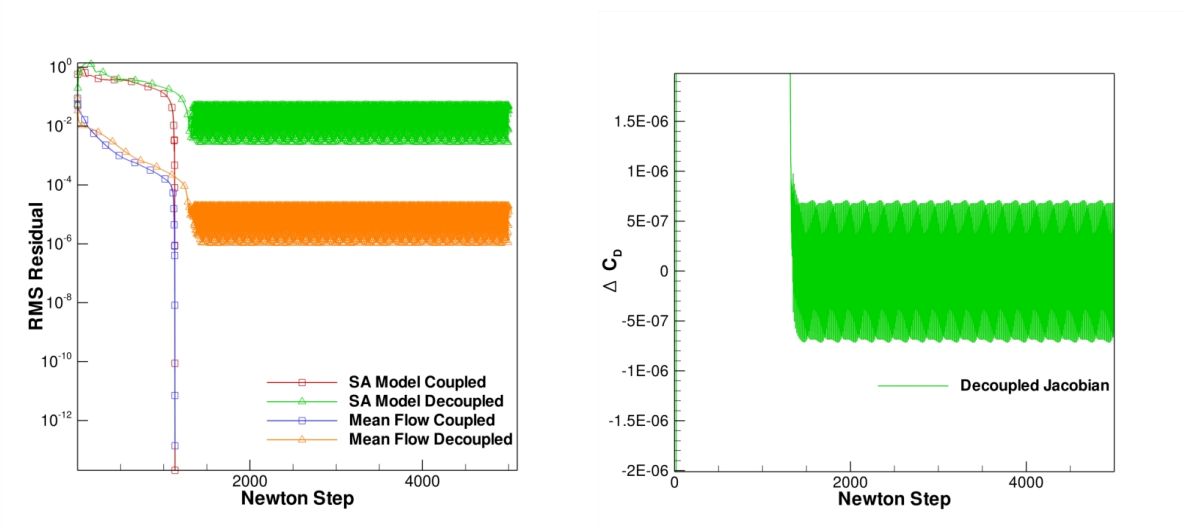
Two conclusions can be drawn from this simple test. One is that the SA turbulence model is sensitive to the amount of artificial diffusion that is applied through the convective flux discretization. Both the discretization order and the form of the discrete convective flux have strong influences on the turbulence model solution. In particular, coupling the turbulence model and mean flow convective flux discretization, results in a numerical scheme with too much artificial diffusion and thus the model does not produce sufficient eddy viscosity. However, initial experiments for scalar convection problems with high-order DG discretizations demonstrated that coupling the convective numerical flux between the equations controlling the background velocity field(in this case the Euler Equations) and

the scalar convection equation is necessary to achieve positive and smooth scalar values, for smooth initial conditions of the scalar. These two conclusions are at odds with one another, leaving few options for discretizing the turbulence model equation to high-order accuracy. Furthermore, the optimal combination for convective term discretization for the turbulence model has been found to be a first-order convection discretization employing an uncoupled upwind flux formulation (i.e a numerical flux where the sound speed does not appear in the artificial diffusion term for the turbulence model). Unfortunately, it seems any second or higher-order convection discretization will not guarantee positive values of $\tilde{\nu}$, regardless of the discretization employed, even if one applies a monotone limiter.

6.4.2 Algebraic Coupling

Constructing implicit solution techniques requires the determination of the flow Jacobian matrix as shown in equation (4.2.3). The flow Jacobian requires the differentiation of the residual $\mathbf{R}_h$ with respect to the discrete solution $\hat{\mathbf{u}}_h$, a process known as linearization. Turbulence models are often linearized in a decoupled fashion i.e. turbulence models are linearized only with respect to the variables that the model equation controls. This results in a loosely coupled implicit solver formulation that in many cases cannot fully converge the discrete equations. However, if the linearization is performed in a fully coupled fashion, it is far more likely that full convergence of the discrete equations can be achieved, at least with the presented DG and finite-volume solvers. Full convergence of the discrete equations is especially important for high-order discretizations, since the magnitude of high-order modal coefficients can be small.

A fully coupled linearization is obtained by differentiating the model equation with respect to every modal coefficient in the system of equations including the mean flow and turbulence model equations. For example, the turbulence model equation requires the molecular viscosity μ for the source and diffusion terms, which depends on all the mean flow quantities ρ, ρu, ρv, and E_t. Therefore, when linearizing the turbulence model there are off-diagonal coupling entries in the block Jacobian matrix corresponding to differentiation with respect to the modal coefficients of the mean flow quantities.



(a) Discrete flow equation residual history

(b) Computed drag difference

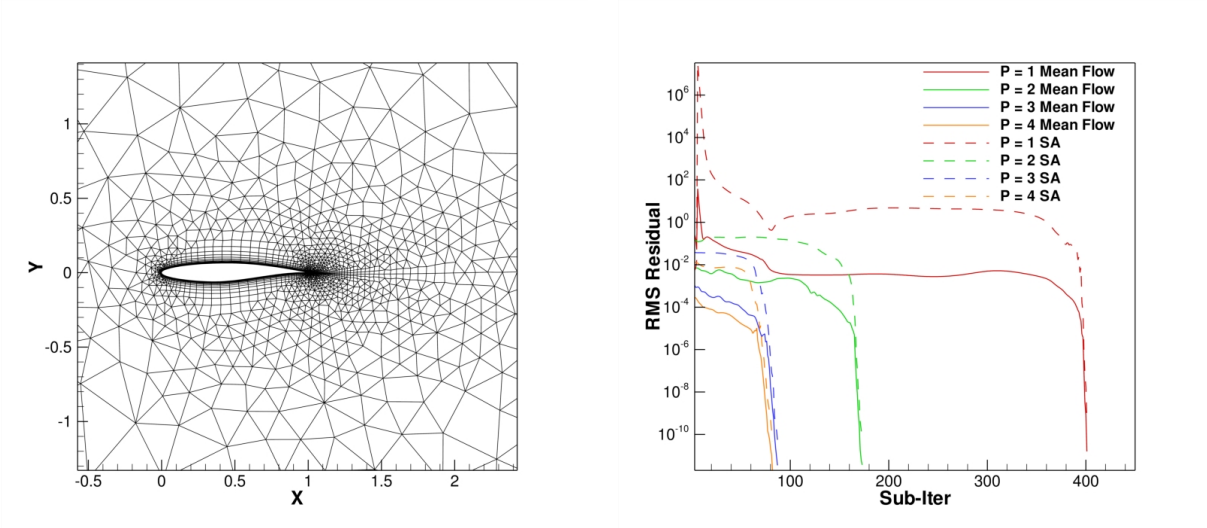
Figure 6.21: Comparison of coupled versus decoupled Jacobians on iterative convergence for turbulent flow over a flat-plate with $M_\infty = .1$ and $Re = 10,000,000$. The solution is obtained using the CGS preconditioned GMRES solver. The drag difference (right) is the difference between the fully converged flow solution computed drag value and the partially converged computed drag value, resulting from employing a decoupled Jacobian in the implicit solver.

Figure 6.21(a) depicts the iterative convergence using the CGS preconditioned GMRES solver for the turbulent flow over a flat-plate, employing a $p = 1$ DG discretization for the mean flow equations and the first-order turbulence model discretization, using both coupled and decoupled linearizations. Figure 6.21(a) shows that employing the decoupled flow Jacobian results in an implicit solver that cannot fully converge the flow equation residuals to machine zero. However, employing the fully coupled flow Jacobian results in an implicit solver that fully converges both the turbulence model and mean flow equations without issue. Figure 6.21(b) depicts the difference between the drag value obtained by fully converging the discrete equations and the drag computed using the partially converged decoupled flow Jacobian approach, $\Delta C_D = \left|C_{D coupled} - C_{D decoupled}\right|$. Notice that the error in the computed drag value induced by partially converging the discrete equations i.e. $\left|C_{D coupled} - C_{D decoupled}\right|$ is approximately $\pm 5.0e-7$. Although the values of ΔC_D are small, values of this magnitude will often be insufficient for use with high-order methods where discretization errors are small. For example, the first AIAA international high-order methods workshop specifies functional error tolerances of $1.0e-6$.

The issue of coupling of the flow Jacobian matrix is raised because of the solution strategy mentioned in reference [41]. In this solution strategy, sub-iterations with a decoupled flow Jacobian are used to solve the SA turbulence model equation. This is mentioned in reference to positivity preservation during the iterative process. The positivity preservation proceedure is not described in reference [41]. This type of proceedure is really only valid when employing a decoupled flow Jacobian in the implicit solver, which results in implicit solvers that are less than satisfactory based on the presented results and the results of reference [97]. Hence these types of positivity preservation techniques will not be able to aid in DG solutions of the RANS equations regardless of the turbulence model discretization or more generally in a situation where low error tolerances are specified.

6.5 Hybrid Discretization Results

A DG solver that is capable of solving the RANS equations with a finite-volume discretization of the turbulence model is applied to several practical aerodynamic flows including two high-lift multi-element airfoil configurations. The combination of a high-order DG discretization with a finite-volume discretization of the turbulence model equation with a first-order convection discretization is denoted as a hybrid discretization. The discretization of the the turbulence model employed for all test problems is described Section 2.1 and includes modifications that are designed to enhance solver robustness when the turbulence model working variable becomes negative(Section B.1), which can occur during the solution of the discrete flow equations. However, negative values of the turbulence model working variable do not exist in the final steady-state solutions when the turbulence model is discretized with a first-order finite-volume method. These cases are presented to illustrate the robustness of this approach and to demonstrate that using a high-order DG discretization for the mean flow equations alone can still be beneficial for overall solution accuracy. All test cases are solved using a damped-Newton method with a line-implicit CGS preconditioned GMRES solver detailed in Chapter 4.



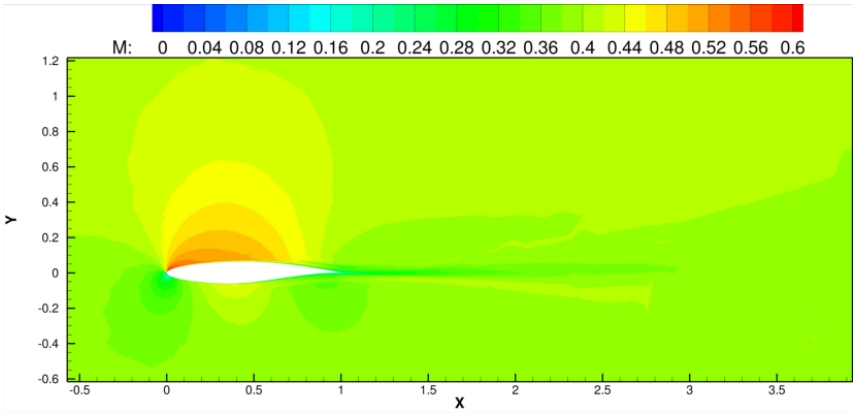
(a) Mesh $N = 5,980$ (b) Convergence history

Figure 6.22: Convergence history and mesh for subsonic flow over a RAE2822 airfoil at $M_\infty = .4$, $\alpha = 2.79^o$, and $Re = 6,500,000$ using discretization orders $p = 1$ to $p = 4$ for the mean flow and a first-order discretization for the turbulence model.

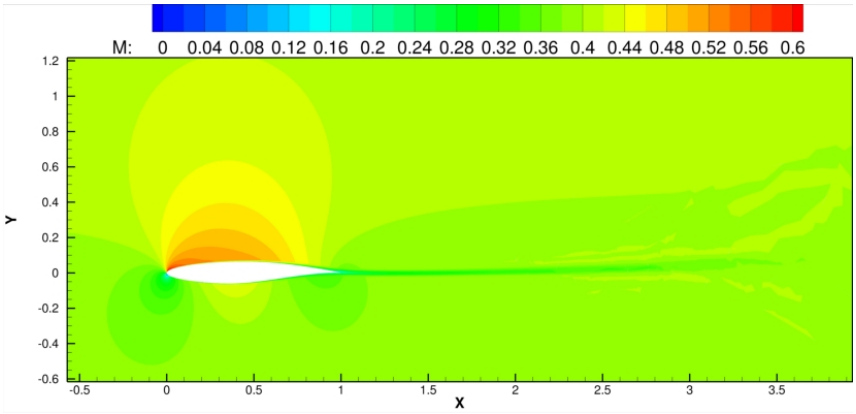
6.5.1 Subsonic RAE2822 Airfoil

The first test case consists of the turbulent flow over an RAE2822 airfoil at $M_\infty = .4$, $\alpha = 2.79^o$, and $Re = 6,500,000$. The computational mesh employed for this test case is a mixed-element unstructured mesh containing $N = 5,980$ elements, with discretization orders ranging from $p = 1$ to $p = 4$. The computational mesh is depicted in 6.22(a). The solution is converged using a line-implicit CGS preconditioned GMRES solver and each solution of discretization order p is initialized with a fully converged solution of discretization order $p - 1$. Due to the first-order upwind finite-volume discretization of the turbulence model equation, increasing the discretization order p does not increase the number of unknowns of the turbulence model discretization i.e. turbulence model resolution remains fixed during p-enrichment.

The convergence history for all discretization orders p is depicted in Figure 6.22(b), which shows that a fully converged solution is obtained at all discretization orders. The convergence rate is slower compared to laminar flow problems such as those in Chapter 4. The slower convergence rate is due to the Newton damping requirements imposed by the

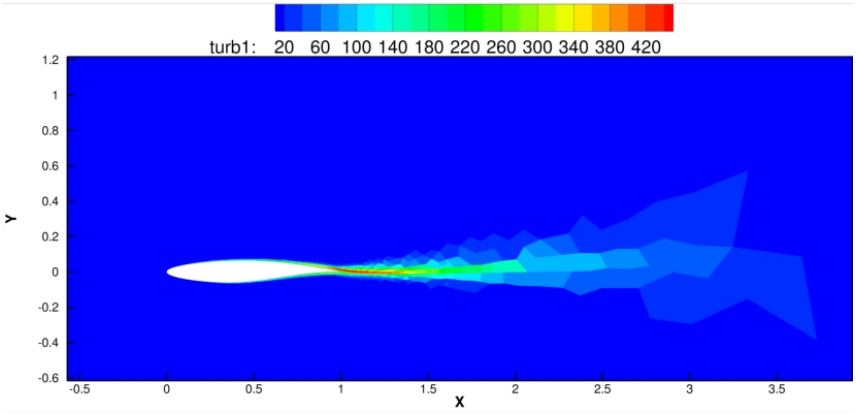


(a) Mach number contours, $p = 1$

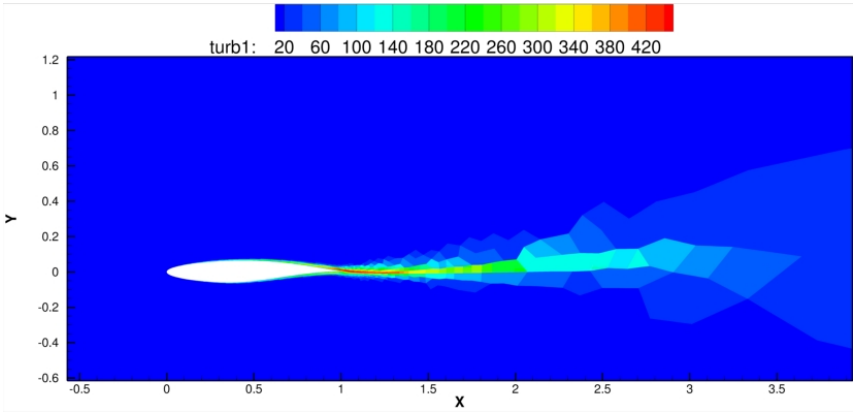


(b) Mach number contours, $p = 4$

Figure 6.23: Computed Mach number contours for the subsonic flow over an RAE2822 airfoil at $M_\infty = .4$, $\alpha = 2.79^o$, and $Re = 6,500,000$.

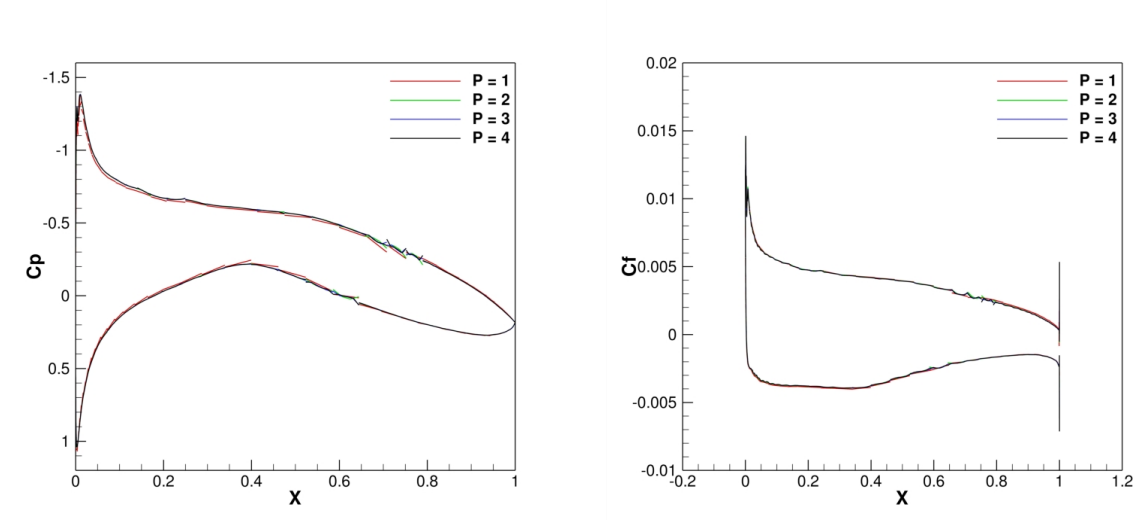


(a) $\rho\tilde{\nu}/\mu_\infty$ contours, $p = 1$



(b) $\rho\tilde{\nu}/\mu_\infty$ contours, $p = 4$

Figure 6.24: Computed $\rho\tilde{\nu}/\mu_\infty$ contours for subsonic flow over an RAE2822 airfoil using the Spalart Allmaras turbulence model at $M_\infty = .4$, $\alpha = 2.79^o$, and $Re = 6,500,000$ with mean flow discretization orders $p = 1$ and $p = 4$.



(a) Surface pressure coefficient

(b) Skin friction coefficient

Figure 6.25: Computed surface pressure coefficient and skin friction for subsonic flow over an RAE2822 airfoil using the Spalart Allmaras turbulence model at $M_\infty = .4$, $\alpha = 2.79^o$, and $Re = 6,500,000$ using $p = 1$ to $p = 4$.

Table 6.1: Computed lift and drag coefficients for the RAE2822 airfoil with $M_\infty = .4$, $\alpha = 2.79^o$, and $Re = 6,500,000$ using $p = 1$ to $p = 4$.

p	NDoF	C_L	C_D
1	21,482	.536226	.016322
2	46,692	.553743	.008991
3	81,548	.554062	.008932
4	126,050	.553607	.008933

turbulence model source terms. However, once the turbulence model source term transient has passed, the solution proceeds rapidly to a fully converged state. Note that Figure 6.22(b) does not show the secondary transient of Figure 6.4. Figures 6.23(a) through 6.24(b) depict the computed Mach number and normalized turbulence model working variable contours for a $p = 1$ and a $p = 4$ discretization respectively. Figure 6.25(a) illustrates the surface pressure distribution for discretization orders $p = 1$ to $p = 4$, and Figure 6.25(b) shows the computed surface skin friction profiles also for discretization orders $p = 1$ to $p = 4$. Note that, as the discretization order is increased, these profiles become smoother indicating that increasing discretization order results in enhanced solution accuracy. Also note that the non-smooth behavior in the computed skin friction profile is located at sharp corners where

the geometry is not differentiable, hence the non-smooth behavior of the skin friction profile is a post-processing artifact.

Table 6.1 shows the discretization order p, N_{DoF}, computed lift and computed drag coefficients across the discretization order range. Table 6.1 shows that the computed drag coefficient has been resolved to .01 counts and the computed lift coefficient has been resolved to approximately 8 counts. As stated in the introduction, studying grid convergence is a primary target of this work, and the results show that the solver is able to obtain grid converged results with respect to polynomial order changes and a fixed turbulence model resolution.

6.5.2 High-lift Multi-element Airfoil Configuration 30P30N

The second test case consists of the turbulent flow over a high-lift multi-element airfoil configuration denoted as the 30P30N configuration. The flow conditions for this test case are $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$. The geometry for this test case consists of: a leading edge slat, a center or main element and a trailing edge flap, which are configured for a landing configuration. The computational mesh employed for this computation is a mixed-element unstructured mesh consisting of $N = 55,964$ elements shown in Figure 6.26.

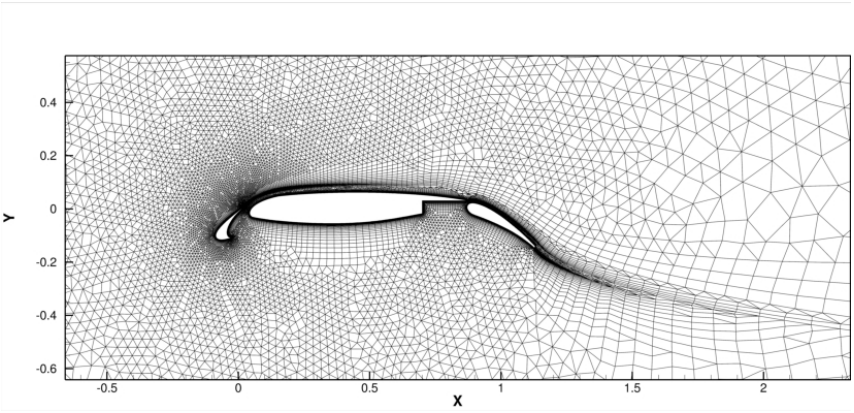
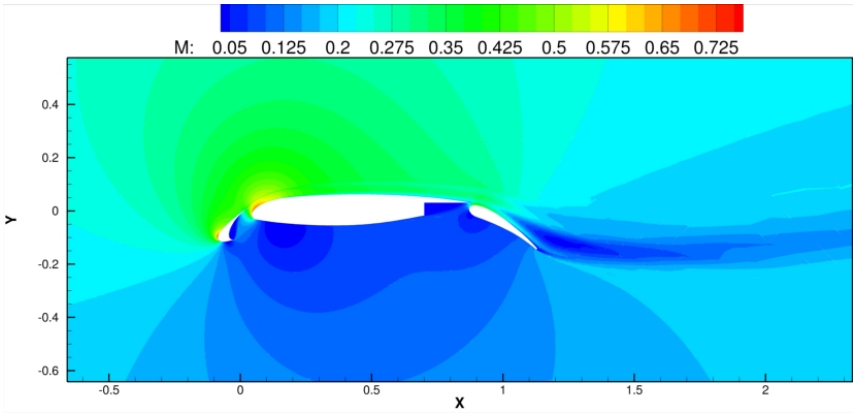
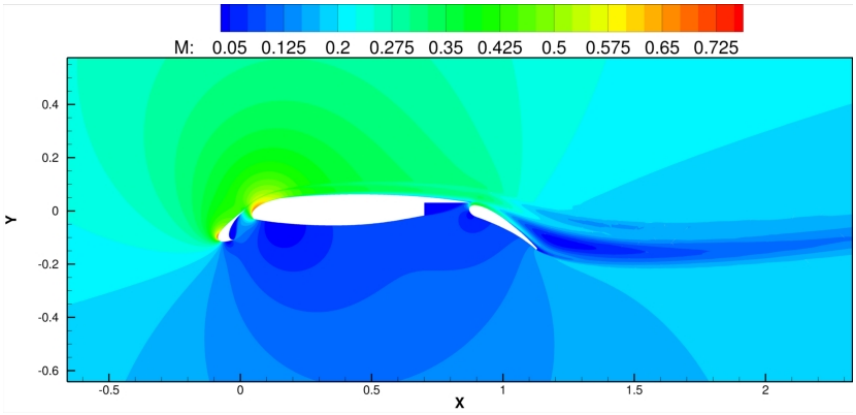


Figure 6.26: Mixed-element unstructured mesh used for computing the flow around the 30P30N high-lift multi-element airfoil configuration at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$.



(a) Mach number contours, $p = 1$



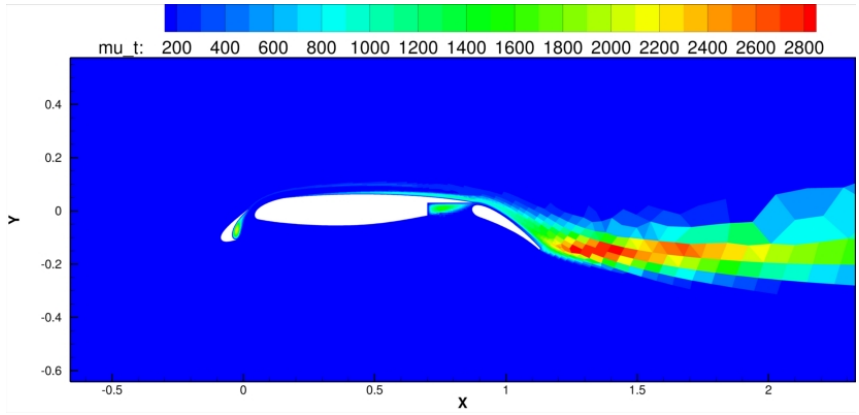
(b) Mach number contours, $p = 3$

Figure 6.27: Computed Mach number contours using the Spalart-Allmaras turbulence model for flow over the 30P30N multi-element airfoil configuration with $p = 1$ and $p = 3$, $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$ using discretization order $p = 1$ to $p = 3$ for the mean flow and a first-order discretization for the turbulence model.

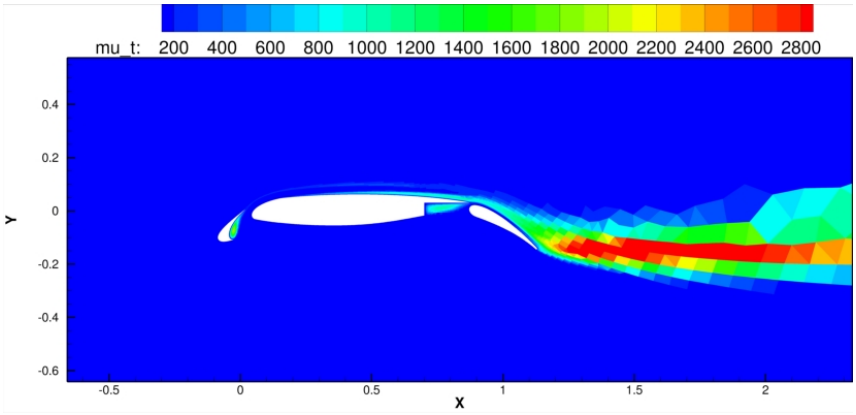
Table 6.2: Computed lift and drag coefficients for the 30P30N multi-element airfoil configuration using mean flow discretization orders $p = 1$ to $p = 3$.

p	NDoF	C_L	C_D
1	195,899	4.12289	.052933
2	419,805	4.10281	.052202
3	727,682	4.10369	.052106

The computed Mach number and normalized turbulence model working variable contours for $p = 1$ and $p = 3$ solutions are depicted in Figures 6.27(a) through 6.28(b) respectively. This flow is also computed using the NSU2D flow solver, which was previously used to compute this case in reference [5]. The NSU2D flow solver is based on a vertex-centered



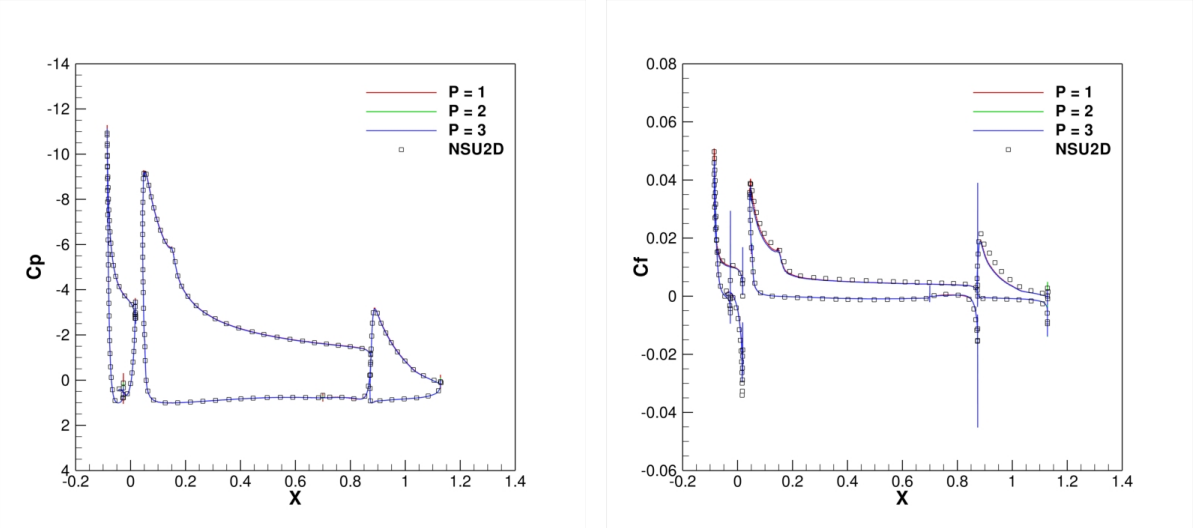
(a) $\rho\tilde{\nu}/\mu_\infty$, $p = 1$



(b) $\rho\tilde{\nu}/\mu_\infty$, $p = 3$

Figure 6.28: Computed $\rho\tilde{\nu}/\mu_\infty$ contours using the Spalart-Allmaras turbulence model for flow over the 30P30N multi-element airfoil configuration airfoil with $p = 1$ and $p = 3$, $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$ using mean flow discretization orders $p = 1$ to $p = 3$ and a first-order discretization for the turbulence model.

second-order accurate finite-volume discretization for the mean flow equations and the same first-order accurate finite-volume discretization of the SA turbulence model equation. The NSU2D solver is used to compute the same flow using a triangular unstructured mesh with 250,000 nodes, which is equivalent to 250,000 DoFs. The computed surface pressure and skin friction coefficients using both the present DG solver and the NSU2D solver are depicted in Figure 6.29(a) and Figure 6.29(b) respectively. Figures 6.29(a) and 6.29(b) show good agreement between the two solvers for both the computed surface pressure coefficient and skin fiction coefficient. There is however, a slight discrepancy in the computed skin friction coefficient on the flap upper surface.



(a) Surface pressure coefficient $p = 1$ to $p = 3$ (b) Skin friction coefficient $p = 1$ to $p = 3$

Figure 6.29: Computed surface pressure and skin friction coefficients using the Spalart-Allmaras turbulence model for flow over the 30P30N multi-element airfoil configuration with mean flow discretization orders $p = 1$ and $p = 3$ at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$.

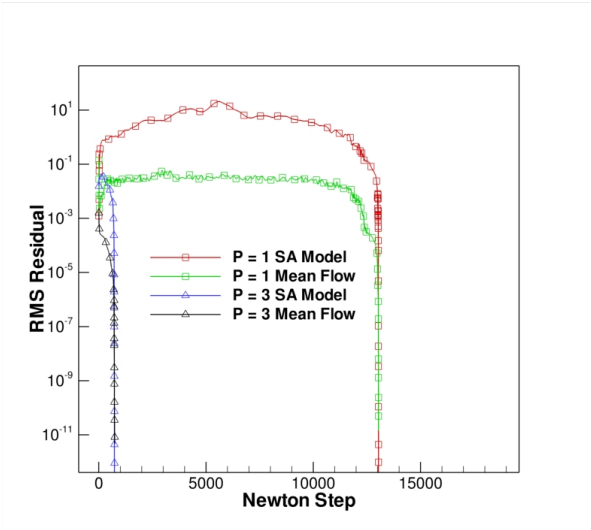


Figure 6.30: Convergence history for flow over the 30P30N multi-element airfoil configuration using the Spalart-Allmaras turbulence model at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$ using mean flow discretization orders $p = 1$ and $p = 3$.

Table 6.2 gives the discretization order p, N_{DoF}, computed lift and computed drag coefficients for this case. One can see a regular decrease of the computed drag coefficient as the discretization order is increased. The computed drag coefficient is resolved to within 1 count. The computed lift coefficient varies non-monotonically as the discretization order p

is increased and is resolved to approximately 8 counts.

This flow involves strong but smooth flow field gradients and is considered relatively challenging. As such, this case is a good test of the robustness of the proposed approach. The DG discretization of the mean flow equations causes no robustness problems for this case. Figure 6.30 depicts the iterative convergence of the flow solver and similarly to the previous RAE2822 airfoil case, once the turbulence model transient has passed the solution converges rapidly to steady-state. Also notice that, for the higher-order DG discretizations of the mean equations, the turbulence model transient is far more benign due to the flow solution initialization from a lower-order solution and the fact that the turbulence model resolution remains unchanged during p-enrichment.

6.5.3 High-lift Multi-element Airfoil Configuration L1T2

The third test case consists of the turbulent flow over the AGARD L1T2 high-lift multi-element airfoil configuration. The geometry consists of a three-element airfoil configuration and the flow conditions are $M = .197$, $\alpha = 20.18^o$, and $Re = 3,520,000$. The mesh employed for this test case is a mixed-element unstructured mesh with $N = 80,742$ elements as shown in Figure 6.31. As with the previous test case, discretization orders ranging from $p = 1$ to $p = 3$ are employed. This case is presented in order to compare the DG results with experimental data provided by AGARD.

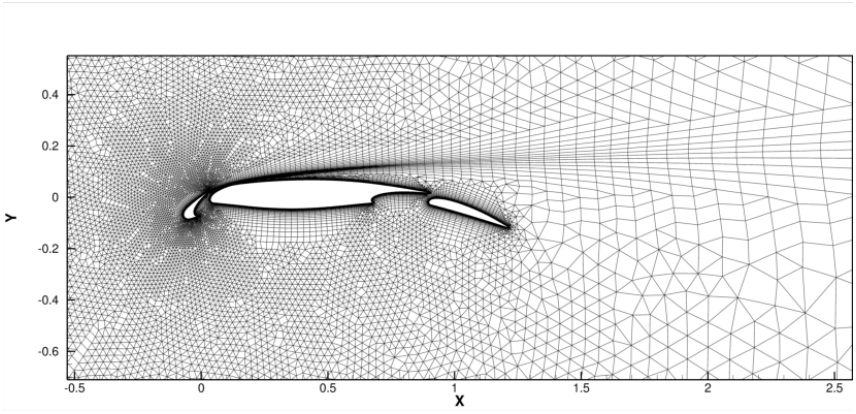
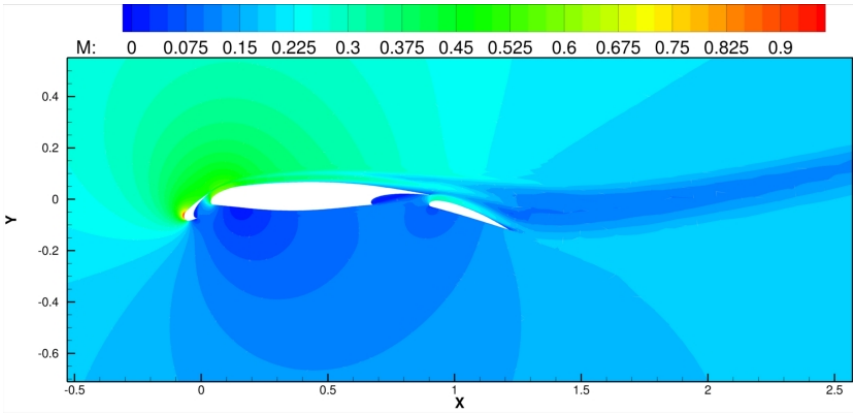
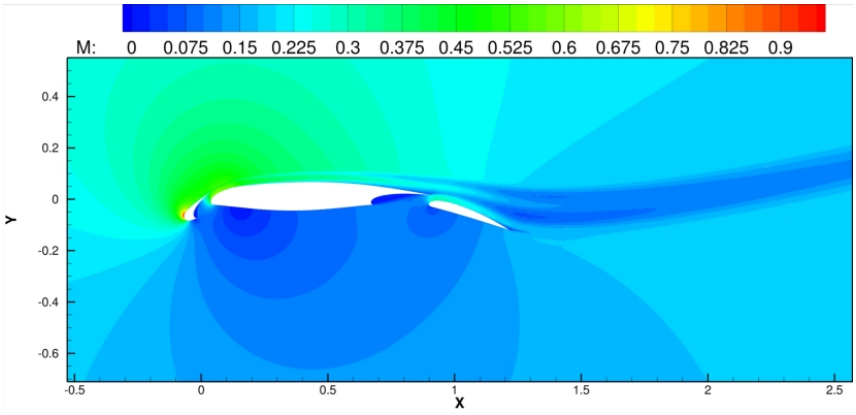


Figure 6.31: Computational mesh used for computing the flow around the AGARD L1T2 high-lift multi-element airfoil configuration at $M_\infty = .197$, $\alpha = 20.18^o$, and $Re = 3,520,000$.



(a) Mach number contours at $p = 1$



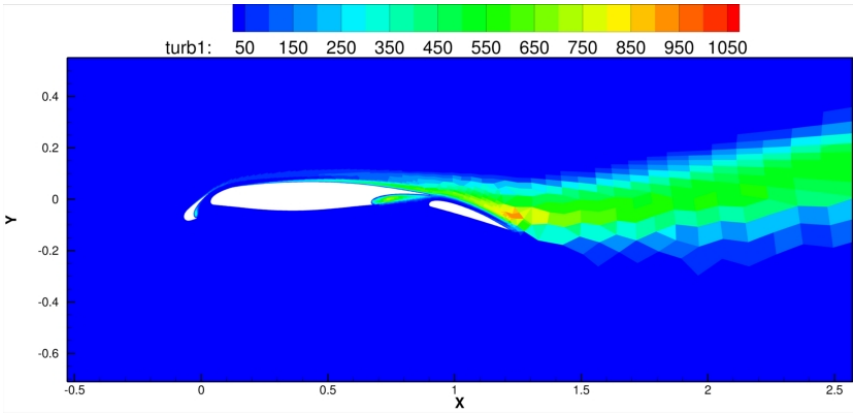
(b) Mach number contours at $p = 3$

Figure 6.32: Computed Mach number contours using a DG discretization with the Spalart-Allmaras turbulence model for flow over the AGARD L1T2 high-lift multi-element airfoil configuration with mean flow discretization orders $p = 1$ and $p = 3$, $M_\infty = .197$ and a first-order discretization for the turbulence model at $\alpha = 20.18^o$, and $Re = 3,520,000$.

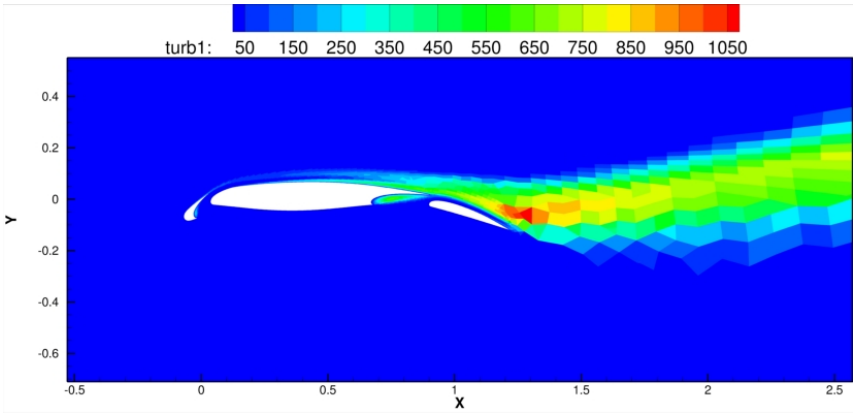
Table 6.3: Computed lift and drag coefficients for the AGARD L1T2 multi-element airfoil configuration at $M_\infty = .197$, $\alpha = 20.18^o$, and $Re = 3,520,000$ using $p = 1$ to $p = 3$

p	NDoF	C_L	C_D
1	265,311	4.036398	.069400
2	553,707	4.012107	.068767
3	945,930	4.010951	.068340

Figures 6.32(a) through 6.33(b) depict the computed Mach number and turbulence model working variable contours for a $p = 1$ and a $p = 3$ solution respectively. From the computed Mach number contours, the flow is seen to approach sonic conditions on the slat leading-edge upper surface. The DG solver is robust enough to compute this flow



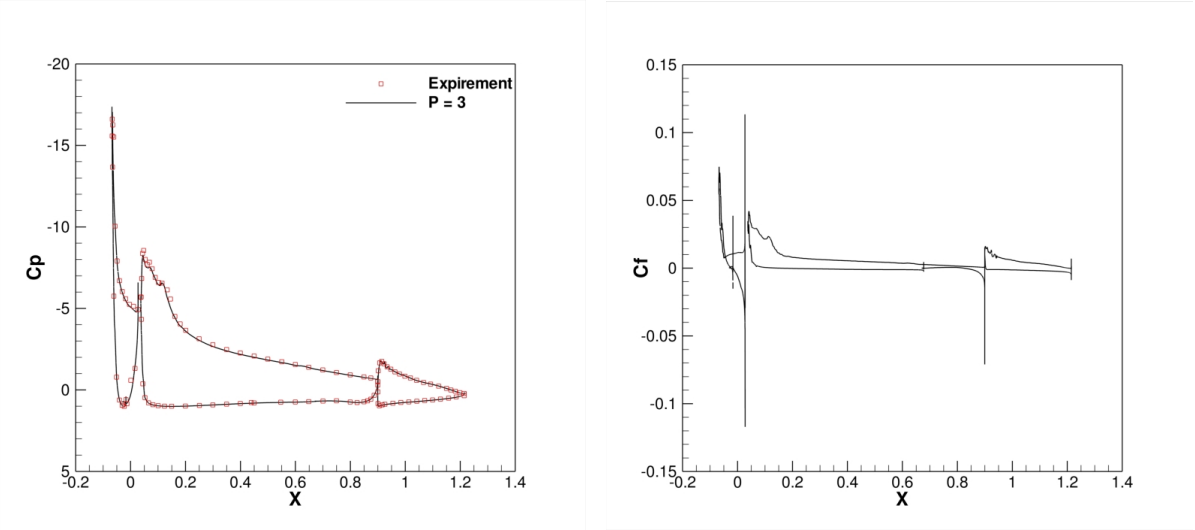
(a) $\rho\tilde{\nu}/\mu_\infty$ at $p = 1$



(b) $\rho\tilde{\nu}/\mu_\infty$ at $p = 3$

Figure 6.33: Computed $\rho\tilde{\nu}/\mu_\infty$ contours using the Spalart-Allmaras turbulence model for flow over the AGARD L1T2 high-lift multi-element airfoil configuration with mean flow discretization orders $p = 1$ and $p = 3$and a first-order discretization for the turbulence model at $M_\infty = .197$, $\alpha = 20.18^o$, and $Re = 3,520,000$.

without any form of artificial diffusion or limitation. Figure 6.34(a) shows a comparison between computed surface pressure coefficients and experimental values. The computed surface pressure coefficient results agree well with experimental values throughout the airfoil sections. Figure 6.34(b) depicts the computed skin friction coefficient using a $p = 3$ DG discretization for the mean flow equations and a smooth skin friction profile is obtained with the exception of the geometry slope discontinuities. Table 6.3 provides the numerical values of the computed lift and drag coefficients for each discretization order p. Table 6.3 shows that the computed drag coefficient is resolved to within 4 counts and the computed lift coefficient is resolved to within 11 counts. Figure 6.35(a) depicts the streamlines around



(a) Surface pressure coefficient at $p = 3$

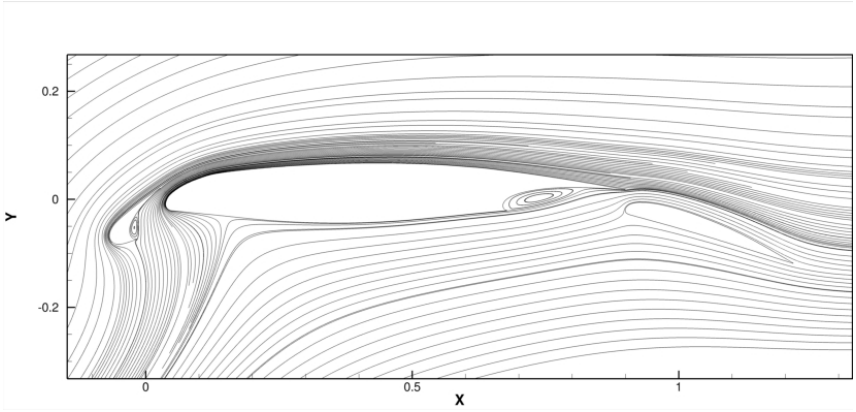
(b) Surface pressure coefficient at $p = 3$

Figure 6.34: Computed surface pressure and skin friction coefficients using the Spalart-Allmaras turbulence model for flow over the AGARD L1T2 high-lift multi-element airfoil configuration with a mean flow discretization order of $p = 3$, $M_\infty = .197$, $\alpha = 20.18^o$, and $Re = 3,520,000$.

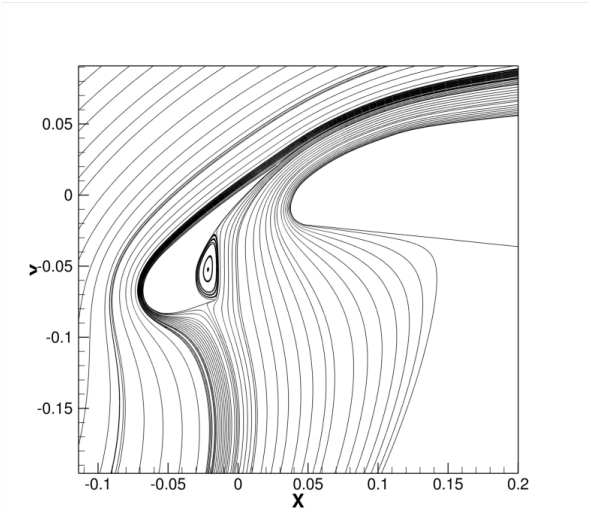
the L1T2 multi-element airfoil configuration, showing the high flow incidence angle and high overall streamline curvature over the configuration. Figure 6.35(b) shows the streamlines near the slat for this case, illustrating the high streamline curvature in this region as the flow is accelerated around the leading edge of the slat and in the gap between the slat and main airfoil. Figure 6.35(c) shows the streamlines near the flap and flap cove on the main element showing a strong re-circulation region in the flap-cove.

6.5.4 High-lift Multi-element Airfoil Configuration 30P30N: *hp*-adaptation

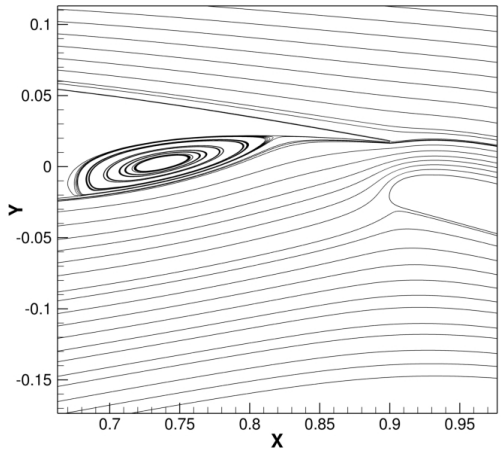
Since the previous hybrid discretization results do not increase turbulence model resolution as the discretization order is varied, the *hp*-adaptation strategy is employed in order to study the effects of increasing turbulence model resolution in regions of the domain where the turbulence model discretization error is the dominant contribution to the functional error estimate. The final turbulent flow test case consists of computing the flow over the 30P30N multi-element airfoil configuration using a lift-driven *hp*-adaptation. This configuration is the



(a) Streamlines



(b) Slat streamlines



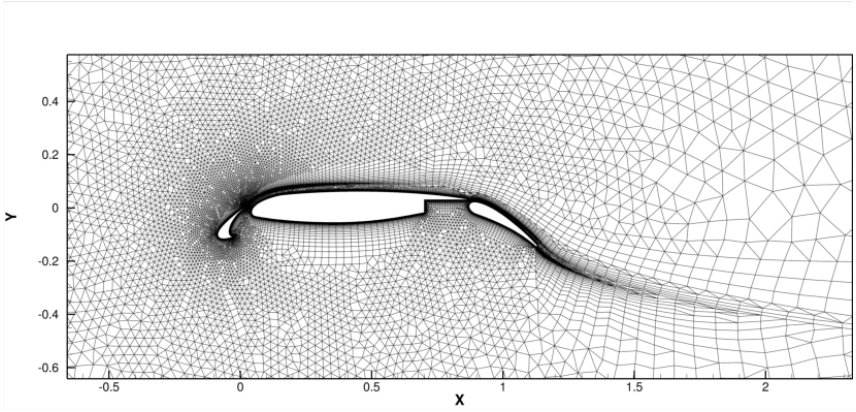
(c) Flap streamlines

Figure 6.35: Streamlines near the slat and flap using the Spalart-Allmaras turbulence model for flow over an L1T2 high lift multi-element airfoil with a mean flow discretization order of $p = 3$, $M_\infty = .197$, $\alpha = 20.18^o$, and $Re = 3,520,000$.

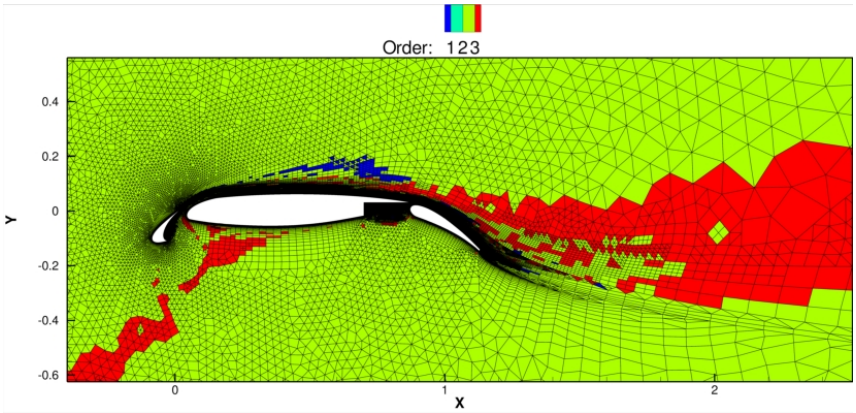
same multi-element airfoil configuration considered in Section 6.5.2. The flow conditions are $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$, as in Section 6.5.2. In this case, the *hp*-adaptation algorithm is slightly modified to accommodate the first-order finite-volume discretization of the turbulence model. The computable error in equation (5.2.11) can be broken up into contributions from the different PDEs in the system by taking the inner product only over the modal coefficients separately for each field in the system of equations. When the contribution to the total error from the turbulence model equation exceeds 50% of the total error for an element, the element is flagged for *h*-refinement regardless of smoothness. This is designed

to target the contribution of the turbulence model discretization error to the functional error estimate with h-refinement, since the standard hp-adaptive approach might otherwise employ p-enrichment everywhere because the mean flow is smooth and the turbulence model smoothness is irrelevant due to the first-order discretization.

Figure 6.36(a) depicts the initial mesh with $N = 55,964$ elements with a discretization order $p = 1$. Figure 6.36(b) depicts the final hp-adapted mesh, which is refined using h-



(a) Initial mesh: $N = 55,964$ and $p = 1$

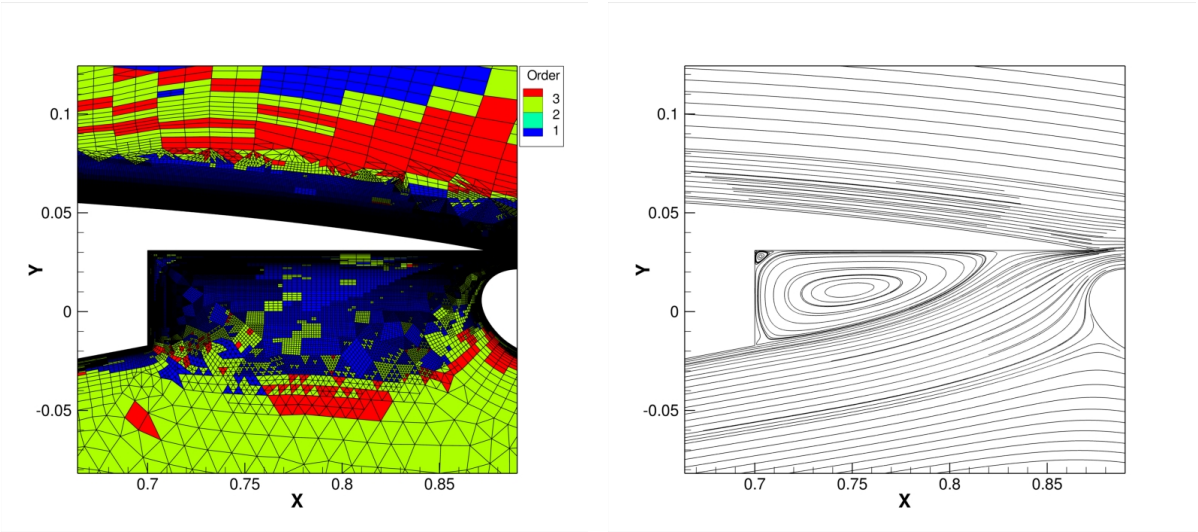


(b) Final mesh: $N = 536,261$ and $p = 1$ to $p = 3$

Figure 6.36: Unstructured mixed-element meshes used for computing flow around the 30P30N high-lift multi-element airfoil configuration at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$ using hp-adaptation.

refinement in regions where the turbulence model discretization error is the dominant source of the lift error estimate. The turbulence model discretization error dominates the lift error estimate in regions such as the flap and slat coves as well as the boundary layer along the upper surface of the main airfoil element. Figure 6.37(a) shows the mesh and discretization

order in the flap cove region on the final *hp*-adapted mesh. Aggressive mesh refinement is applied in this region due to the dominance of turbulence model error contribution to the lift error estimate. Unfortunately, the mean flow would benefit from additional *p*-enrichment in this region due to the purely smooth reverse flow behavior as shown in Figure 6.37(b). In



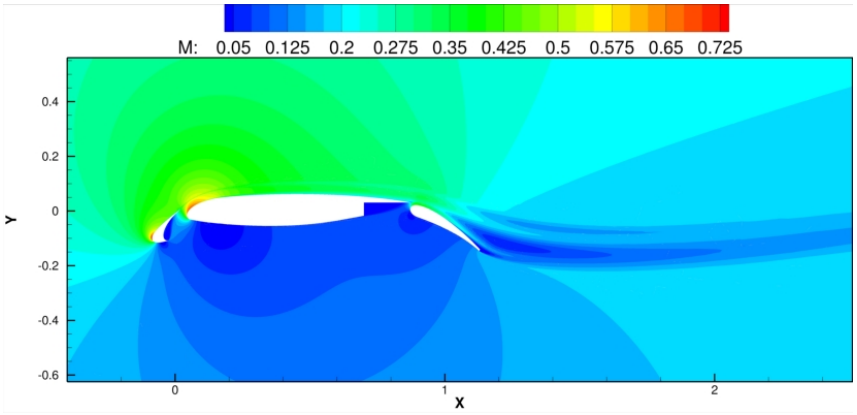
(a) Close-up of flap cove mesh

(b) Streamlines in flap cove

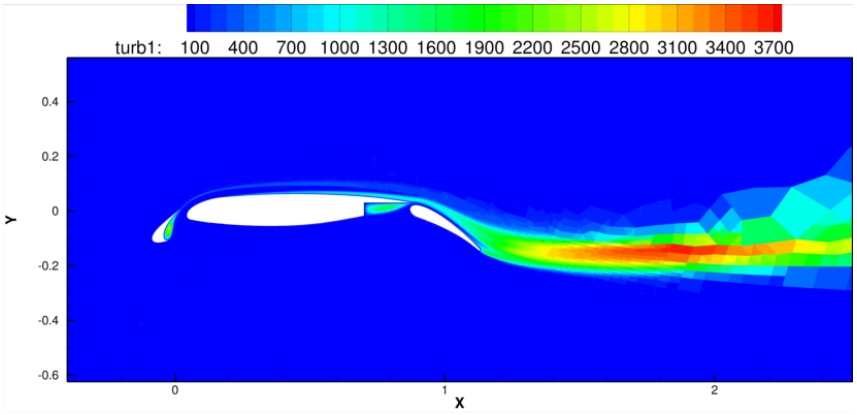
Figure 6.37: Close-up of the mesh and stream lines in the flap cove on the final *hp*-adapted mesh for the flow around the 30P30N high-lift multi-element airfoil configuration at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$.

general, the part of the flow that is external to the boundary layer and the wake is refined using *p*-enrichment, since in these regions the turbulence model influence on the solution is minimal.

The *hp*-adaptation is initialized with the $p = 1$ solution shown in Figure 6.27(a) and Figure 6.28(a). Figure 6.38(a) shows the computed Mach number contours on the final *hp*-adapted mesh, and Figure 6.38(b) shows the eddy viscosity contours on the final mesh. Clearly, *hp*-adaptation has added significant resolution to the discretizations of both the mean flow and turbulence model equations. As with the previous hybrid discretization results, this solution maintains positive values of the turbulence model working variable throughout the entire domain. As with the uniform *p*-enrichment case, the DG solver handles the high mean flow gradients without artificial viscosity or limitation. Figure 6.39(a) shows the computed surface pressure distribution on the final *hp*-adapted mesh compared



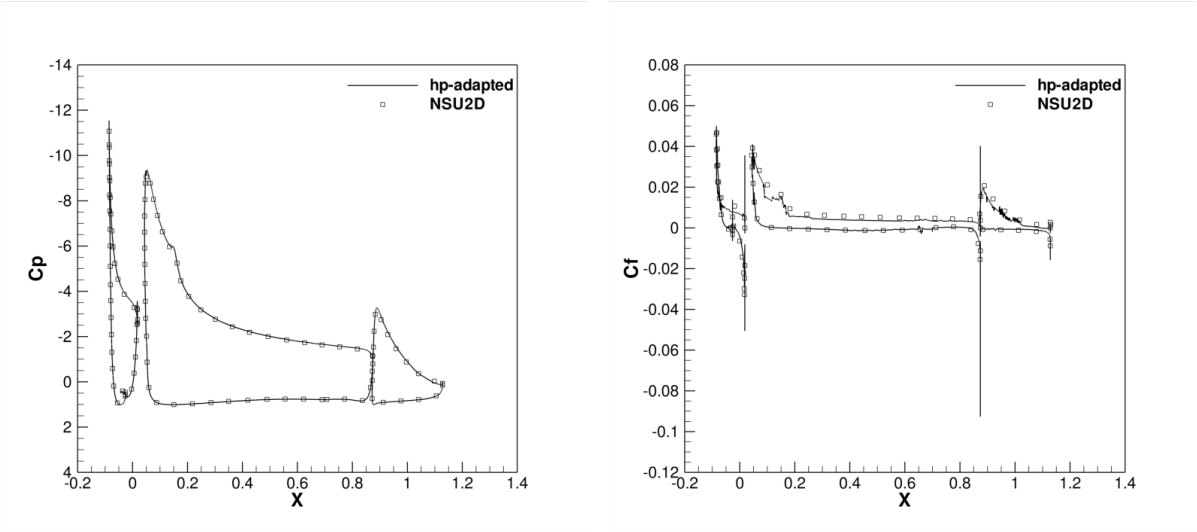
(a) Final Mesh: Mach Contours



(b) Final Mesh: $\frac{\rho\tilde{\nu}}{\mu_\infty}$ Contours

Figure 6.38: Computed Mach number and eddy viscosity contours on the final *hp*-adapted mesh for the flow around the 30P30N high-lift multi-element airfoil configuration at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$.

with the results obtained by the NSU2D flow solver. Figure 6.39(a) shows good agreement in the computed surface pressures obtained with both solvers. Furthermore, the computed surface pressure profile obtained using the DG solver is smooth. Figure 6.39(b) shows the computed surface skin friction profile compared with the NSU2D flow solver. In this case, reasonable agreement of computed skin friction profile between the two solvers is also obtained. However, the computed skin friction coefficient result obtained by the DG solver is significantly less smooth than the computed skin friction profile obtained by the NSU2D solver. Furthermore, comparing Figure 6.39(b) to Figure 6.29(b) shows that the *hp*-adapted computed skin friction coefficient is significantly less smooth than the computed skin friction coefficient obtained using uniform *p*-enrichment. The noise in the computed skin friction pro-



(a) Final mesh: Surface Pressure

(b) Final mesh:Skin Friction

Figure 6.39: Computed surface pressure and skin friction coefficients on the final *hp*-adapted mesh for turbulent flow over the 30P30N high-lift multi-element airfoil configuration at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$.

file results from the contribution of the turbulence model discretization error to the lift error estimate that is sufficiently high in some regions, that p-enrichment of the flow is prevented in an area where the mean flow would benefit from p-enrichment. One such region is the leading edge of the main element as shown in Figure 6.40, where the elements next to the wall employ a discretization order of $p = 1$ for the mean flow.

The computed lift and drag coefficient histories over the *hp*-adaptation process are shown in Figure 6.41(a) and Figure 6.41(b) respectively. In addition to the *hp*-adaptation histories, the uniform p-enrichment computed lift and drag coefficient histories are also shown. The computed lift and drag grid convergence behavior represents a significant improvement over that of the NACA0012 airfoil case in Section 6.3.3. Both the computed lift and drag coefficients are approaching a fixed value. Unfortunately, the number of unknowns employed for the *hp*-adapted computation is very high, over 2 million in this case. The high number of unknowns employed is due to the first-order discretization of the turbulence model equation, and as such true grid convergence is beyond the computational budget available. Furthermore, comparison between the *hp*-adapted and uniformly p-enriched results show the computed lift and drag coefficients are approaching two different values for each refinement

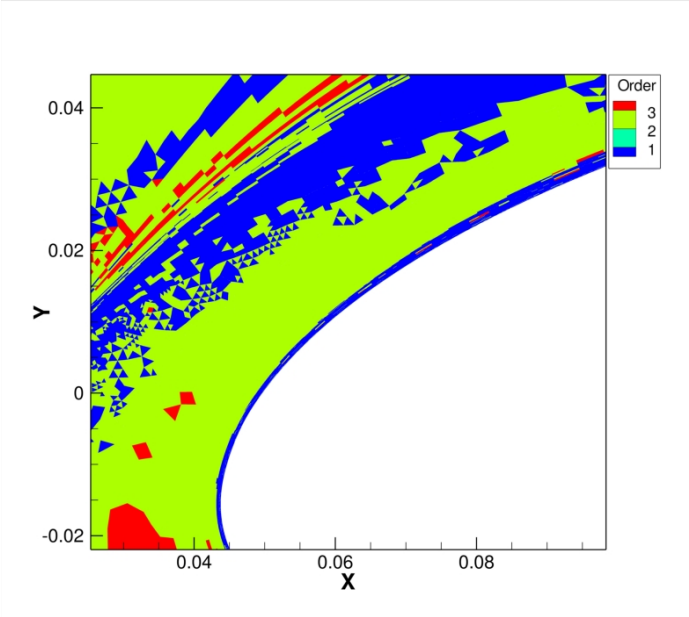
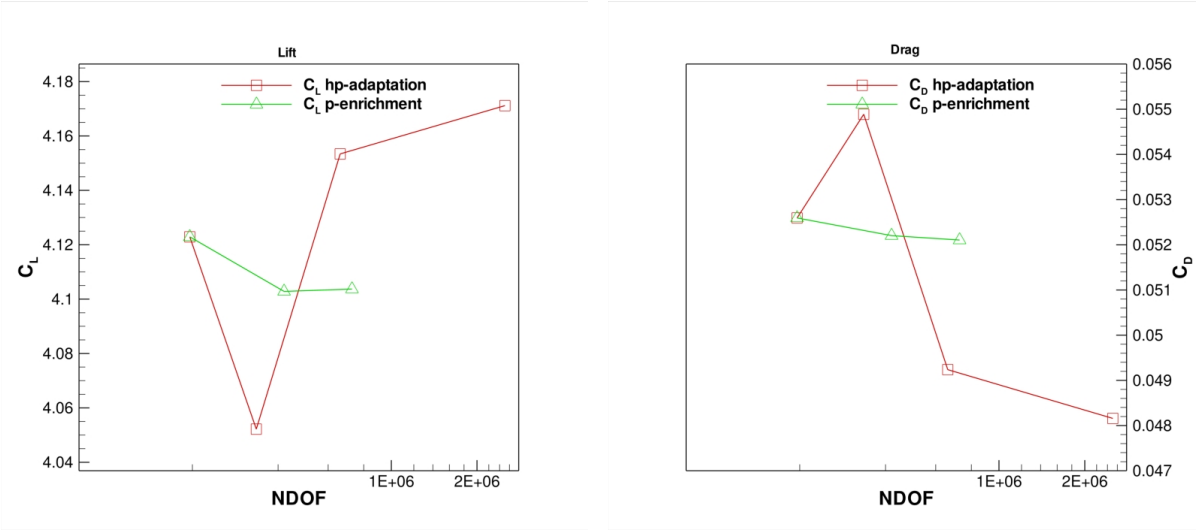


Figure 6.40: Close-up of order distribution at the nose of the main element of 30P30N high-lift multi-element airfoil configuration. Note that the elements in the boundary layer near the wall are employing a discretization order of $p = 1$.



(a) Computed Lift Coefficient

(b) Computed Drag Coefficient

Figure 6.41: Computed lift and drag coefficients history during adaptation of the flow around the 30P30N high lift configuration at $M_\infty = .2$, $\alpha = 16^o$, and $Re = 9,000,000$.

method respectively. The two refinement strategies converge to different computed lift and drag coefficients because, whereas *hp*-adaptation increases the turbulence model resolution, *p*-enrichment fixes the turbulence model resolution at the number elements in the initial

mesh. Clearly the turbulence model resolution has a significant impact on the computed lift and drag coefficient results for this case. Finally, Figure 6.42 depicts the adjoint lift error estimate, and, as expected, the estimate does not converge towards zero as *hp*-adaptation is applied. Just as was the case when applying a $p = 0$ discretization for shocked flows, the

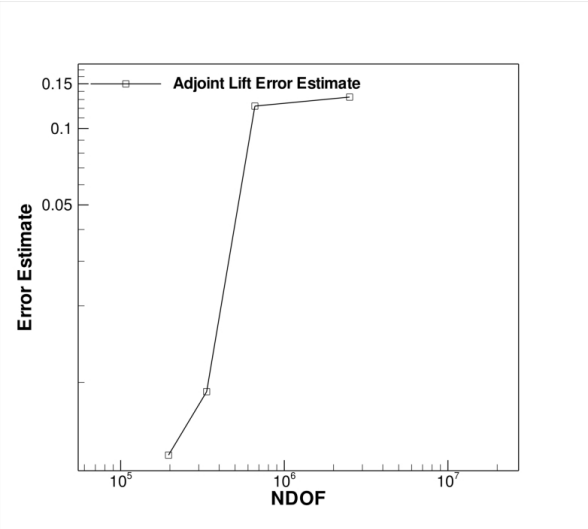


Figure 6.42: Computed adjoint error estimate of lift during the *hp*-adaptation for the flow over the 30P30N high-lift multi-element airfoil configuration.

projection into the high-order finite element space, which is used to approximate the fine level solution, causes a corrupted fine level residual estimate and hence a corrupt functional error estimate. Unfortunately, the techniques used to remedy this situation for shocked flows have proven ineffective for turbulent flows. However, the first-order approach to discretizing the turbulence model has at least removed the effect of the turbulence model discontinuity on the solution, allowing the presented DG solver to obtain improved functional convergence, as well as a robust solution strategy as demonstrated by the ability to compute high-lift flow solutions both with and without adaptation.

6.6 Summary

A DG solver capable of solving the RANS equations has been developed. When applying high-order methods to the discretization of the turbulence model a discontinuity in the turbulence model working variable develops at the edge of boundary layers and wakes, which

can often leads to solver failure. It was shown that the turbulence model working variable discontinuity appears not only when high-order DG discretizations are applied to the turbulence model equation, but also when higher than first-order finite-volume discretizations of the turbulence model equation are employed. This discontinuity in the turbulence model working variable induces negative effects on functional convergence and error estimation as well as robustness. In the finite-volume framework a first-order discretization of the turbulence model convection terms is often employed as in references [5, 64–66]. Employing this type of first-order finite-volume discretization of the turbulence model is shown to remove the negative turbulence variable values from the turbulence model solution, for both the finite-volume and DG mean flow discretizations allowing for robust high-order discretizations of the mean flow equations . The method is shown to be robust for a variety of problems including two high-lift multi-element airfoil configurations. Furthermore, the first-order turbulence model discretization is the recommended approach according to reference [41].

Thus, high-order DG discretizations of the mean flow equations are sufficiently robust for RANS simulations. Similar to finite-volume methods, a robust discretization of the turbulence model that provides sufficient eddy viscosity distributions can be constructed using a first-order accurate upwind discretization of the turbulence model convective terms. This approach is robust, however it is inefficient for reducing discretization error. In particular, whereas functional convergence behavior improves even for complicated problems, refinement becomes excessive, preventing true grid convergence of functional outputs. The results for all the presented test cases indicate that the turbulence model has a strong influence on functional accuracy. Unfortunately the benefits of high-order methods will not be realized until turbulence models that do not suffer from non-smooth behavior are developed. Such models will be more amenable to high-order discretization, thereby removing discretization error efficiently. Nevertheless, the presented approach to solving the RANS equations with a high-order DG discretization is robust and capable of solving are large variety of challenging aerodynamic flows. Since the discontinuity in the working variable of the turbulence model is not confined to DG methods (as shown in this work), the discontinuity in the turbulence model working variable is not a pitfall of DG methods, but rather a symptom of a fundamen-

tal flaws in the turbulence model equation that are largely undiscussed in the literature (with the exception of references [15, 20, 70, 74]). Unfortunately high-order DG discretizations of the turbulence model make the turbulence model working variable discontinuity more difficult to treat, due to the coupling between the number of degrees of freedom and order of accuracy. Optimal high-order methods for high Reynolds number turbulent flows($Re > 10^6$) will require the development of new turbulence models that are designed to be used with high-order methods allowing for low discretization error solutions.

Based on the results presented in this chapter it is questionable whether the combination of high-order discretization and RANS turbulence models represent an enhancement of the state-of-the-art. The discontinuity in the turbulence model working variable affects the robustness of the solver as well as the accuracy of functional outputs. However, alternative turbulence treatments such as LES and DNS, where many of the effects described in this chapter, may benefit from the low discretization error properties of high-order methods. However, LES and DNS simulations are very computationally expensive especially for the Reynolds numbers considered in this work.

While the presented results, including those of previous sections indicate that the turbulence model working variable discontinuity causes poor functional convergence and incorrect error estimation, the possibility remains that the high mean flow gradients often found in turbulent flows may contribute to poor functional convergence. The next chapter deals with the treatment of high Mach and Reynolds number but laminar hypersonic flows. The results of this chapter will show that in high mean flow gradient situations high-order DG discretizations can achieve grid convergence of functionals and accurate error estimates, provided all sources of non-smooth behavior can be adequately regularized(i.e. shock waves using artificial diffusion) and provided that the discretization is dual consistent.

Chapter 7

Hypersonic Flows

Shock capturing techniques for high-order methods have become a rich and intense area of research. However, significantly less attention has been paid to the effectiveness of capturing shock waves using high-order discretizations. This work examines the effectiveness of capturing shock waves using high-order polynomials from a robustness and error reduction point of view. Particular attention is paid to the most appropriate method of refinement for a shock wave. It is shown that when one considers exclusively shock capturing accuracy, mesh refinement employing a second-order discretization is significantly more effective at reducing error than order enrichment, despite the sub-cell shock wave resolution of high-order solutions. Furthermore, for flows involving shock waves in combination with smooth features, high-order discretizations are shown to be particularly effective when used as part of an *hp*-adaptation strategy. *hp*-adaptation is shown to yield superior efficiency compared with mesh refinement at second-order accuracy alone. To demonstrate the robustness of the proposed approach, hypersonic ($M_\infty \geq 6.0$) applications are considered exclusively. It is demonstrated that *hp*-adaptation combined with PDE-based artificial viscosity is capable of robustly obtaining low discretization error with accurate and smooth surface heating profiles.

7.1 Introduction

In recent years shock capturing techniques for high-order methods have been investigated by many authors [16, 17, 28, 34, 37, 38, 47, 67, 95, 99, 100]. Limiters for higher-order DG methods have been actively pursued in references [35, 67, 95, 99] and others. As mentioned in reference [47], discontinuous Galerkin(DG) methods couple the order of accuracy and the number of degrees of freedom (DoFs). Contrarily, finite-volume or finite-difference methods, from which the ideas of limitation originate, do not couple the order accuracy and DoFs. As such, limiters for high-order DG methods must take the coupling between the order of accuracy and DoFs into account. Preferably, accounting for this aspect of DG methods should not extend the numerical stencil beyond the nearest neighbors, which can degrade the parallel efficiency of the solver by increasing communication cost. Unfortunately, the limiter-based stabilization methods mentioned above do just that. Additionally, exact linearizations of the limiter functions are difficult to obtain, impacting the efficiency and robustness of implicit solution techniques. Furthermore, it is yet unclear how limiters will affect the discretization of viscous fluxes as the high-order information that is truncated to stabilize the inviscid fluxes might be required in the viscous flux construction.

As an alternative, many researchers [28, 34, 36–38, 100] have proposed methods that seek to capture shock waves using selective artificial viscosity(A.V.) in a diffusion operator. The goal of such as technique is to smear the shock wave over a scale that is resolvable by the polynomial basis that exists within an element. Chapter 5 demonstrated the difficulty of applying the method of reference [34] to the high-order solution of an inviscid transonic flow over a NACA0012 airfoil. In this case, despite the achieved sub-cell resolution of the shock wave, the accuracy of the computed lift coefficient was not improved as the discretization order p was increased. Furthermore, reference [71] stated that robustness problems prevented the author of this reference from using higher than a discretization order of $p = 2$ for an adaptive mesh simulation of a viscous hypersonic flow. However, despite these limitations, artificial viscosity methods are still considered in this work because there are several advantages to this type of robustness enhancement technique. These advantages that make artificial viscosity methods attractive for use with high-order DG discretizations are: poten-

tially compact stencil, straightforward application to viscous flow problems, availability of the exact linearization for implicit flow solution, control of the fine level residual estimates used in output error estimation (equation (5.2.10)) and awareness of the higher number of unknowns associated with high-order DG elements. It should be mentioned that of the methods in references [28, 34, 37], the PDE-based artificial viscosity method of references [37, 71] has been found to be the most robust. However, the PDE-based method is also the most diffusive and least likely to attain sub-cell shock wave resolution. In order for an artificial viscosity method to be considered robust it must be able to adequately stabilize the solution, which may be contradictory to the requirements of sub-cell shock wave resolution. Even when employing a $p = 1$ discretization, a robust artificial viscosity method is required to stabilize the solution for the strong shock waves considered in this chapter. For clarity, attempting to attain sub-cell shock wave resolution is what leads to robustness problems. Observations made during the course of this work have demonstrated that smoothly varying artificial viscosity is critical to a robust artificial viscosity method.

Yet another alternative technique for shock capturing is to employ an *hp*-adaptive procedure. This was initially applied to inviscid flows by Wang and Mavriplis in reference [17], and later applied to inviscid and viscous flows in combination with piece-wise constant artificial viscosity in Chapter 5 and in references [47, 68]. The basic idea is to initialize the domain with a uniformly first-order solution everywhere and raise the discretization order only in regions of the domain where the solution is smooth, and apply mesh refinement in regions of the solution that are non-smooth. This idea is very similar to slope limitation with the exception that it is a bottom up approach rather than a top down approach. By making limitation a bottom up approach and applying mesh refinement in areas where the solution is non-smooth, this method can take the coupling between order of accuracy and number of unknowns into account, without performing any extended stencil operations. Also, no information is truncated in a bottom up approach, rather information is added over the adaptive simulation. While this approach shows great promise, *hp*-adaptation has two key drawbacks. Firstly, *hp*-adaptation without artificial viscosity requires initializing the simulation with a uniformly first-order solution everywhere, which is inappropriate for viscous

flows using compact DG diffusion discretizations, such as SIP [57]. Secondly, if the shock wave moves into a high-order cell at any time during the *hp*-adaptation procedure the solver will fail. While this is rather uncommon for inviscid flows it can easily occur for viscous flows, such as a shock boundary layer interaction on an inclined wedge. However, it seems logical that a solver that employs a good artificial viscosity method and *hp*-adaptation could result in a significantly more robust yet still accurate solver.

The goal of this work is to quantitatively determine if resolving shock waves using high-order DG discretizations is effective compared with using mesh refinement employing a second-order $p = 1$ discretization. Furthermore, this work will demonstrate how high-order DG discretizations can be applied such that the solver remains robust and retains beneficial error properties. In this work a discretization order $p = 1$ is employed as the minimum discretization order so that viscous flow problems may be discretized adequately at the outset of the refinement proceedure. However, employing a minimum discretization order of $p = 1$ necessitates the use of a shock capturing strategy. In this work, the PDE-based artificial viscosity of references [37, 71], which is described in Section 2.7.2, is employed as that shock capturing strategy. The purpose of this work is not to determine whether capturing shock waves with high-order discretizations can be accomplished, which has been determined conclusively in previous work [37,71]. Rather the focus of this work is to determine if capturing shock waves using high-order discretizations offers any efficiency benefits over capturing shock waves using mesh refinement and a second-order accurate discretization. Furthermore, this work focuses on appropriate refinement strategies so that the overall solution method is as robust as the current state-of-the-art low-order (e.g. second-order finite-volume) methods but retains the error properties of high-order methods. Assuming that capturing shock waves efficiently and robustly is a matter of choosing the appropriate refinement strategy, then perhaps capturing shock waves using high-order methods, such as DG, can be considered a solved problem. Both artificial viscosity and refinement method play large roles in obtaining a robust high-order DG based CFD solver.

7.1.1 Artificial Viscosity Settings

The artificial viscosity method of Section 2.7.2 used in this work requires the setting of various constants. The constant ϵ_0 controls the magnitude of the source term in equation (2.7.4) and equation (2.7.5). The constants κ and c_{s_0} control minimum value of the shock detector that will trigger artificial viscosity as seen in equation (2.7.5). This sections gives the values of these constants for the cases considered in this chapter. Table 7.1 gives the artificial viscosity constants used for the test cases presented in this Chapter. These settings

Table 7.1: Artificial viscosity constants for each case in this work.

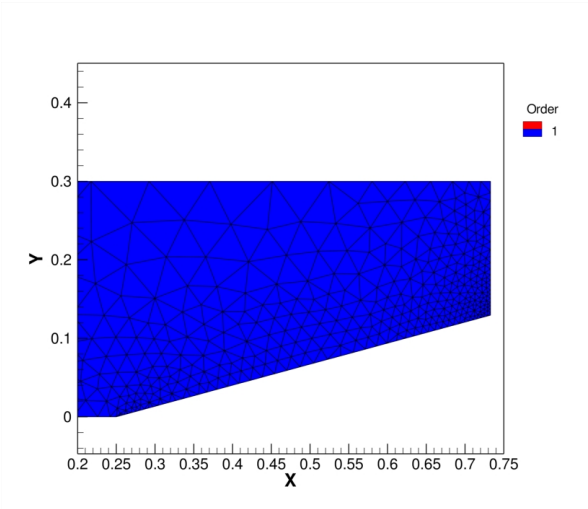
Case	ϵ_0	c_{s_0}	κ
Inviscid Ramp (Section 7.2)	1.0	1.0	2.75
Inviscid Half Cylinder (Section 7.3)	1.0	1.0	2.5
Viscous Half Cylinder (Section 7.4)	1.0	1.0	3.0

are kept identical across all adaptation cycles for each case.

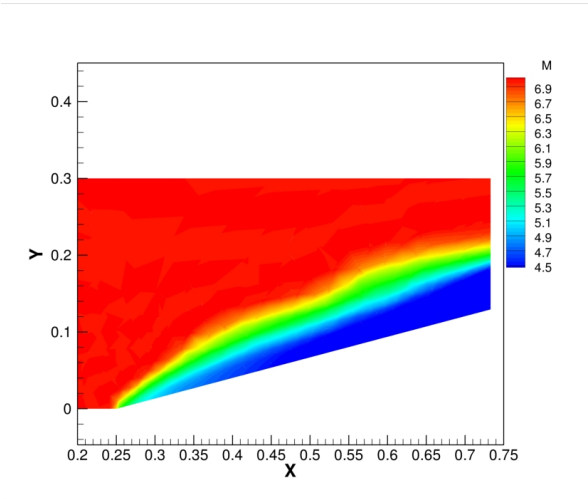
7.2 Hypersonic Inviscid Wedge

This test solves the inviscid flow over a 15^o wedge. The flow conditions are $M_\infty = 7.0$, $\alpha = 0.0^o$. The flow structure is made of up two constant states separated by an oblique shock. Thus the only flow feature in the solution that is required to be resolved by the DG solver is the shock wave, as resolving constant states is trivial. This case has an exact solution that can be computed via standard gas dynamics methods [101, 102]. The local error estimate given by equation (5.2.12) is used to drive the adaptive procedure. In this case h-refinement and p-enrichment are employed in isolation in order to determine the most effective method for reducing the error in the computed drag on the wall. Furthermore, since computed drag is a function of the pressure P behind the shock, which is a constant, obtaining accurate computed drag values depends only on adequately resolving the shock wave, i.e. obtaining a sharp shock wave at the proper location. For this test case the computed drag error is determined by comparing the computed drag value to a reference drag value which is determined from the exact solution, i.e $C_{D_{error}} = \left|C_D - C_{D_{ref}}\right|$. A

total of four adaptation cycles are performed using both the h-refinement and p-enrichment strategies.

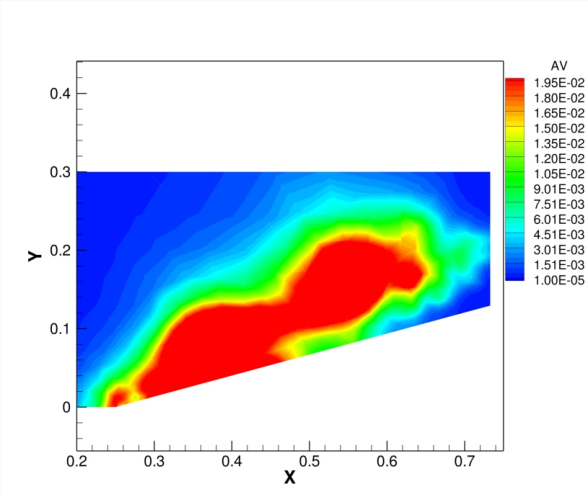


(a) Initial grid: 713 elements, $p = 1$



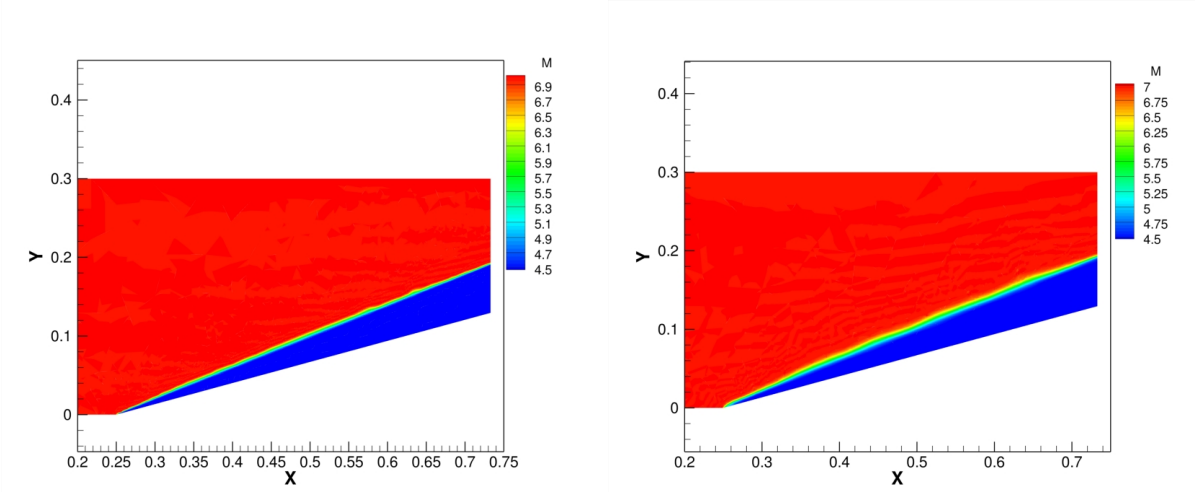
(b) Mach number contours: initial grid

Figure 7.1: Initial mesh and Mach number contours of inviscid hypersonic flow over 15^o wedge with $p = 1$, $M_\infty = 7$, $\alpha = 0^o$.



(a) Artificial viscosity contours: Initial Grid

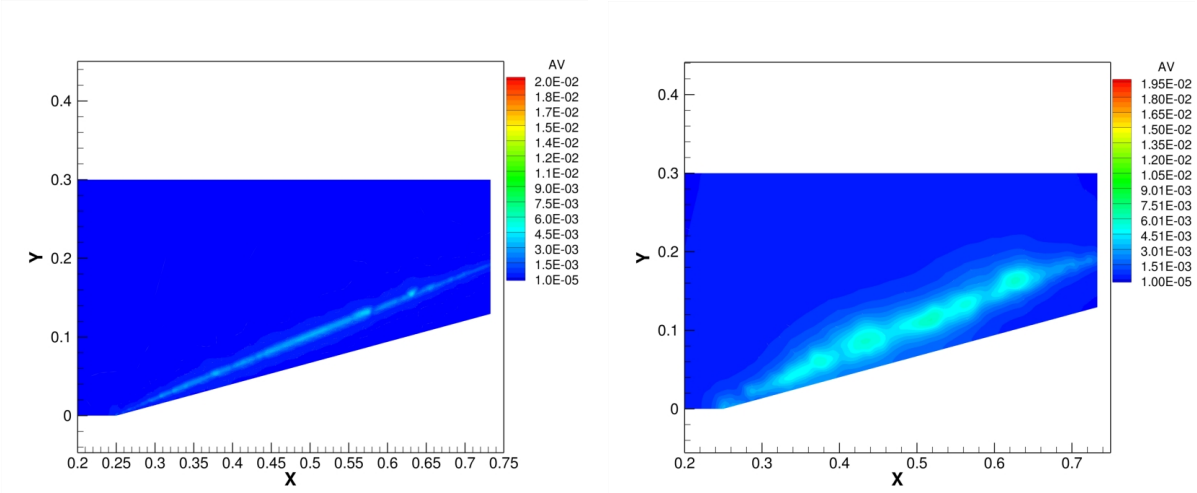
Figure 7.2: Initial artificial viscosity contours of inviscid hypersonic flow over 15^o wedge with $p = 1$, $M_\infty = 7$, $\alpha = 0^o$.



(a) Mach number contours: final h-refined grid, $p = 1$

(b) Mach number contours: final p-enriched grid, $p = 1$ to $p = 4$

Figure 7.3: Final h-refinement and p-enrichment Mach number contours of inviscid hypersonic flow over 15^o wedge with , $M_\infty = 7$, $\alpha = 0^o$.

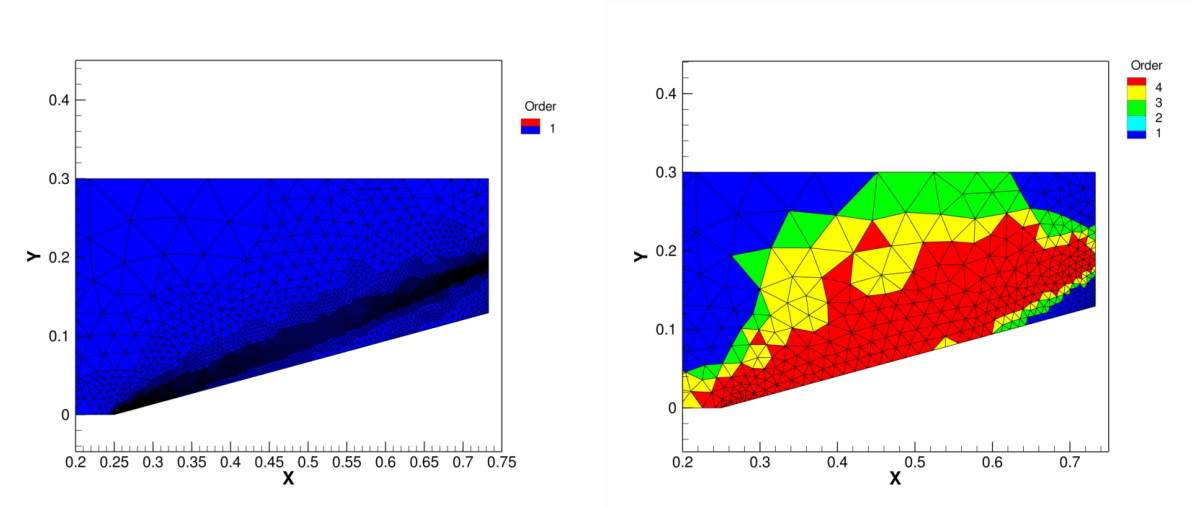


(a) Artificial viscosity contours: final h-refined grid, $p = 1$

(b) Artificial viscosity contours: final p-enriched grid, $p = 1$ to $p = 4$

Figure 7.4: Final h-refinement and p-enrichment artificial viscosity contours of inviscid hypersonic flow over 15^o wedge with , $M_\infty = 7$, $\alpha = 0^o$.

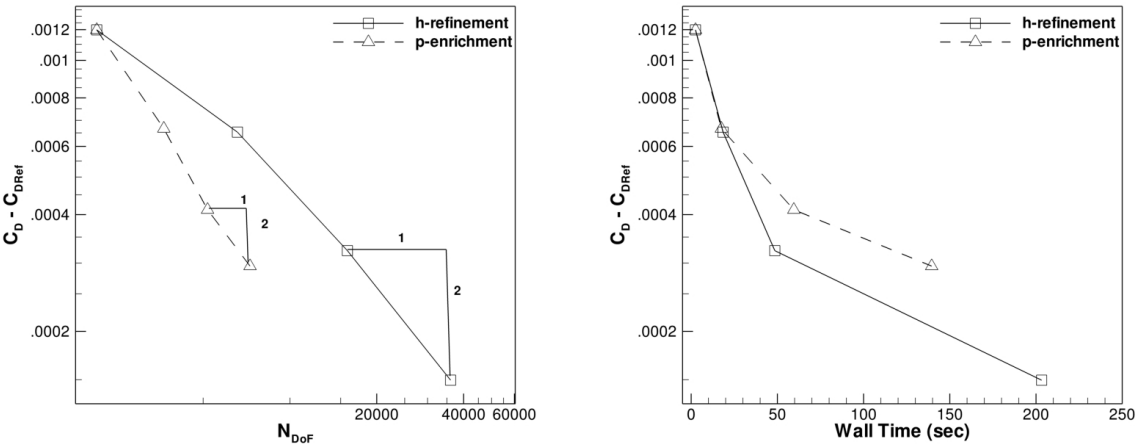
The initial grid and Mach number contours for this case are depicted in Figure 7.1(a) and Figure 7.1(b). As expected, for this coarse mesh the shock wave is quite poorly resolved but is captured robustly thanks to the smooth artificial viscosity. Figure 7.2(a) depicts the initial artificial viscosity contours, illustrating the wide extent of the artificial viscosity distribution



(a) Final grid: h-refinement, $p = 1$

(b) Final grid: p-enrichment, $p = 1$ to $p = 4$

Figure 7.5: Final h-refinement and p-enrichment grid used for computing the inviscid hypersonic flow over 15^o wedge, $M_\infty = 7$, $\alpha = 0^o$.



(a) Drag error versus N_{DoF}

(b) Drag error versus wall time (sec)

Figure 7.6: Drag error versus N_{DoF} and wall time for an inviscid hypersonic flow over 15^o wedge with $p = 1$, $M_\infty = 7$, $\alpha = 0^o$.

resulting from the use of the PDE-based artificial viscosity for a strong shock wave. The wide extent of the artificial viscosity is a property of the PDE-based method, which makes this approach more robust than other artificial viscosity methods. The extent and magnitude of the artificial viscosity distribution can be used as qualitative measures of the solution. Adaptive simulations employing both h-refinement and p-enrichment are performed and the

resulting computed Mach number and artificial viscosity contours are shown in Figure 7.3(a) through Figure 7.4(b). Additionally, the computational meshes, which are purely triangular unstructured meshes, are shown in Figure 7.4(a) and Figure 7.4(b) for h-refinement and p-enrichment adaptations respectively. While both adaptation methods improve the shock resolution, the h-refinement adaptation has reduced the area of effect of the artificial viscosity significantly more than p-enrichment, which results in an overall sharper computed shock wave using the same number of adaptation cycles. While not immediately apparent from Figure 7.3(b), p-enrichment has captured the shock wave within approximately one cell.

Figure 7.6(a) and Figure 7.6(b) show the computed drag error versus N_{DoF} and computational time, employing both adaptation strategies, respectively. While p-enrichment uses fewer DoF for a given error level it is not the most effective adaptive method in terms of computational time, as depicted in Figure 7.6(b). Despite the near sub-cell shock wave resolution of the p-enrichment adaptation, h-refinement using a discretization order of $p = 1$ is more effective in terms of the computational time required to reduce the computed drag error. p-enrichment is well known to be more effective than h-refinement(at $p = 1$) for smooth flows [17, 31, 47, 96]. This is due largely to the exponential error convergence behavior of high-order methods applied to smooth problems. Exponential error convergence is the primary reason high-order methods add fewer degrees of freedom and generate lower error than h-refinement at $p = 1$. However, for shock waves the exponential behavior of the error is lost and p-enrichment behaves more like h-refinement for a non-smooth flow with artificial viscosity. The loss of exponential error convergence is the primary reason why p-enrichment is not as effective in terms of computational time. A second source of increased computational time can be attributed to the extra Newton steps required to obtain the steady-state solution. These extra Newton steps result from transients in the PDE governing the artificial viscosity. These transients are induced by adjustments to the artificial viscosity distribution required to stabilize the progressively higher-order solutions. Thus as far as shock wave resolution is concerned h-refinement is more effective than p-enrichment.

As stated previously, a special effort is made to maintain consistent parameters of the artificial viscosity PDE source term during the p-enrichment procedure as shown in Table

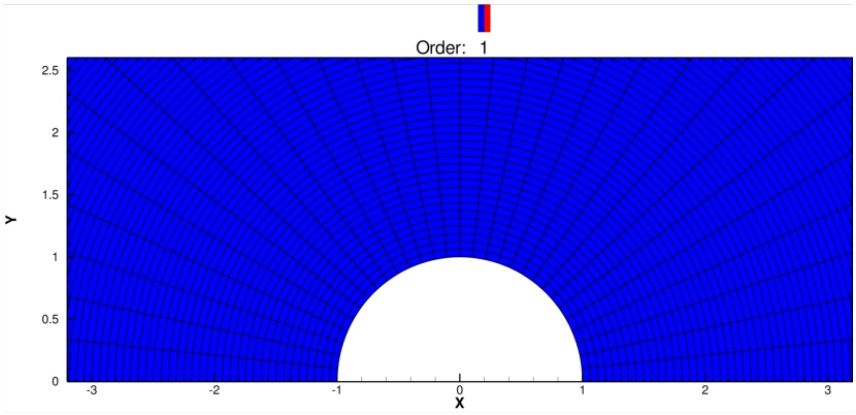
7.1. Consequently there is a small point in the final p-enrichment mesh where the pressure and sound speed have undershot the free stream value. While the undershoot is not enough to cause solver failure, this does speak to the robustness of capturing shock waves with high-order elements, namely h-refinement at a discretization order of $p = 1$ is more effective and more robust than p-enrichment, since all h-refinement solutions remain entirely monotone. Increasing the artificial viscosity in order to eliminate all undershoots would result in increased functional error in the p-enrichment results compared to the presented results.

7.3 Hypersonic Inviscid Cylinder

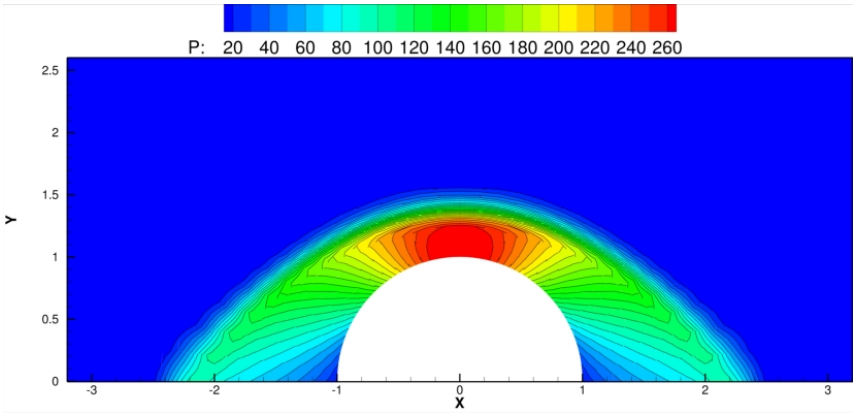
7.3.1 h-refinement versus p-enrichment

The second test case is the inviscid hypersonic flow over a half cylinder at $M_\infty = 17.605$ and $\alpha = 270^o$, which is an inviscid version of a hypersonic benchmark case given in reference [65]. As with the previous inviscid ramp test case, the local error estimate in equation (5.2.12) is used to drive the adaption procedure for this case. The error in computed drag is used as the basis to compare the accuracy of the h-refinement and p-enrichment results. In this case the drag error is measured relative to a reference solution obtained using a uniformly $p = 4$ solution on the finest adaptive mesh. Four adaptation cycles are performed using both the h-refinement and p-enrichment strategies. While this flow is governed very strongly by the shock wave, the post shock wave region contains a smooth flow feature (known as the shock layer) and therefore p-enrichment can be expected to perform better with this flow as compared to the previous test case.

Figure 7.7(a) through Figure 7.8(a) illustrate the initial mesh, pressure and artificial viscosity contours respectively. As with the ramp test case, the shock wave is thick and the artificial viscosity covers a wide area of the domain, extending well inside the shock layer and even touching the wall at the stagnation point. Both h-refinement and p-enrichment adaptations are performed using the local error indicator in equation (5.2.12) to drive the adaptation. The pressure contours on the final h-refinement and p-enrichment meshes are shown in Figure 7.9(a) and Figure 7.9(b) respectively. The artificial viscosity contours and



(a) Initial grid: $1,720$ elements, $p = 1$

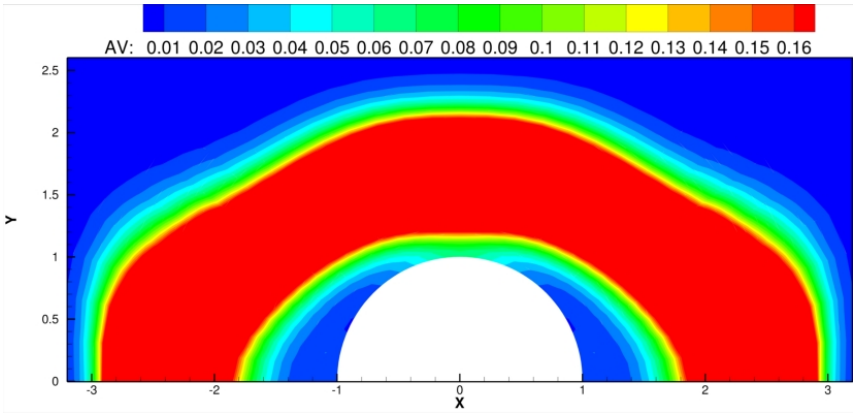


(b) Pressure contours: initial grid

Figure 7.7: Initial mesh and pressure contours of an inviscid hypersonic flow over a half cylinder with $p = 1$, $M_\infty = 17.605$, $\alpha = 270^o$.

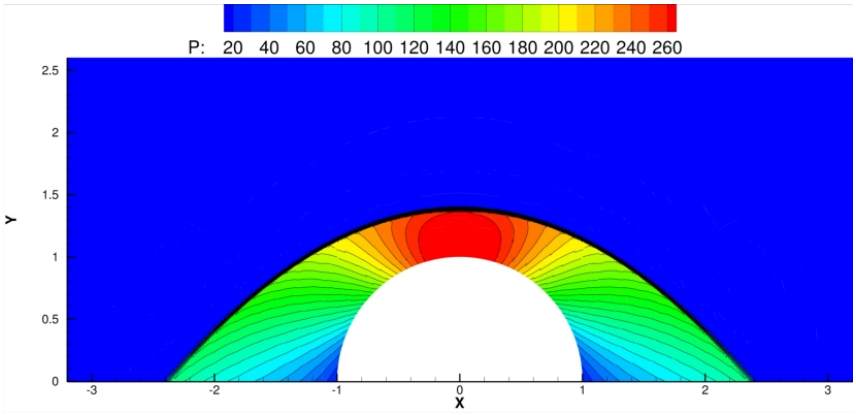
meshes are depicted in Figure 7.10(a) through Figure 7.11(b). This case clearly demonstrates that h-refinement has reduced the extent and magnitude of the artificial viscosity compared to p-enrichment. The area covered by non-zero values of the artificial viscosity is more than twice as large on the final p-refinement mesh compared to the final h-refinement mesh. Similarly to the ramp test case, both forms of refinement improve shock wave resolution, with h-refinement producing a sharper shock wave and p-enrichment producing a shock wave that is captured over approximately one and a half elements.

Figure 7.12(a) and Figure 7.12(b) show the drag error versus N_{DoF} and wall clock time using both adaptation methods. Performing p-refinement results in fewer degrees of freedom than performing h-refinement, but requires significantly more computation time

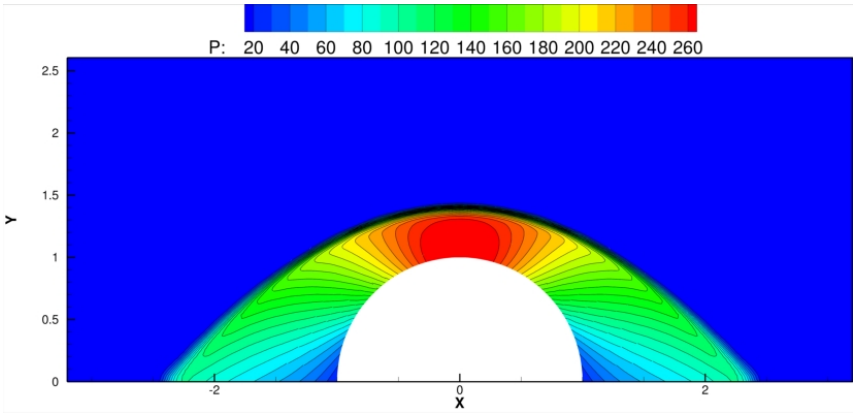


(a) Artificial viscosity contours: initial grid

Figure 7.8: Initial artificial viscosity contours of inviscid hypersonic flow over a half cylinder with $p = 1$, $M_\infty = 17.605$, $\alpha = 270^o$.

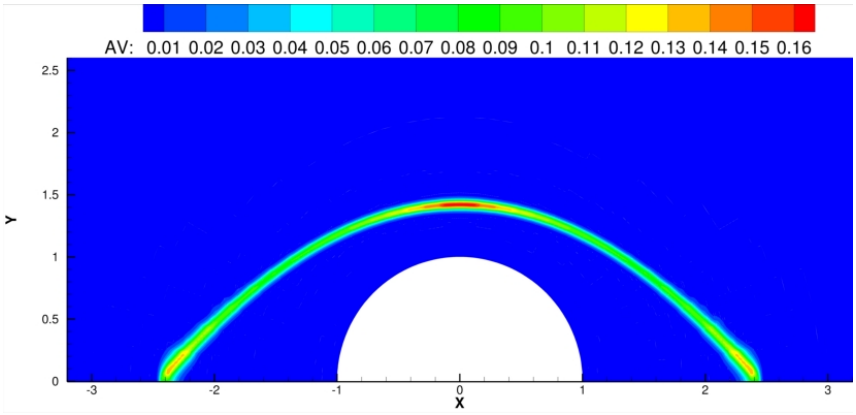


(a) Pressure contours: final h-refined grid, $p = 1$

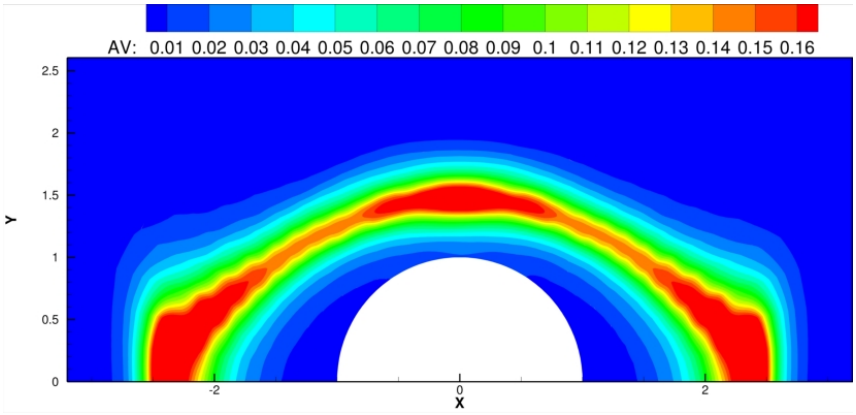


(b) Pressure contours: final p-enriched grid, $p = 1$ to $p = 4$

Figure 7.9: Computed pressure contours on the final h-refinement and p-enrichment meshes for inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$.



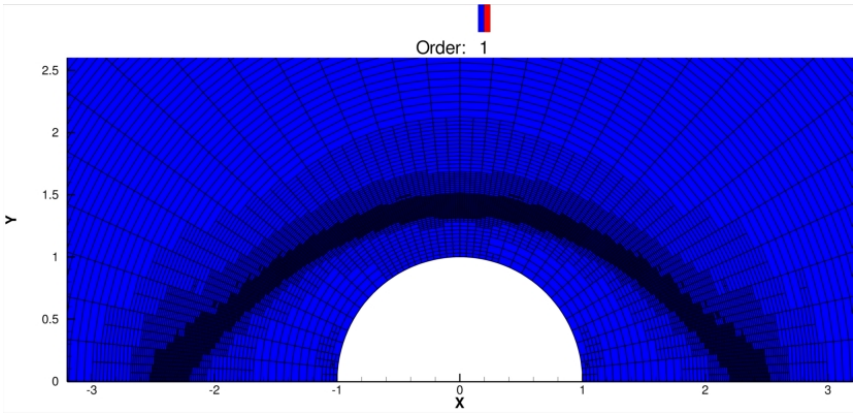
(a) Artificial viscosity contours: final h-refined grid, $p = 1$



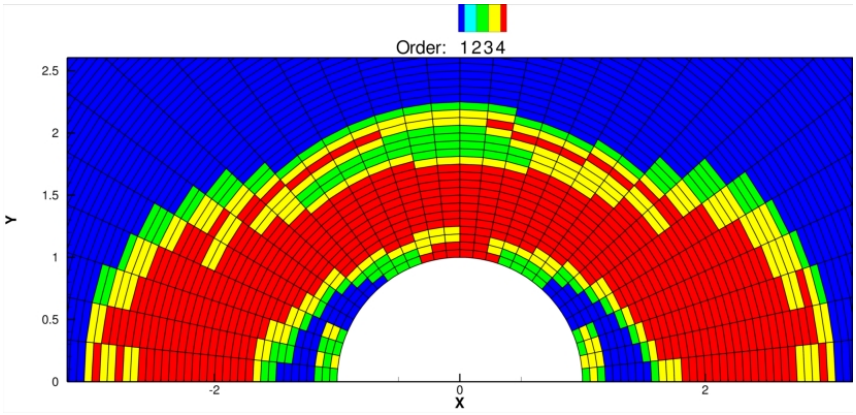
(b) Artificial viscosity contours: final p-enriched grid, $p = 1$ to $p = 4$

Figure 7.10: Artificial viscosity contours on the final h-refinement and p-enrichment meshes for inviscid hypersonic flow a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$.

than h-refinement. In fact this case shows more dramatically the superior efficiency of h-refinement because at the final adaptive step, h-refinement requires 1/3 the wall clock time to compute a solution with 2.25 times less or 55% lower error than the solution generated using p-enrichment. The sources of increased computational cost are similar to the previous inviscid wedge test case. In particular, the loss of exponential error convergence of the high-order solutions as well as the additional Newton steps required to converge the high-order solutions are sources of the additional computation time required to perform p-enrichment for this case. Additionally, there is a point of sizable undershoot in the sound speed at the final stage of the p-enrichment adaptation. Again this sound speed undershoot is not strong enough to cause solver failure, but induces some transients in the solution, which delay the



(a) Final grid: h-refinement, $p = 1$



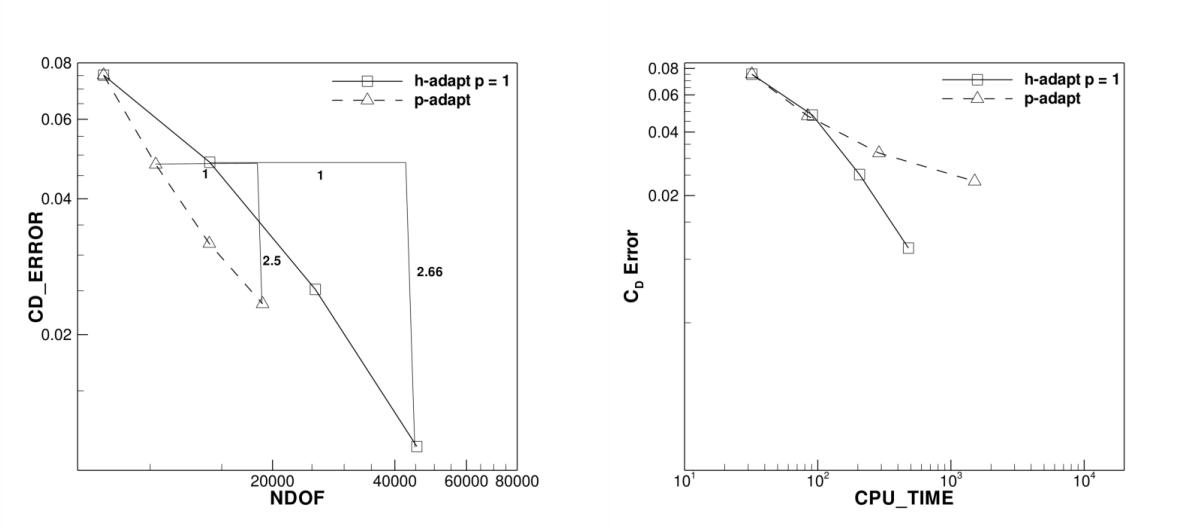
(b) Final grid: p-enrichment, $p = 1$ to $p = 4$

Figure 7.11: Final h-refinement and p-enrichment meshes for inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$.

convergence of the non-linear solver. h-refinement with a discretization order of $p = 1$ shows monotone behavior and as a result the convergence of the non-linear solver is not delayed at any point during the h-adaptation proceedure. Lastly Figure 7.13(a) depicts the stagnation line pressure distribution, which shows that a significantly sharper shock wave profile is obtained on the final h-refined mesh compared to the final p-enriched mesh.

7.3.2 h-refinement at Higher-Order

The comparison of h-refinement and p-enrichment shows that h-refinement is more effective at resolving shock waves than p-enrichment. However, these results have not substantiated that $p = 1$ is the most efficient discretization order at which to employ h-refinement. Thus it



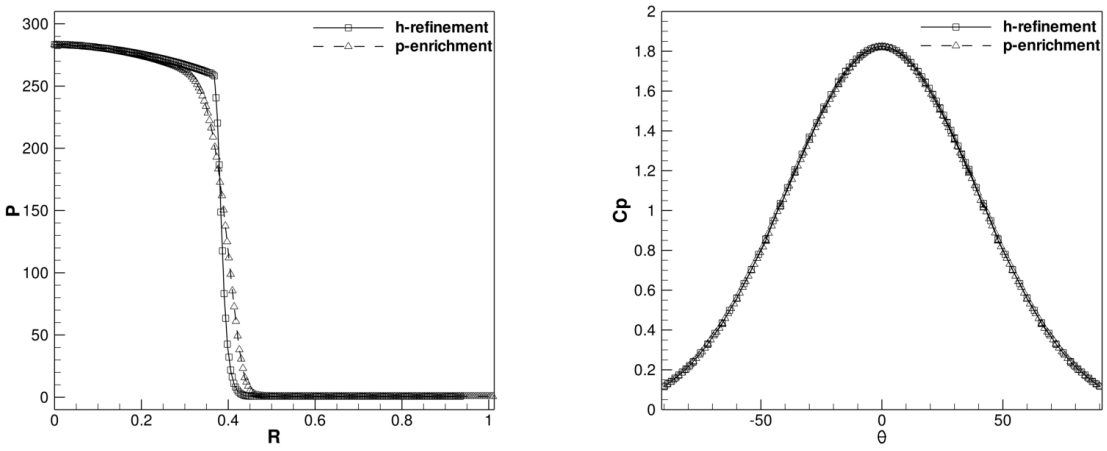
(a) Drag error versus N_{DoF}

(b) Drag error versus wall time (sec)

Figure 7.12: Computed drag error versus N_{DoF} and wall clock time for the inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$.

still remains to be determined if h-refinement using higher discretization orders is more effective than h-refinement using $p = 1$. In order to investigate the most effective discretization order at which to apply h-refinement, adaptive mesh refinement using uniform discretization orders of $p = 2$ and $p = 3$ is also applied to this test case. Figure 7.14(a) and Figure 7.14(b) show the final h-refined meshes for discretization orders $p = 2$ and $p = 3$ respectively. The h-refinement proceeds very similarly to the previously described $p = 1$ case. However, due to the sharper shock wave resolution on the initial mesh, the refinement zone narrows with increasing p. The resulting computed pressure contours are shown in Figure 7.15(a) and Figure 7.15(b) for discretization orders $p = 2$ and $p = 3$ respectively. The shock wave is captured very sharply in both cases as also seen by the narrow artificial viscosity profiles in Figure 7.16(a) and Figure 7.16(b). Figure 7.17 shows the computed stagnation pressure profiles on the final h-refined meshes employing discretization orders $p = 1$, $p = 2$, and $p = 3$, showing that sharper shock wave profiles are obtained as the discretization order p is increased.

The computed drag error across the h-refinement histories for discretization orders $p = 1$, $p = 2$, and $p = 3$ is shown in Figure 7.18(a) and Figure 7.18(b). While the slopes of computed

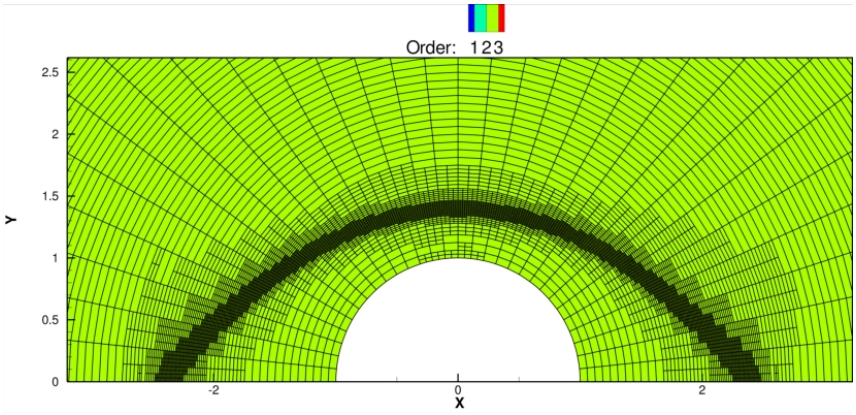


(a) Stagnation point pressure

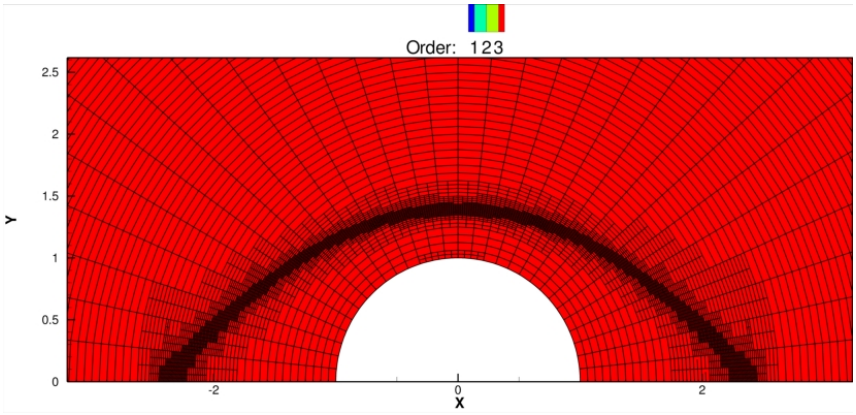
(b) Surface pressure coefficient

Figure 7.13: Stagnation point pressure profiles on the final adapted mesh using h-refinement and p-enrichment for inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$.

drag error curves show a slight increase as the discretization order p is increased, the increase in the slope of the computed drag error curve is not sufficient to see the benefits of high-order methods in terms of computation time. The benefits of high-order methods are not observed for this test because the slopes of the computed drag error curves versus $h(h = \sqrt{N_{DoF}})$ are not optimal. As discussed in Section 3.3 for a dual consistent discretization, which is employed for this case, the functional error should converge asymptotically as $O\left(h^{2p}\right)$. Thus optimal slopes of the drag error versus $h(h = \sqrt{N_{DoF}})$ would be 2, 4, and 6 for $p = 1$, $p = 2$, and $p = 3$ respectively. However, optimal computed drag error curve slopes are not obtained for higher-order discretizations of this flow problem. In particular for a discretization order of $p = 2$ the optimal slope should be 4, but the slope of the computed drag error data is 3 and for a discretization order of $p = 3$ the optimal drag error slope should be 6, but the computed drag error slope is 4. The comparison in terms of computational time shows that h-refinement at $p = 1$ is the most efficient approach overall.



(a) Final grid: h-refinement, $p = 2$

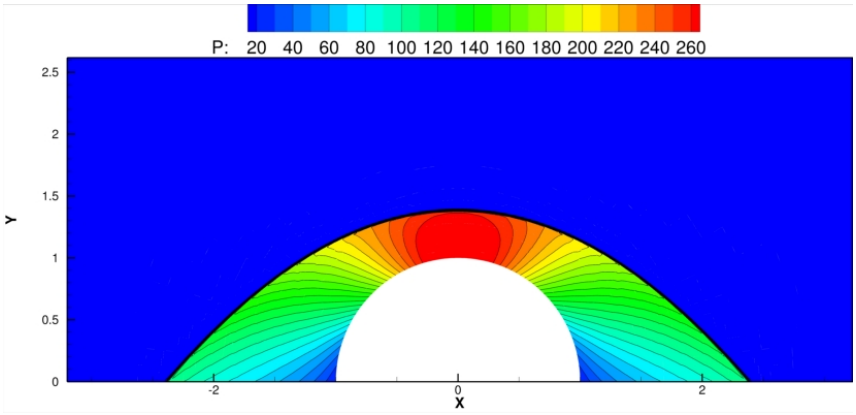


(b) Final grid: h-refinement, $p = 3$

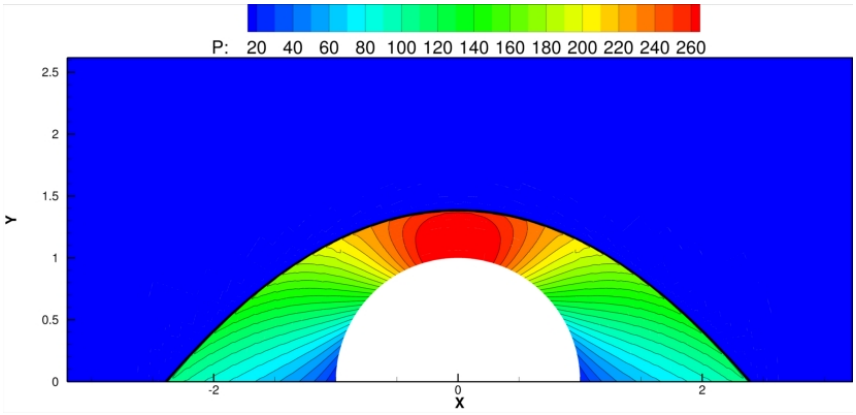
Figure 7.14: Final h-refinement grids used for computing the inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$ at $p = 2$ and $p = 3$.

7.4 Hypersonic Viscous Cylinder

The third and final test case consists of a half cylinder at hypersonic conditions. The flow is viscous and laminar with flow conditions $M_\infty = 17.605$, $Re = 376,930$ and $\alpha = 270^o$. In this case a fixed temperature wall is used for the cylinder wall boundary condition with $T_{wall} = 500K$. This case corresponds to the viscous hypersonic benchmark from reference [65]. The flow conditions are based on a low-earth orbit re-entry at 5 km/s and assuming a perfect gas model for the equation of state. This is a common hypersonic benchmark case and is used to test both the accuracy and smoothness of surface heating. The initial computational mesh containing $N = 1,711$ elements with a uniform discretization order of $p = 1$ is shown in Figure 7.19(a). Based on the results for inviscid flows in the previous sections it is now



(a) Final grid: h-refinement, $p = 2$



(b) Final grid: h-refinement, $p = 3$

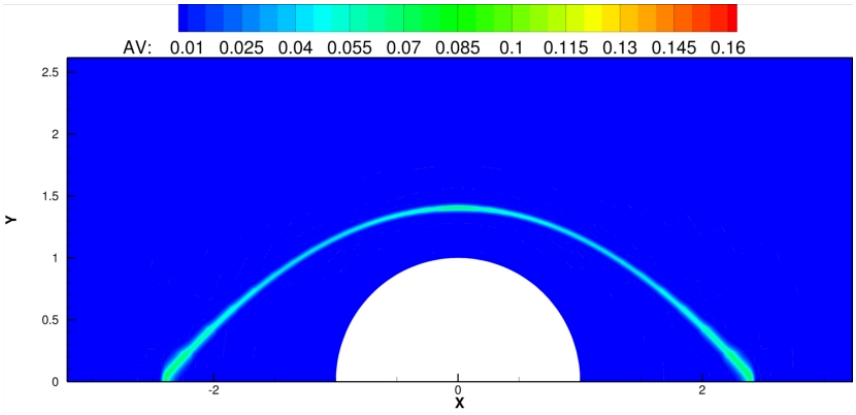
Figure 7.15: Computed pressure contours on the final for the inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$ at $p = 2$ and $p = 3$.

questionable whether high-order methods should be considered for computing high speed flows. This case demonstrates when and how high-order discretizations can be advantageous for high speed flows.

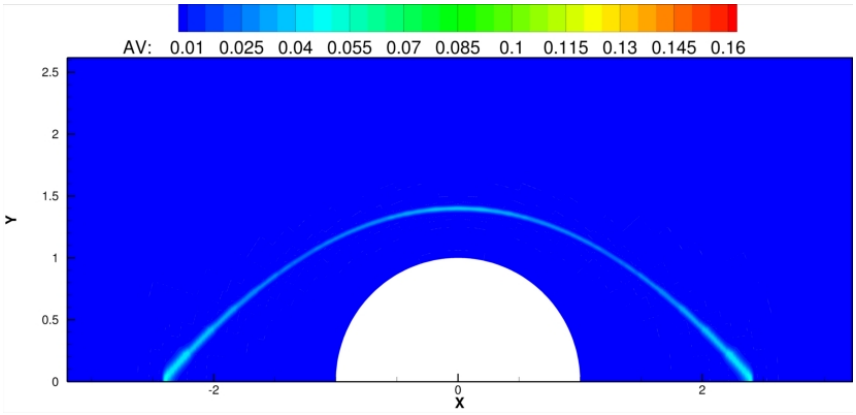
In this case the adaptation is driven via the adjoint-based error estimate in the computed surface heating (equation (7.4.1)) given in equation (5.2.10). The surface heating coefficient C_H, which is the target of the adjoint error estimate, is given by

$$\begin{aligned} C_H &= \frac{\mu_b}{P_r} \frac{\nabla T_b \cdot \vec{n}}{\frac{1}{2}\rho_b U_\infty^3} \\ U_\infty &= \sqrt{u_\infty^2 + v_\infty^2} \end{aligned} \tag{7.4.1}$$

where $P_r = .72$ is the Prandtl number, μ_b is the viscosity at the surface, T_b is the surface temperature, $\vec{n}$ is the surface normal vector, ρ_b is the surface density, u_∞ is the free-stream



(a) Final grid: h-refinement, $p = 2$



(b) Final grid: h-refinement, $p = 3$

Figure 7.16: Computed artificial viscosity contours on the final for the inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$ at $p = 2$ and $p = 3$.

u-velocity, and v_∞ is the free-stream v-velocity.

Based on the results of the previous two test cases, which have shown that p-enrichment is not as effective at computing shocked flows as h-refinement, this flow is computed with h-refinement and hp-adaptation. hp-adaptation is employed to demonstrate the advantage of using high-order discretizations in the smooth regions of the flow while simultaneously treating the shock with h-refinement. In order to draw comparisons with reference [71] the adaptation is allowed to run until the surface heating error estimate has reached a value of .01%.

The Mach number and temperature contours on the initial mesh are depicted in Figure 7.20(a) and Figure 7.20(b) respectively. In these figures the shock wave is much thicker than

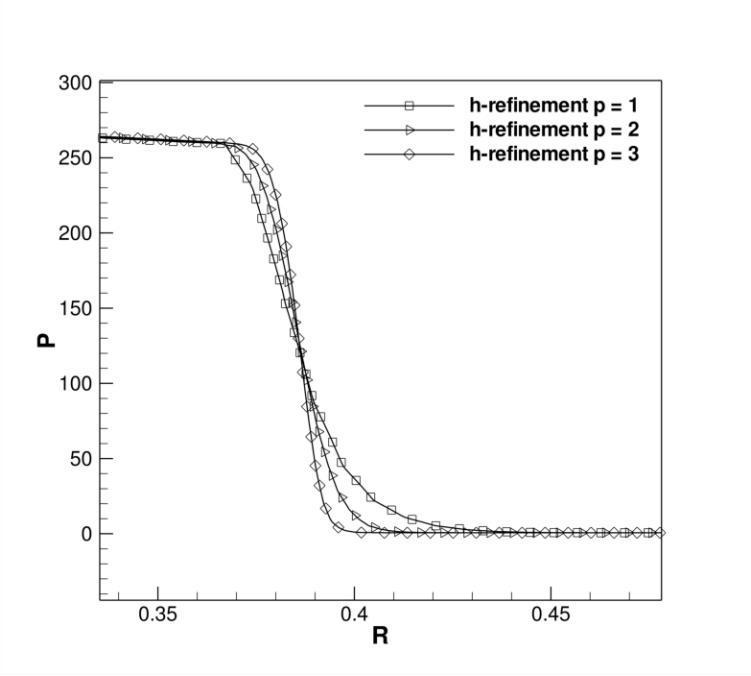
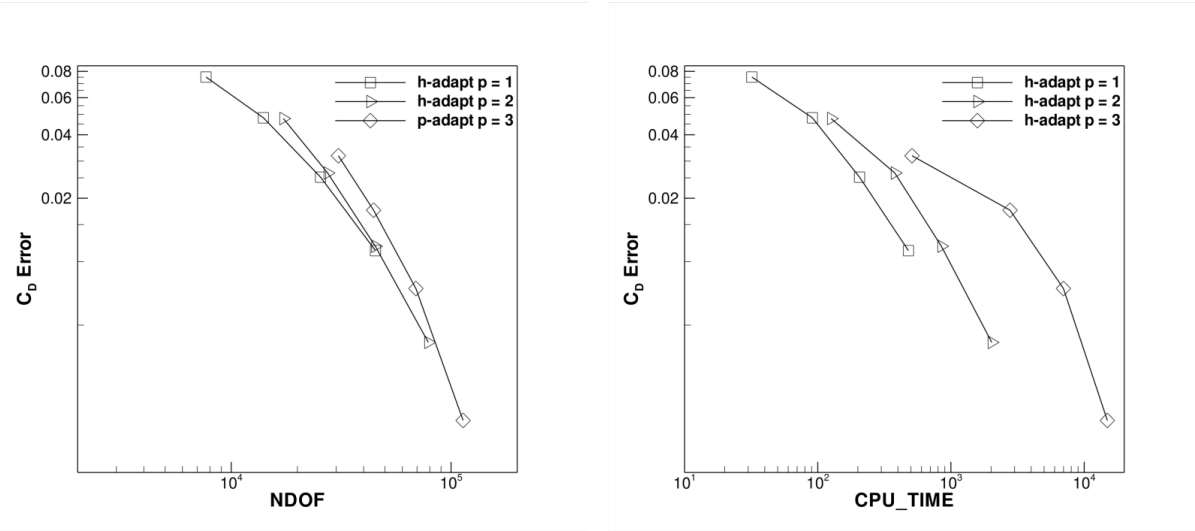


Figure 7.17: Stagnation pressure profile on the final mesh, for the inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$ at $p = 1$,$p = 2$ and $p = 3$.

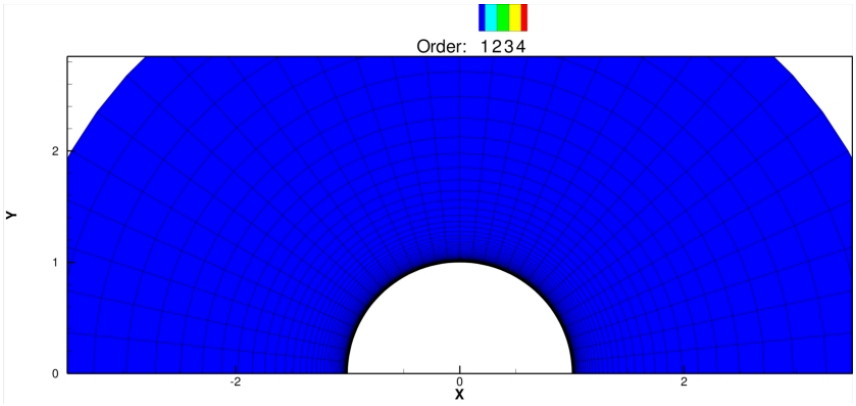


(a) Drag error versus N_{DoF}

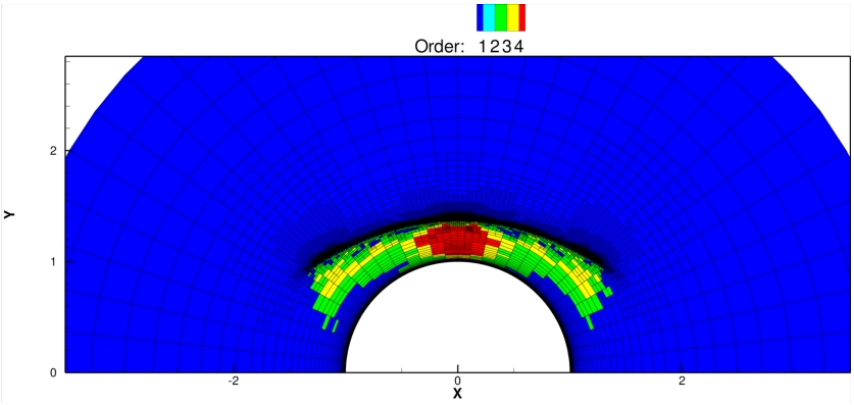
(b) Drag error versus wall time

Figure 7.18: Drag error for inviscid hypersonic flow over a half cylinder, $M_\infty = 17.605$, $\alpha = 270^o$ at $p = 2$ and $p = 3$.

in the previous inviscid case employing the same number of elements, because the initial mesh elements are clustered at the boundary to resolve the boundary layer. Therefore, the



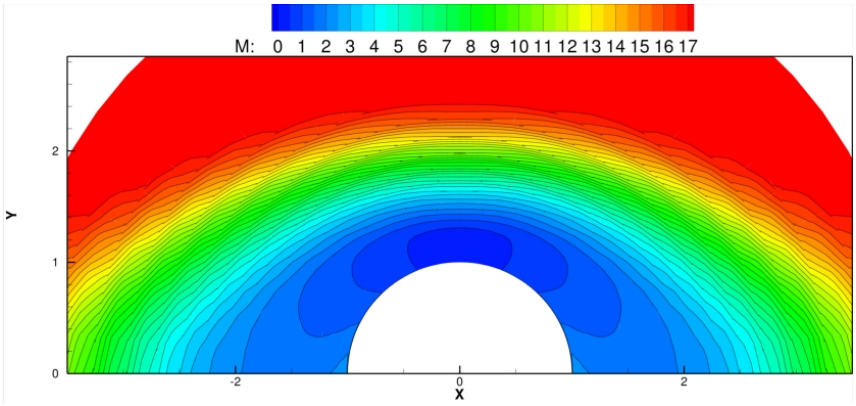
(a) Initial mesh: $N = 1711$, $p = 1$



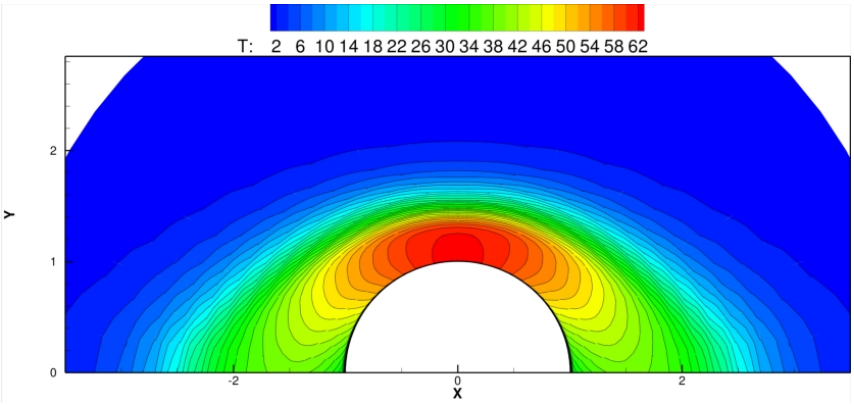
(b) Final *hp*-adapted mesh: $N = 37,575$, $p = 1$ to $p = 4$

Figure 7.19: Initial mesh and final *hp*-adapted mesh employed for viscous hypersonic flow over a half cylinder at $M_\infty = 17.605$, $\alpha = 270^o$, $Re = 376,930$.

elements in the vicinity of the are much larger in this initial mesh than in the previous inviscid test case. Six cycles of *hp*-adaptation are performed after which the surface heating error is reduced to approximately .01%. Figure 7.21(a) and Figure 7.21(b) show the Mach number and temperature contours on the final *hp*-adapted mesh respectively. These figures illustrate that output-based adaptation has been performed on this flow, since the shock wave is only well resolved over a section of the cylinder surrounding the stagnation point. A similar refinement pattern of the shock wave region is also observed in references [59, 71]. The *hp*-adaptation algorithm places high-order elements throughout the boundary layer and shock layer regions due to the smooth solution behavior in these regions. Furthermore, the surface heating is strongly dependent on these regions of the flow as well as on the portion of the shock wave targeted with *h*-refinement, which is illustrated in Figure 7.19(b). The *hp*-



(a) Mach number contours

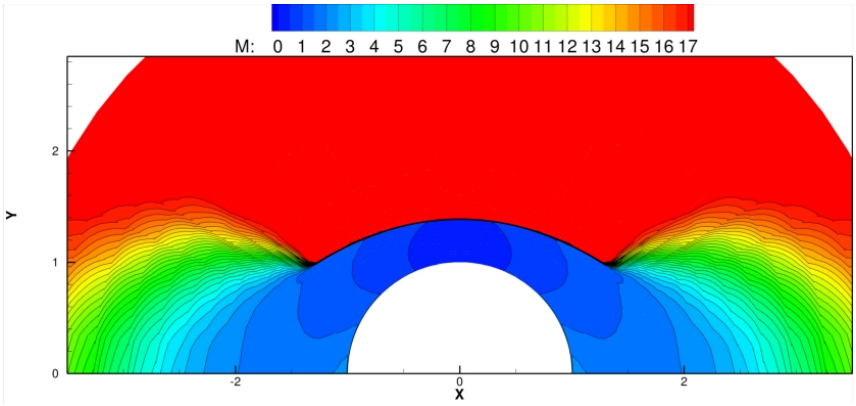


(b) Temperature $\left(\frac{T}{T_\infty}\right)$ contours

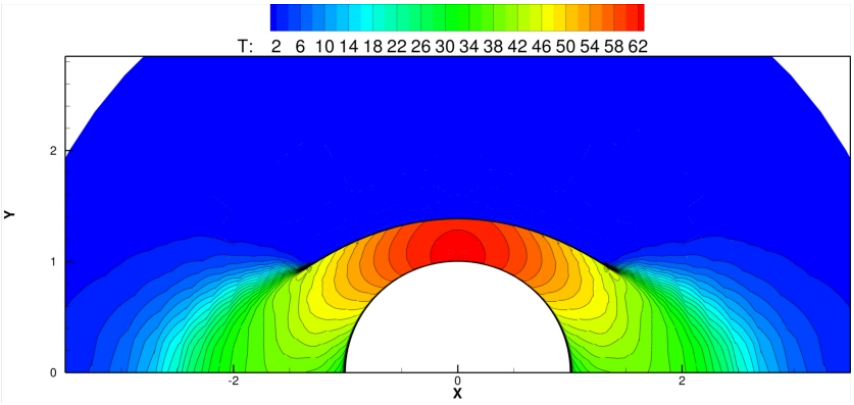
Figure 7.20: Computed Mach number and temperature contours of viscous hypersonic flow over a half cylinder on the intial grid using $N = 1,711$ with $p = 1$ at $M_\infty = 17.605$, $\alpha = 270^o$, $Re = 376,930$.

adaptation algorithm applies h-refinement in the vicinity of the shock wave while maintaining a fixed discretization order of $p = 1$.

Figure 7.22(a) shows the smooth computed surface heating profile. Smooth computed surface heating profiles are of significant concern in hypersonic simulations. While this mesh is made of quadrilaterals there is no intentional shock alignment in the mesh, in fact only at the stagnation point is there even a close alignment with the shock, which is not intentional. In this case, high-order methods obtain smooth surface heating without a shock-aligned mesh. Noisy surface heating is normally associated with entropy errors that propagate from the shock through the shock layer and it appears that high-order elements in the shock layer and a fine mesh at the shock have eliminated these errors. Reference [71] has computed this



(a) Mach number contours

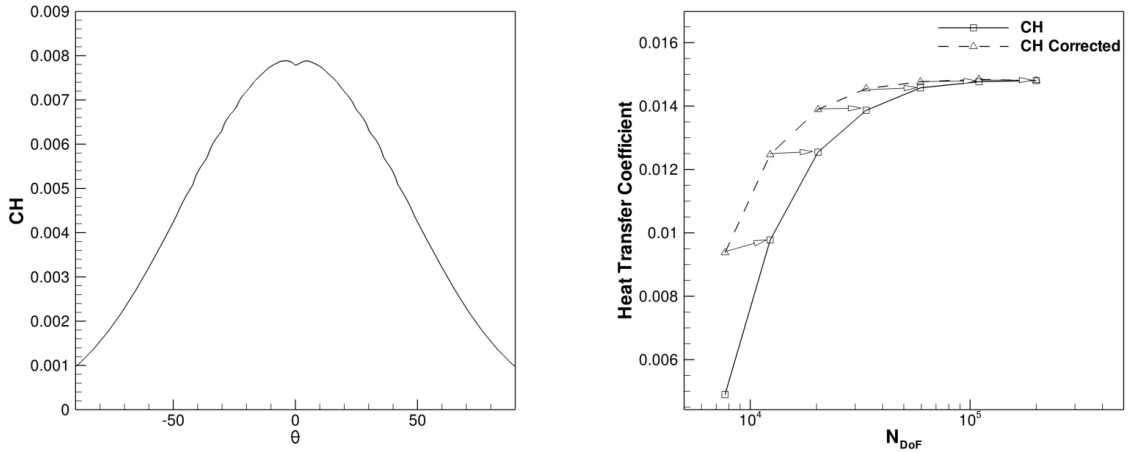


(b) Temperature $\left(\frac{T}{T_\infty}\right)$ contours

Figure 7.21: Computed Mach number and temperature contours of viscous hypersonic flow over a half cylinder on the final *hp*-adapted grid using $N = 37,575$ with $p = 1$ to $p = 4$ at $M_\infty = 17.605$, $\alpha = 270^o$, $Re = 376,930$.

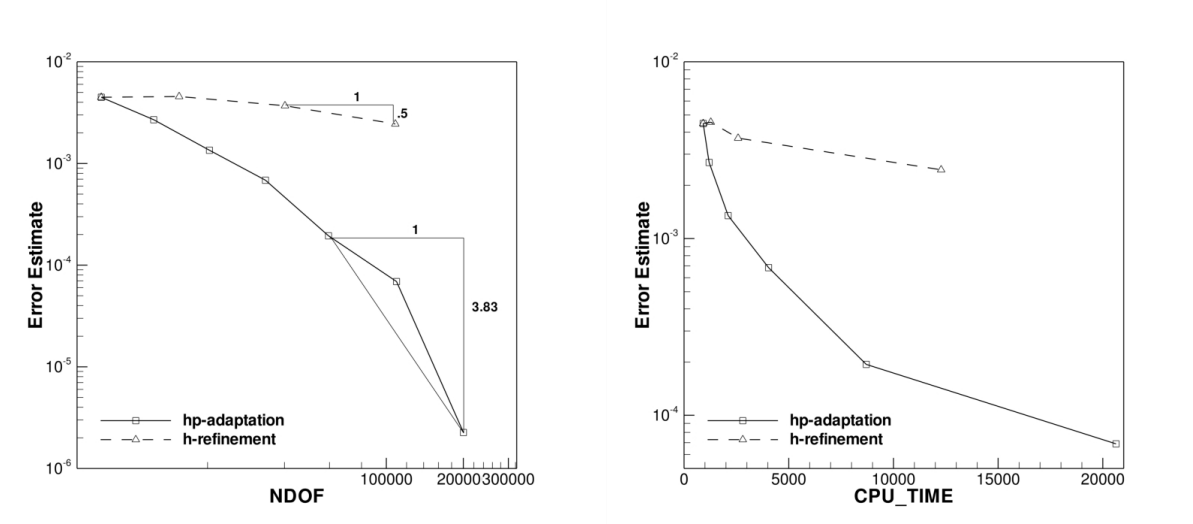
flow using output-based mesh refinement at a discretization order of $p = 2$. However, smooth surface heating was only obtained using a modified adaptation sequence(see reference [71] for details of the modified adaptation sequence), which artificially augmented the error estimate in cells close the wall. In this work no such modifications were necessary and the adaptation method is the same as in the previously presented test cases.

Figure 7.22(b) shows the surface heating and adjoint-corrected surface heating for this case. Grid converged computed surface heating vales are obtained and the computed surface heating error estimate provides accurate functional corrections throughout the adaptive procedure. The accuracy of the surface heating error estimate demonstrates another positive characteristic of artificial viscosity methods. The particular form of the error estimate used



(a) Surface heating profile on final *hp*-adapted mesh (b) Surface heating convergence history

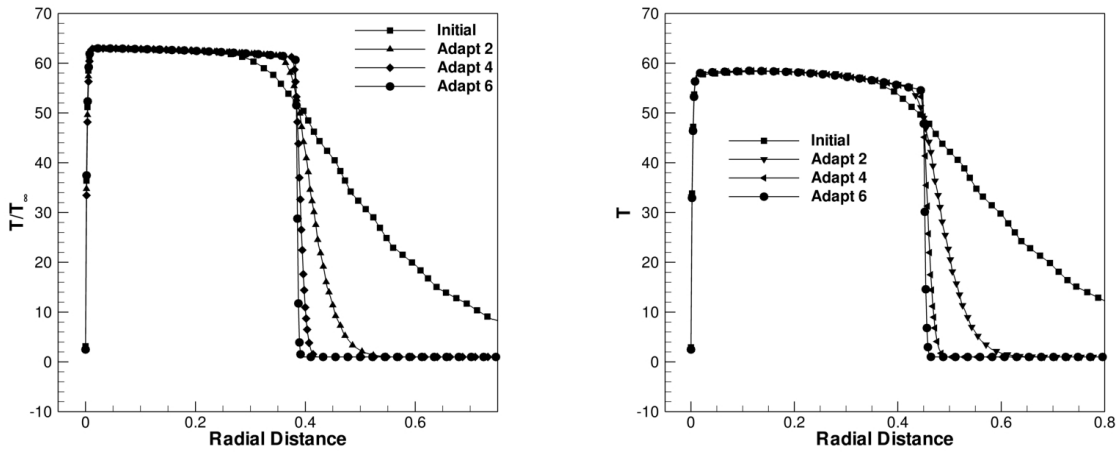
Figure 7.22: Surface heating profile on final *hp*-adapted mesh integrated surface heating convergence history using *hp*-adaptation for viscous hypersonic flow over a half cylinder at $M_\infty = 17.605$, $\alpha = 270^o$, $Re = 376,930$.



(a) Surface heating error versus N_{DoF} (b) Surface heating error versus wall clock time (sec)

Figure 7.23: Comparison of surface heating error of viscous hypersonic flow over a half cylinder using output driven *hp*-adaptation and *h*-refinement, $M_\infty = 17.605$, $\alpha = 270^o$, $Re = 376,930$.

in this work (equation (5.2.10)) can be susceptible to corruption due to Gibbs phenomena polluting the fine level residual as discussed in Chapter 5 and Chapter 6. However, the artificial viscosity adequately regularizes the fine level residual estimate, enabling correct functional error estimates.



(a) Stagnation point temperature

(b) 30^o point temperature

Figure 7.24: Temperature profiles versus radial distance at the stagnation point an 30^o circumferential point across the *hp*-adaptation for the hypersonic viscous flow over a half cylinder at $M_\infty = 17.605$, $\alpha = 270^o$, $Re = 376,930$.

Finally the computed surface heating error obtained from the *hp*-adaptation is analyzed and compared against a computed surface heating error obtained by employing *h*-refinement with a discretization order of $p = 1$. Figure 7.23(a) shows the computed surface heating error convergence versus N_{DoF}. Clearly the *hp*-refinement scheme is more effective at reducing the computed surface heating error than *h*-refinement alone. Figure 7.23(b) depicts the computed surface heating error versus wall clock time. Again the *hp*-adaptation strategy is significantly more effective at reducing the computed surface error than *h*-refinement with a discretization order of $p = 1$. While in this case *hp*-adaption is more effective in both N_{DoF} and a wall clock time sense, *hp*-adaptation is deemed more effective overall based solely on the wall clock time result, because a method is only more efficient if lower discretization error is obtained using less wall clock time. The flow and objective for this test case are strongly dependent on both smooth and non-smooth flow features, hence *hp*-adaptation is very effective because it targets each type of flow feature with the most effective form of refinement for that particular feature. *hp*-adaptation combines the lessons of this work with what is already known about high-order methods for smooth flows to achieve superior error convergence for this test case. In particular *hp*-adaptation refines the shock wave with *h*-

refinement, which this work has shown to be particularly effective compared to employing high discretization order at the shock wave. Furthermore, *hp*-adaptation targets smooth flow features with *p*-enrichment, which has proven more efficient for the error reduction of smooth flows in Chapter 5. Note that employing high discretization orders in the vicinity of the shock wave should not degrade the quality of the results. However, reference [71], which uses the same artificial viscosity method as this work, has stated that robustness problems prevented the author of this reference from employing higher than a $p = 2$ discretization during adaptive mesh refinement of this same case. The *hp*-adaptation strategy shows no such robustness issue and can employ a discretization order of $p = 4$ in the smooth regions of the domain where high-order discretizations are most effective.

Figure 7.24(a) and Figure 7.24(b) show the temperature profiles extracted radially from the surface of the cylinder along the stagnation line and along a line 30^o from the stagnation line on the cylinder respectively. These figures illustrate how the resolution proceeds over the adaptation history, which shows that the shock wave is resolved most substantially in the first four *hp*-adaptation cycles. Examination of the temperature profiles at the final *hp*-adaptation cycle shows a sharply captured shock wave and a thin boundary layer.

7.5 Summary

A quantitative assessment of refinement methods for high speed flows using high-order methods is presented. Results for shock wave dominated flows show that *h*-refinement employing a discretization order of $p = 1$ is most effective in terms of computational cost. Therefore employing high-order discretizations to capture shock waves is not necessary or recommended, given the robustness problems associated with resolving shock waves with *p*-enrichment. Application of the *hp*-adaption strategy to a standard hypersonic benchmark has shown that high-order methods can be successfully applied to high speed flow problems when smooth and non-smooth flow features are present. High-order DG discretizations show significant efficiency benefits for the computation of hypersonic viscous flows provided that the appropriate form of refinement is used. One should not expect *p*-enrichment alone to perform very

well for high speed shocked flows.

In conclusion this work has shown that the challenges faced when computing shocked flows with high-order methods do not originate from inadequate shock capturing techniques. Rather the challenges stem from choosing appropriate refinement strategies for high speed shocked flows. The currently available shock capturing techniques, especially that of reference [37] can robustly capture shock waves. The role of a shock capturing technique is to provide an extra layer of robustness to the solver. This work has shown that the accuracy and efficiency of a high-order solver applied to shocked flows is largely governed by the refinement method employed. This work has shown that *hp*-adaptation is an efficient and robust refinement method for shocked flows. Furthermore, high-order methods employing *hp*-adaptation can outperform second-order accurate methods for computing hypersonic viscous flows as shown by the final test case. For a viscous hypersonic flow *hp*-adaptation combined with the PDE-based artificial viscosity is robust and delivers highly accurate results.

Furthermore, the presented hypersonic cases contain high but smooth gradients in the shock and boundary layers(in the cases that contain these features). Yet this has not affected functional accuracy, error estimation or functional convergence in any way. The computed surface heating in the viscous hypersonic flow test case shows textbook convergence behavior and adjoint based output error estimation provides adequate error estimates for both error quantification and functional correction. The results clearly indicate that high-order DG methods are both robust and effective for this challenging case.

Chapter 8

Comparisons of Discontinuous Galerkin and Finite-Volume Methods

The final chapter considers a quantitative comparison between the presented DG solver and a second-order accurate finite-volume solver. Both solvers are unstructured CFD solvers and the finite-volume solver is an example of the current stat-of-the-art in CFD. Three comparisons of the DG and finite-volume solvers are considered. In particular, solver performance at second-order accuracy as well as uniform refinement performance are considered. Finally, the shock capturing techniques of both solvers are compared to test the monotonicity of the computed surface pressure profiles for an inviscid transonic flow. The monotonicity of the computed surface pressure has implications for computing flight envelope design spaces.

In previous chapters high-order accurate methods were shown to be more efficient than second-order accurate methods for a variety of flow problems. However, all of the previous comparisons between second and higher-order methods were conducted using the presented DG solver with a discretization order of $p = 1$. Therefore, it is of interest examine the performance of the presented DG solver compared with a second-order finite-volume solver to determine if the previous comparisons were legitimate. In order to compare DG and finite-volume methods, the comparisons are made using the presented DG solver and a finite-volume solver also written by the author. In order to make the comparisons as exact as possible, a large amount of source-code is shared between the two solvers.

This chapter addresses the comparison between DG and finite-volume methods in three ways. The first comparison addresses the second-order accurate performance of both solvers. This comparison determines if the previously presented comparisons between second and higher-order DG discretizations were adequate to determine the efficiency gains of high-order discretizations. The second comparison considers uniform refinement strategies for both solvers. Finally, the DG solver and two finite-volume solvers are compared for the prediction of a flight envelope design space problem, which tests the shock capturing techniques of these three solvers. From these comparisons, conclusions about the future of DG methods as a practical tool for scientific and engineering analysis are discussed. For all comparisons between finite-volume and DG methods, the far-field boundary location is identical between the meshes employed for both solvers. Furthermore, the far-field boundary is located 60 chords from the airfoil in all test cases.

8.1 The Finite-Volume Solver

This section outlines the finite-volume solver that is employed for the comparisons in this chapter. In order to compare the discretization methods, and not the specific implementation, the DG and finite-volume solvers share as much source code as possible. First of all, the linear algebra subroutines are identical as well as the subroutines that compute the numerical convective and viscous fluxes. Furthermore, the finite-volume solver contains most of the same features as the DG solver; including line-implicit relaxation (Section 4.3.1 and Section 4.3.3), a GMRES linear solver(Section 4.5), adaptive Newton damping via equation (4.2.4), and parallelization using MPI. All comparisons conducted in this chapter utilize the GMRES solver preconditioned by the CGS method.

While for a first-order accurate discretization finite-volume and DG methods result in identical discretizations, second-order accurate discretizations are constructed in fundamentally different ways. DG methods add additional degrees of freedom(DoFs) within the elements to obtain higher than first-order discretizations. Contrarily, finite-volume methods reconstruct interpolation polynomials using data from surrounding cells in the mesh. Thus

the data stored in the finite-volume method is always piecewise constant within each cell and the order of accuracy is determined by fitting interpolation polynomials over groups of cells. Since the order of accuracy is determined by interpolation polynomials that use data from outside the cell to derive higher than first-order accuracy, the order of accuracy and number of degrees of freedom are not coupled for finite-volume discretizations.

While a large amount of code is shared between the two solvers, the very nature of the discretization methods requires some differences. The finite-volume solver is second-order accurate based on linear reconstruction. The gradients i.e. the slopes of the linear interpolation polynomials are reconstructed using a weighted least-squares approach from reference [81]. Additionally, a slope limiter is used for shock capturing as well as general robustness enhancement. In fact, for viscous flows it has been found that employing the limiter makes start-up transients for the Newton solver less difficult to overcome. The slope limiter employed by the finite-volume solver is the smooth van Albada limiter of reference [1]. Additionally, a pressure switch [103] is used to augment the limiter, which allows for more rapid convergence of the flow equations to machine zero. In finite-volume methods the exact Jacobian is difficult to construct and often the so-called first-order Jacobian matrix is formed for use in the preconditioner. The first-order Jacobian is the exact linearization of a first-order accurate discretization. Employing the first-order Jacobian in the preconditioner can significantly degrade the convergence of the Newton solver. However, the sizes of the blocks of the Jacobian matrix are smaller for a finite-volume discretization than a DG discretization. Hence, finite-volume discretizations have an advantage in terms of cost per iteration. Since the GMRES method is employed, it might be beneficial to use the exact flow Jacobian to compute the Krylov basis vectors. As such a Jacobian-free GMRES method is implemented, in which the product of the exact Jacobian onto the Krylov basis vectors is computed in a matrix-free way [104]. Numerical experiments with a variety of flow problems have shown that this approach significantly speeds up the convergence of Newton's method for the finite-volume solver and is employed for the comparisons in this chapter. For a particularly well presented and detailed description of finite-volume methods on unstructured grids see reference [1].

The presented finite-volume solver can solve the Euler, laminar Navier-Stokes and RANS equations. This solver has been validated against exact solutions, experiment [98],and various other second-order CFD solvers [5, 66]. In this chapter the performance comparisons are relegated to simple laminar viscous flows, in order to prevent the turbulence model resolution discrepancy from affecting the results of the DG solver. The resolution discrepancy between the mean flow and turbulence model equations will not allow a fair comparison between the two solvers for turbulent flows. This is due to the fact that the finite-volume solver does not have a resolution discrepancy between the discretization of the mean flow and turbulence model equations.

8.2 Solver Performance Comparison

The performance comparison between the DG and finite-volume solvers is conducted by computing the laminar viscous flow over a NACA0012 airfoil at $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$. In order to asses the performance of the DG solver versus the finite-volume solver, two studies are conducted. The first study involves computing this flow using similar numbers of degrees of freedom at second-order accuracy. The second study involves comparing the flow solutions on a sequence of nearly uniformly refined meshes.

8.2.1 Second-order Performance Comparison

The first test considers the performance of each solver employing a second-order accurate discretization. In this case the finite-volume solver employs a mesh with $N = 14,203$ elements, resulting $14,203$ degrees of freedom (DoFs). The DG solver employs a uniform discretization order of $p = 1$ on a mesh containing $N = 4,487$ elements for a total $15,555$ DoFs. Both grids are mixed-element anisotropic grids and in general the DG grid contains higher anisotropy due to the coarse chord wise resolution of the mesh. The goal is to compare the computed drag error and wall clock time required using both DG and finite-volume methods for a fixed order of accuracy and number of unknowns. Both solvers are run in parallel on 4 cores of the same processor. Furthermore, to solve the linear system, each CFD solver employs the GM-

RES method of Section 4.5 which is preconditioned by the colored Gauss-Seidel method of Section 4.3.3. Each solver utilizes 30 Krylov vectors per Newton step with 10 preconditioning iterations per Krylov vector. In this case, the finite-volume solver employs a Jacobian-free matrix vector product [104] to build the Krylov subspace. Numerical experiments with the finite-volume solver were performed, both with and without the Jacobian-free matrix vector product, and the Jacobian-free matrix vector represents the fastest solution method for this problem. The discrete flow equations of each solver are converged 12.5 orders of magnitude. The computed drag and computed drag error results are depicted in Table 8.1 along with the wall clock time required to compute each result. The reference drag coefficient C_{DRef} is the result of computing the same flow conditions with a $p = 4$ DG discretization employing $250,000$ DoFs, which is the same reference solution used in Section 5.4.1.

Table 8.1: Comparison of finite-volume and DG solvers at second-order accuracy.

Method	Wall Time(s)	C_D	C_{DRef}	$\lvert C_D - C_{DRef} \rvert$
FV	55	.0562423	.0559061	.0003362
DG	28	.0557026	.0559061	.0002035

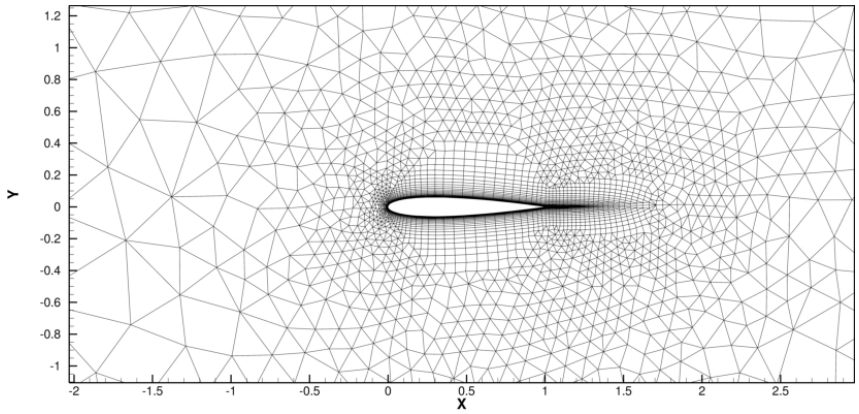
The data presented in this table shows that DG is significantly more efficient than finite-volume even at second-order accuracy. With nearly half the compute time, the DG solver employing a second-order $p = 1$ discretization attains a 39% reduction in computed drag error over the second-order finite-volume solver. This is quite a staggering result considering that the comparative situation is slanted slightly in favor of the finite-volume solver, due to the higher mesh anisotropy and additional DoFs used by the DG solver for this case. Furthermore, the results of this test case justify using the DG solver with a $p = 1$ discretization for comparison with high-order discretizations. Hence, the comparisons shown in the previous chapters(Chapter 5,7) were sufficient to judge the efficiency of high-order methods.

8.2.2 Uniform Refinement Comparison

The second comparison considers the same flow conditions and solver configuration as the comparison in Section 8.2.1. However, this comparison examines two different types of refinement for the solvers. The finite-volume solver employs nearly uniformly refined meshes

generated by the UMESH2D [80] mesh generator, which cannot generate nested meshes. However, the sequence of meshes employed are nearly uniformly refined. The DG solver employs true uniform p-enrichment, which is the uniform increase of discretization order.

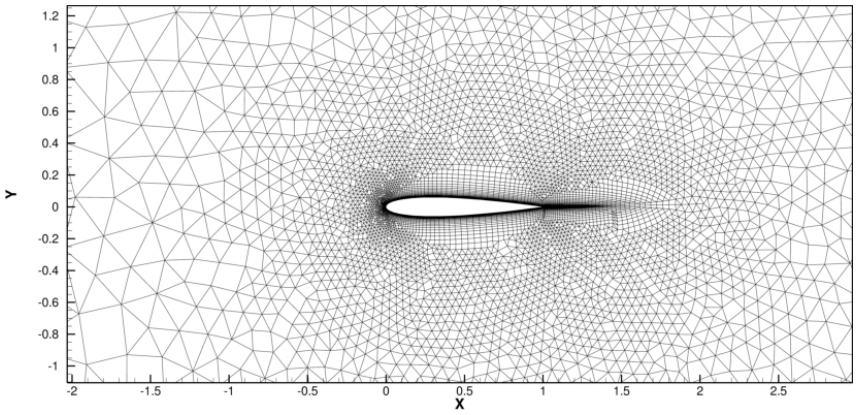
The computed drag error is compared for each solver versus both N_{DoF} and wall clock time. The finite-volume solver employs three meshes ranging from $N = 14,203$ elements to $N = 142,246$ elements. The DG solver employs a mesh containing $N = 4,487$ elements with discretization orders $p = 1$ to $p = 3$ resulting in $15,555$ DoFs to $57,434$ DoFs respectively. The mesh employed by the DG solver is shown in Figure 8.1(a) and Figure 8.2(a) through Figure 8.2(c) depict the sequence of meshes employed by the finite-volume solver. Figure



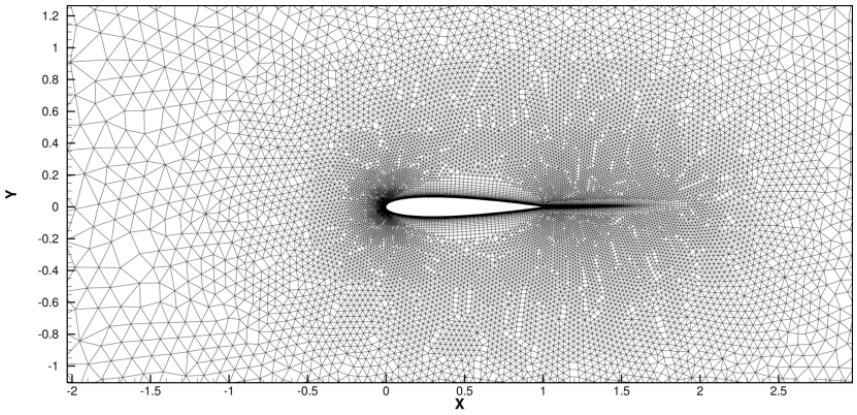
(a) DG mesh: $N = 4,487$

Figure 8.1: Mesh employed for DG solver for viscous flow over a NACA0012 airfoil.

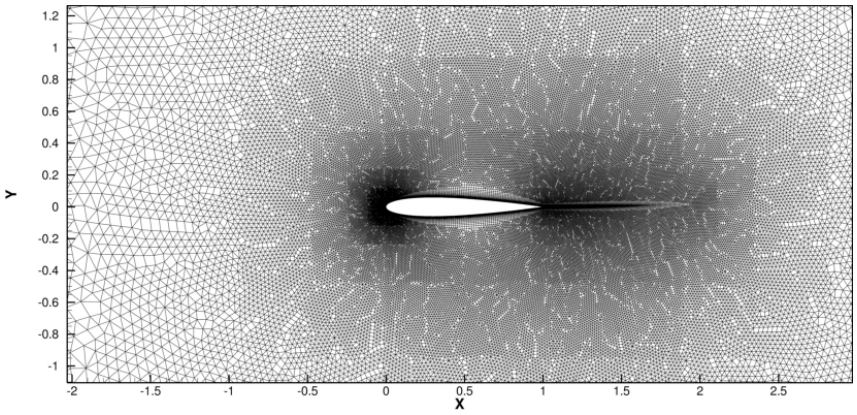
8.3(a) and Figure 8.3(b) show the computed Mach number contours using the finite-volume solver on the finest mesh and the DG solver employing a discretization order of $p = 3$ respectively. These figures demonstrate that similar solutions are obtained using both CFD solvers and that the wake is captured up to a distance of approximately four chords from the trailing edge. Figure 8.4(a) and Figure 8.4(b) show the computed streamlines at the trailing-edge separation region using the finite-volume and DG solvers respectively, indicating that the trailing edge separation is adequately captured by both solvers. Though the solutions generated using both solvers are similar, the DG solver uses one third the number of DoFs compared to finite-volume solver in this case. The wake and trailing edge separation regions both have a strong impact on the computed drag coefficient, as discussed in Section 5.4.1.



(a) Coarse mesh: $N = 14,203$

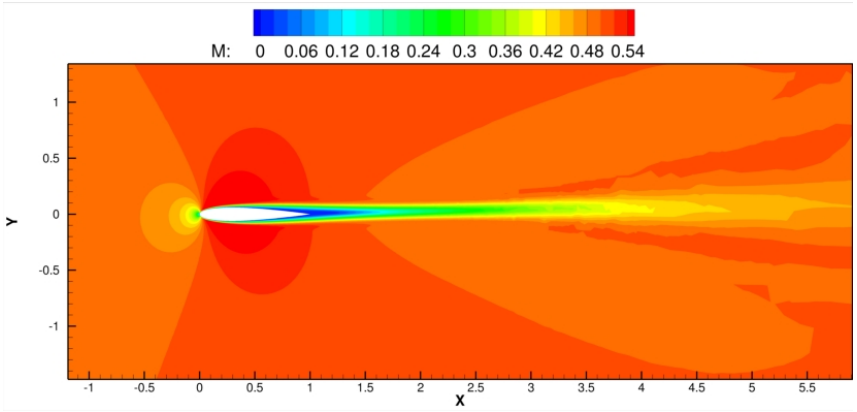


(b) Medium mesh: $N = 38,173$

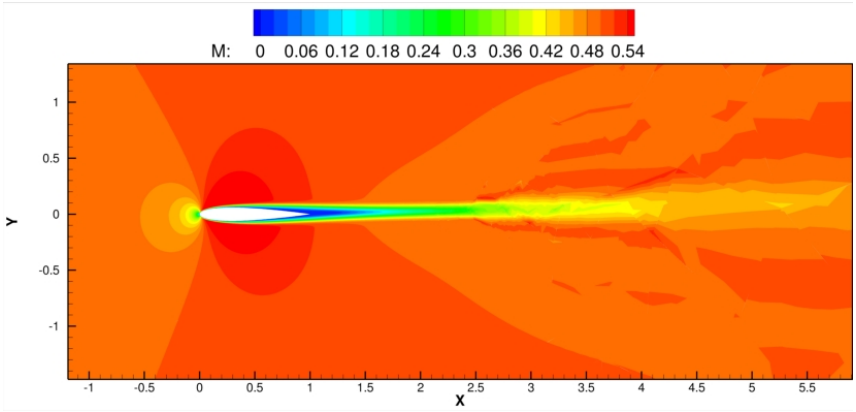


(c) Fine mesh: $N = 142,246$

Figure 8.2: Sequence of meshes employed by the finite-volume solver for viscous flow over a NACA0012 airfoil.

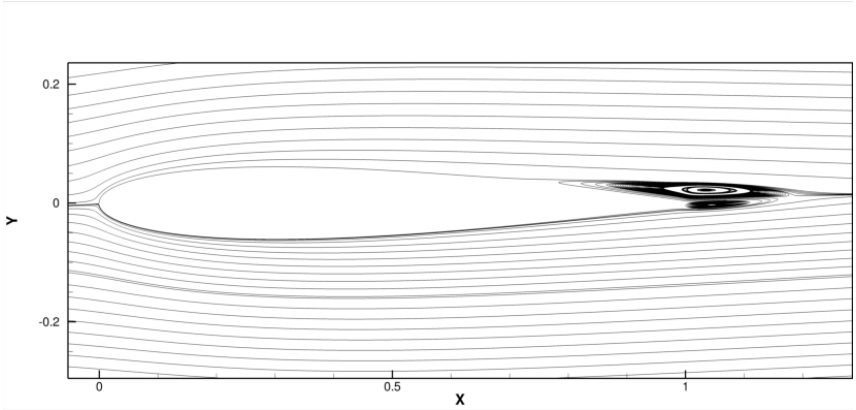


(a) Final finite-volume grid: Mach Contours

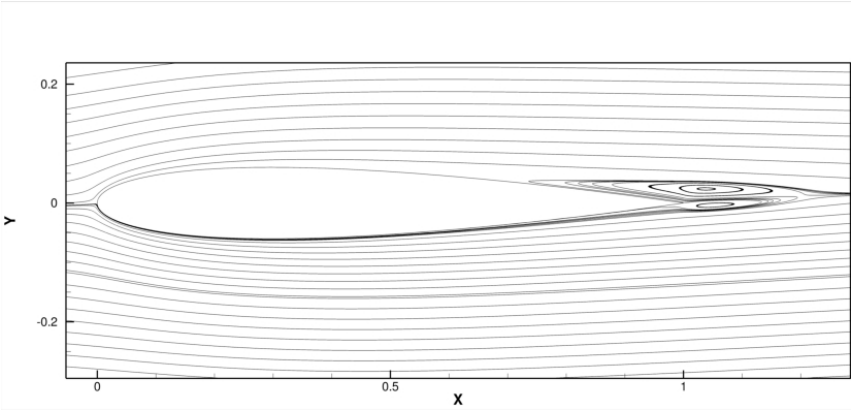


(b) DG $p = 3$: Mach Contours

Figure 8.3: Mach number contours for laminar flow over a NACA0012 airfoil, $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$ using a second-order finite-volume solver on the finest mesh of $N = 142,246$ and a DG solver with a discretization order of $p = 3$ on a mesh with $N = 4,487$ resulting in $57,434$ DoFs .



(a) Final finite-volume grid: Streamlines



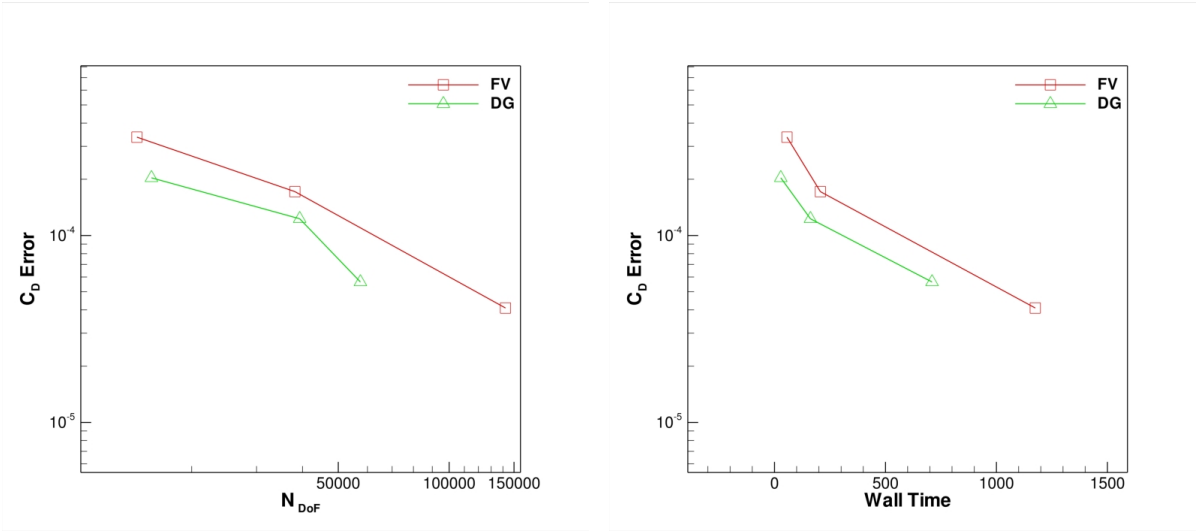
(b) DG $p = 3$: Streamlines

Figure 8.4: Close-up of trailing edge separation zone of the laminar flow over a NACA0012 airfoil, $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$ using a second-order finite-volume solver and a DG solver with a discretization order of $p = 3$.

The metric for this comparison is the computed drag error measured across the refinements. The computed drag error is determined using a reference solution, which is the result of computing the same flow conditions with a $p = 4$ DG discretization employing $250,000$ DoFs, which is the same reference solution used in Section 5.4.1. The computed drag error is given by:

$$C_D Error = \left| C_D - C_{D Ref} \right| \tag{8.2.1}$$

and is measured versus both N_{DoF} and wall clock time. Figure 8.5(a) shows the computed drag error versus N_{DoF} and Figure 8.5(b) depicts the computed drag error versus wall clock time. The results show that the DG solver produces a lower drag error throughout the



(a) Drag error vs. N_{DoF}

(b) Drag error vs. wall clock time

Figure 8.5: Comparison of drag error over refinement for laminar flow over a NACA0012 airfoil, $M_\infty = .5$, $\alpha = 1^o$, and $Re = 5,000$ using nearly uniform mesh refinement with a second-order finite-volume solver and uniform p-enrichment with a DG solver from $p = 1$ to $p = 3$.

resolution range considered, which demonstrates that the DG solver is more efficient than the finite-volume solver in terms of number of DoFs and wall clock time. Furthermore, these results indicate that it is more efficient to employ a higher-order discretization with fewer unknowns than a low-order discretization using more unknowns, based solely on the computed drag error versus wall clock time results. However, there may a point at which the discretization order should become fixed. For example, one might not want to employ $p = 10$ using only a small number of elements. The optimal discretization order p will likely

be problem dependent, based on the smoothness of the solution. Hence further study is required to determine what is the most effective p-order to use for a given error tolerance. However, for the moderate orders of accuracy considered in this comparison, high-order methods are clearly more efficient than low-order methods for these computed drag error levels.

These results show that high-order DG methods can be more efficient than second-order accurate finite-volume methods for a specified error level or a given amount of wall clock time invested. Furthermore, DG methods are flexible enough to be efficient throughout the error range, provided one does not try to use high-order discretizations for high functional error levels. Therefore, if the objective of a CFD solver is to be able to solve a wide range of problems at a wide range of error tolerances efficiently, then DG methods provide an enticing discretization method based on these results. Furthermore, additional flexibility for DG methods is offered by employing hp-adaptation, which allows for a potential increase in solver efficiency.

Lastly, attempts were made to compare the DG solver against other available viscous second-order unstructured finite-volume solvers. However, the present finite-volume solver proved to be the fastest finite-volume solver available for comparison. In summary, discontinuous Galerkin discretizations are efficient, flexible and robust discretization methods, making these methods an excellent discretization option for the next generation of CFD solver.

8.3 Design Space Considerations: Beyond Accuracy and Performance

Beyond increasing solution accuracy and improving solver efficiency, there is also a drive to examine the effects of discretization on computed design spaces. In particular, this section considers a flight envelope design space. A flight envelope design space consists of the lift or drag dependence on flight condition parameters such as free-stream Mach number M_∞ and angle of attack α. Recently there has been significant research conducted into constructing

gradient-enhanced surrogate models of aerodynamic design spaces [105–107], where the samples and the gradients at these points are obtained using a CFD solver. These surrogates are constructed for a variety of applications including optimization and uncertainty quantification. In particular, gradient-enhanced surrogate models for flight envelope design spaces are considered in references [105, 106, 108].

Surrogate models can be utilized as a means of sample reduction for uncertainty quantification and sensitivity analysis, as well as for computationally inexpensive optimization. Surrogate models are typically premised on the assumption that the design space is free of small scale oscillations or noise. This assumption is critical when considering gradient-enhanced surrogate models. Some examples of gradient-enhanced surrogates are Gaussian process regression (Kriging) [108] and polynomial chaos [109]. Excessive noise in the computed design space can impede the accuracy and validity of gradient-enhanced surrogates. This is especially true if the noise is the result of an artifact of the discretization and does not represent the physics of the underlying continuous PDEs. [110].

As an example of a design space which is artificially noisy, consider the inviscid flow over a NACA0012 airfoil at an angle of attack $\alpha = 3.5^o$ over a range of Mach numbers from $M_\infty = 0.5$ to 1.5. These flow conditions represent a one-dimensional design space where the Mach number is the design variable. The computed design spaces of lift and drag over this range of Mach number, computed with the second-order accurate finite-volume solver of reference [19], is depicted in Figure 8.6(a). Figure 8.6(a) demonstrates that the computed design space is noisy i.e. contains small scale oscillations, especially in the transonic region. Figure 8.6(b) shows a closer view of the noisiest section of the computed design space.

Motivated by this example, this work examines the source of computed design space noise. The goal is to determine if the noise in the computed design space is the result of numerical artifacts from the solver or if the noise is physical. Three CFD solvers are used to investigate the source of the design space noise. Assuming that the noise in the computed design space is not physical, it is interesting to examine if any of the three solvers is capable to preventing noise.

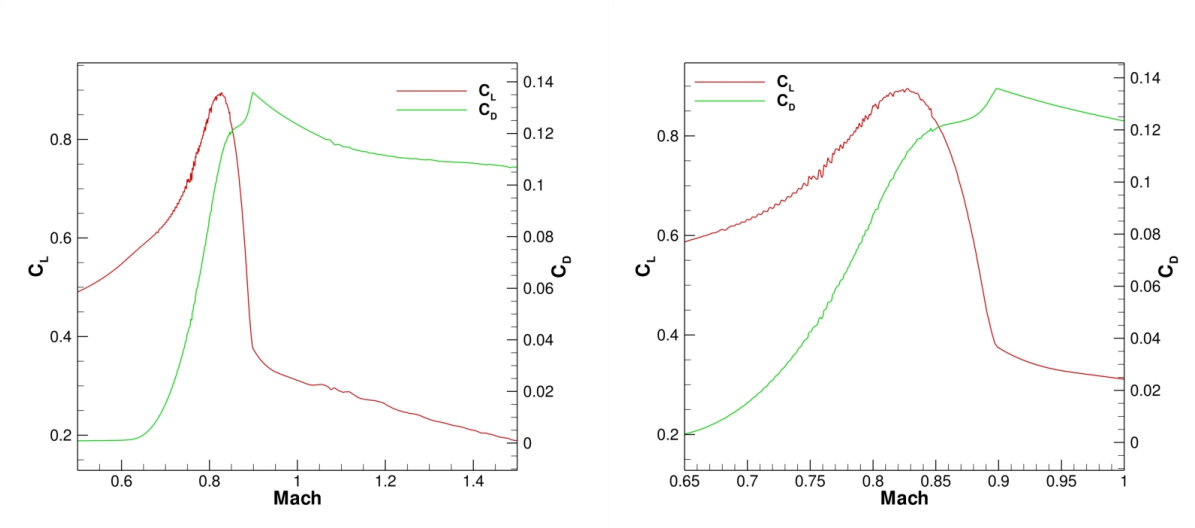
In order to limit the amount of required computations, a section of the noisiest part of

the computed design space is chosen for this comparison. The section of the design space under consideration is $0.75 \geq M_\infty \leq 0.76$), where the Mach number is sampled at intervals of .001, which necessitates 10 CFD solutions per solver. The three CFD solvers utilized for this comparison are: the presented DG and finite-volume solvers as well the finite-volume solver of reference [19]. The specified range of the design space is computed using both finite-volume solvers at first and second-order accuracy. The computational mesh employed by the present finite-volume solver is depicted in Figure 8.7(a) and contains $N = 24,090$ elements, which results in $24,090$ DoFs. The solver of reference [19] used a mesh containing approximately $N = 20,000$ elements(not shown). Additionally, the presented DG solver employs a discretization order of $p = 1$ on a mesh with $N = 7,672$ elements shown in Figure 8.7(b), resulting in $23,016$ DoFs. The PDE-based artificial viscosity described in Section 2.7.2 is used as the shock capturing technique for the DG solver. First-order computations are not performed with the DG solver since a $p = 0$ or first-order discretization is equivalent to a first-order finite-volume discretization. In this case, the results of the presented second-order finite-volume solver are generated using only the limiter and no pressure switch. Table 8.2 matches the legend markings to the CFD solvers and discretization order used to obtain the results.

Table 8.2: Listing of legend markings for the results of the computed design space of transonic inviscid flow over a NACA0012 airfoil.

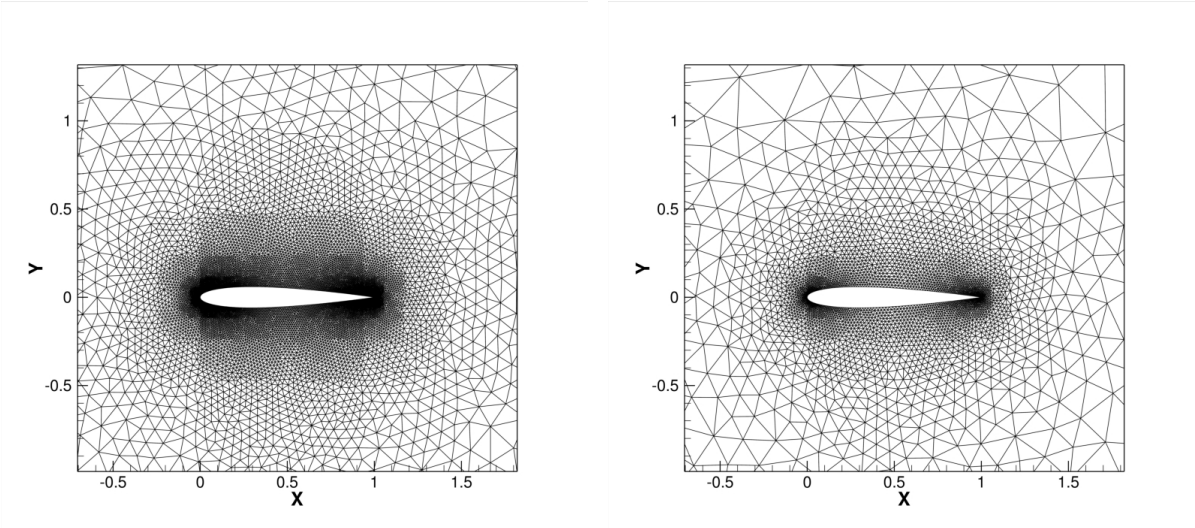
Legend Label	Solver	Discretization Order
FV 1st Order	Presented finite-volume solver	first-order
FV 2nd Order	Presented finite-volume solver	second-order
DG P = 1	Presented DG solver	second-order
FV 1st Order M. Rumpfkeil	Finite-volume solver reference [110]	first-order
FV 1st Order K. Mani	Finite-volume solver reference [19]	first-order
FV 2nd Order M. Rumpfkeil	Finite-volume solver reference [110]	second-order
FV 2nd Order K. Mani	Finite-volume solver reference [19]	second-order

Figure 8.8(a) shows the computed lift coefficient over the specified section of the design space and Figure 8.8(b) shows the computed drag coefficient. All the first-order accurate discretization results yield smooth design space behavior, while the second-order accurate finite-volume solver results exhibit noise. However, the results of the DG solver with a



(a) Full Design Space (b) Transonic Section of Design Space

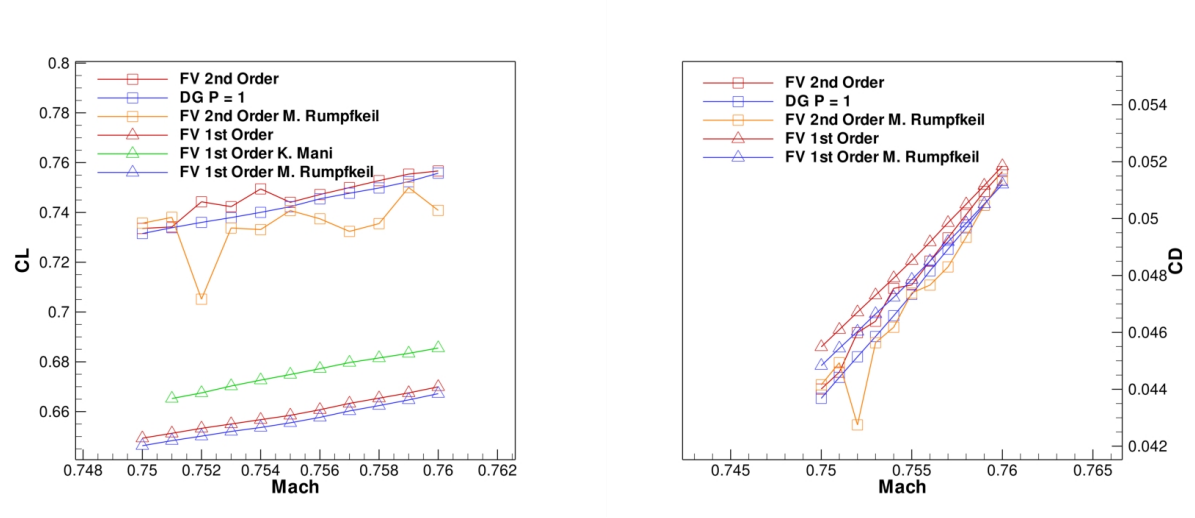
Figure 8.6: Computed design space for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$ and $M_\infty = .5$ to 1.5 using a second-order finite-volume solver.(Courtesy of M. Rumpfkeil Personal Communication.



(a) Mesh employed by finite-volume solver (b) Mesh employed by DG solver

Figure 8.7: Unstructured meshes employed for the finite-volume and DG solvers for the computation of the Mach number design space.

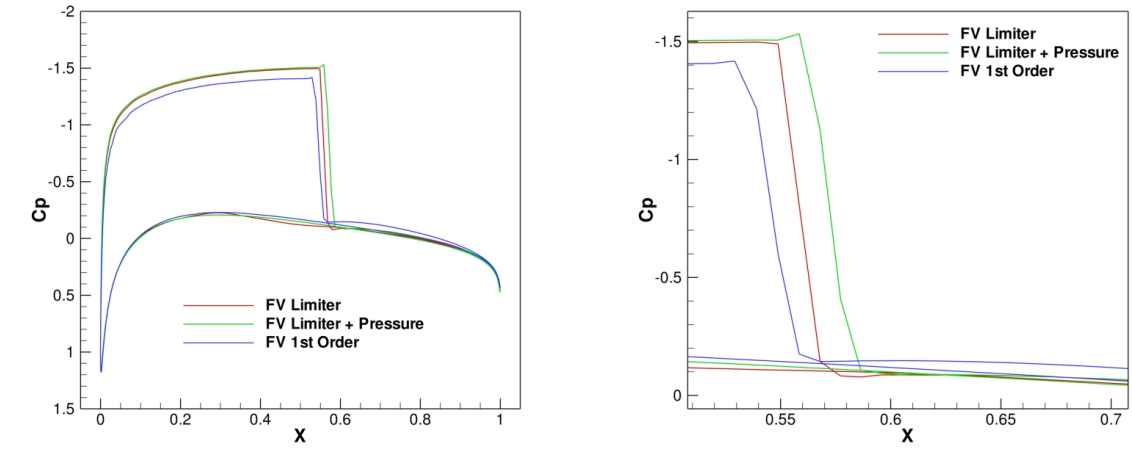
discretization order of $p = 1$ yields a smooth design space. Furthermore, these results demonstrate that some finite-volume solvers yield more noise than others. Based on these results the computed surface pressure profiles were examined which revealed non-monotone



(a) Lift coefficient

(b) Drag coefficient

Figure 8.8: Computed lift and drag design spaces for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$ using several CFD solvers $M = (.75)...(.76)$ by .001 at both first and second-order accuracy.



(a) Surface Pressure

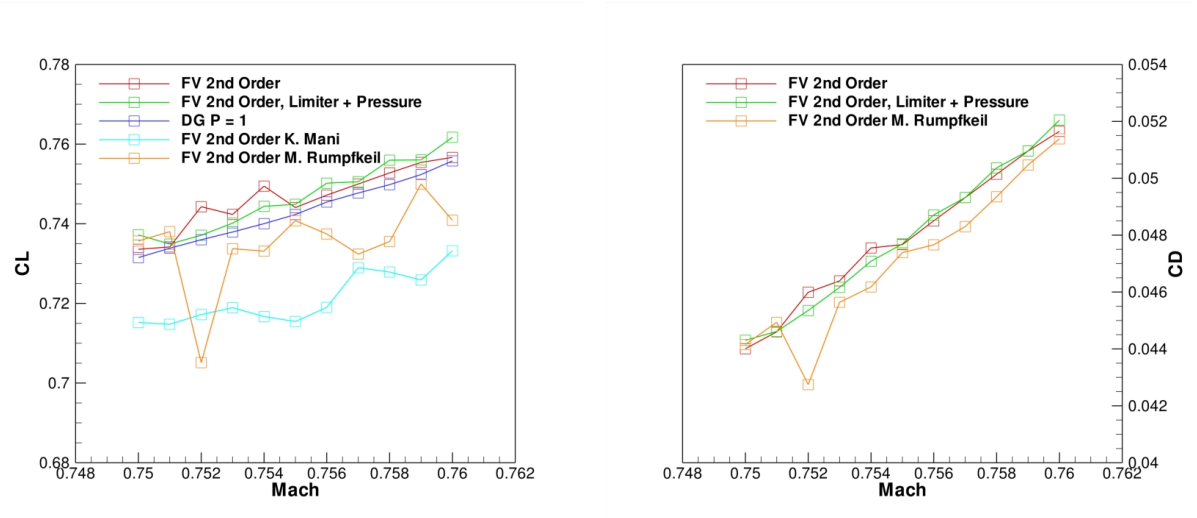
(b) Shock Close-up

Figure 8.9: Computed surface pressure for NACA0012 airfoil at $\alpha = 3.5^o$ and $M = .75$ using the finite-volume solver with two different limiter settings and a first-order discretization, limiter+pressure is the more diffusive setting.

oscillations in the computed surface pressure. Example surface pressure profiles for $M_\infty = .75$, computed using the presented finite-volume solver at first and second-order accuracy are shown in Figure 8.9.

The small non-monotone oscillations of the computed surface pressure in the vicinity of the shock wave are the cause of the design space noise. Therefore the noise in the design space is non-physical, and is the result of numerical artifacts related to the discretization, since the new surface pressure extrema at the shock wave are artifacts from the numerical solution.

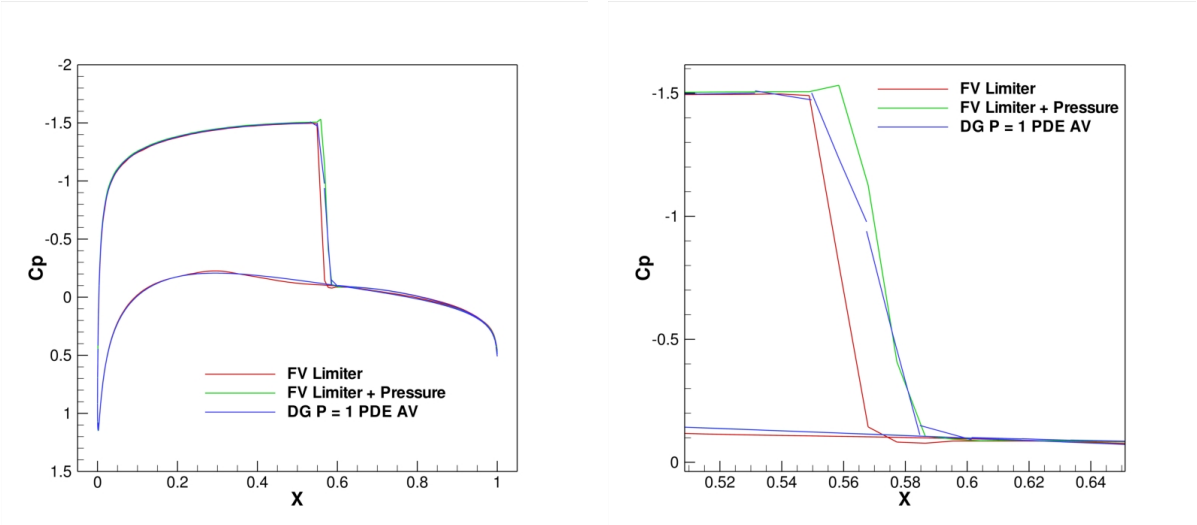
The essential problem is a lack of monotonicity of the surface pressure distribution computed using second-order finite-volume methods. In order to determine if the finite-volume solver can simply be adjusted to generate monotone surface pressure distributions, modifications were made to the presented finite-volume solver and to the solver from reference [19]. The presented finite-volume solver employs a limiter and a pressure switch to stabilize the second-order discretization in the presence of shock waves. The adjustments to the presented finite-volume solver consisted of activating the pressure switch, which was not used in the first set of computations. The solver of reference [19] is based on the matrix dissipation scheme of reference [4] and adjustments to the shock capturing scheme consisted of increasing the magnitude of the matrix dissipation coefficients. Using these modified limitation settings, additional comparisons are conducted, as shown in Figure 8.10(a) and Figure 8.10(b). Note that the results marked "K. Mani" are generated using the solver of reference [19] but with modified matrix dissipation coefficients, while the results marked "M. Rumpfkeil" use the same solver with the original coefficients [110]. The results in these figures show marked improvement but again the only truly smooth computed design space is produced by the DG solver using a discretization order of $p = 1$. The DG solver is the only solver that is able to generate a smooth computed design space, which is the result of the monotone surface pressure distribution obtained with the DG solver, as shown by the example surface pressure profiles in Figure 8.11. As a further point of comparison the design space is re-computed using the DG solver with discretization orders $p = 1$ to $p = 3$. These computations resulted in smooth computed design spaces when employing the DG solver with high-order discretizations, as demonstrated in Figure 8.12.



(a) Lift coefficient

(b) Drag coefficient

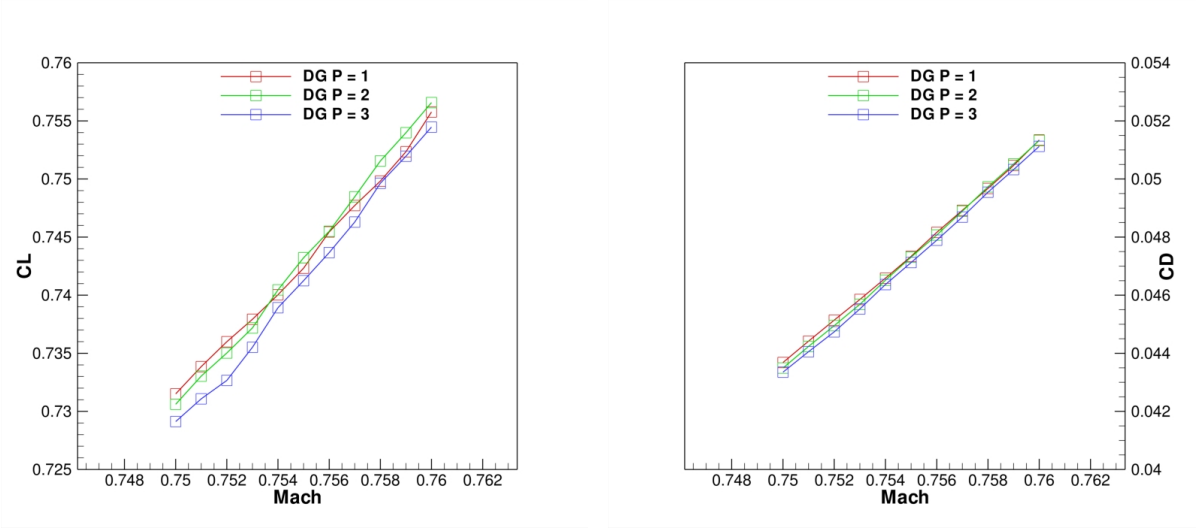
Figure 8.10: Computed design space for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$ using three CFD solvers at exclusively second-order accuracy, $M = (.75)...(.76)$ by .001. The limiter settings of the finite-volume solver were adjusted in an attempt to generate monotone surface pressure profiles.



(a) Surface pressure

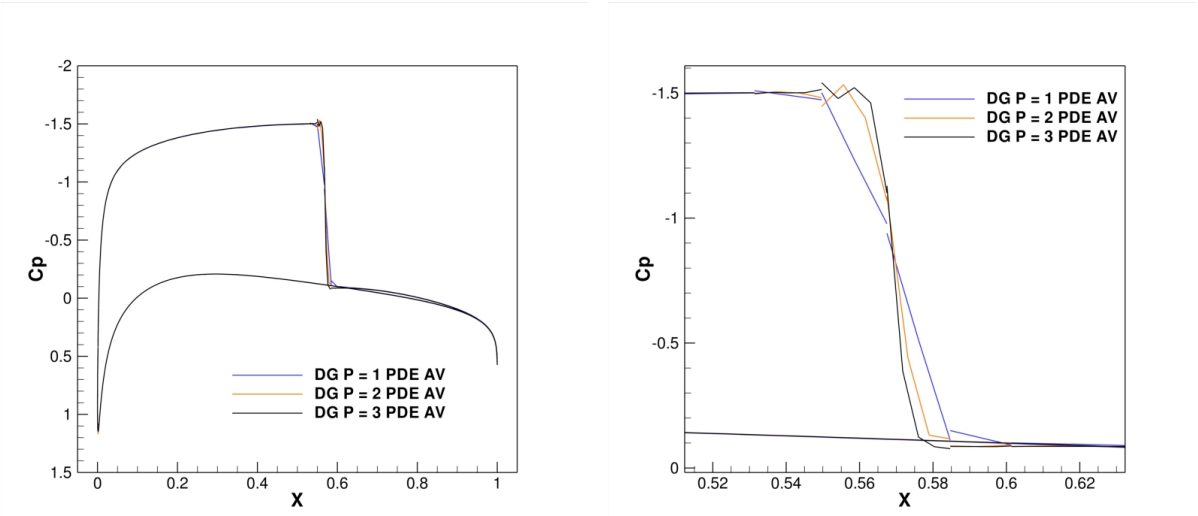
(b) Shock close-up

Figure 8.11: Computed surface pressure for NACA0012 airfoil at $\alpha = 3.5^o$ and $M = .75$ using the DG solver with a discretization order of $p = 1$ and a second-order finite-volume solver with two different limiter settings.



(a) Lift (b) Drag

Figure 8.12: Computed design space results for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$ and $M = (.75)...(.76)$ by .001 using the DG solver with discretization orders $p = 1$ to $p = 3$.



(a) Surface Pressure (b) Surface Pressure: Shock Close-up

Figure 8.13: Computed surface pressure coefficient for inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$ and $M = .75$ using the DG solver with discretization orders $p = 1$ to $p = 3$.

The DG solver is able to obtain monotone surface pressure distributions due to the use of the PDE-based artificial viscosity shock capturing technique described in Section 2.7.2. In order for a high-order DG solver to be robust, the DG solver must be able to capture shock waves using second and higher discretization orders. The coupling between DoFs and order of accuracy places an additional constraint on the shock capturing technique as discussed in Chapter 7. The more rigorous constraints on shock capturing techniques for DG discretizations, have produced a robust yet accurate shock capturing technique that maintains the accuracy of smooth solutions, by remaining inactive in regions of smooth flow and simultaneously applying artificial viscosity in the vicinity of the the shock wave. The artificial viscosity is applied in sufficient quantities to obtain monotone surface pressure profiles. The surface pressure distributions computed using the DG solver with discretization orders $p = 1$ to $p = 3$ are depicted in Figure 8.13. Since the non-monotone surface pressure distributions are not present in the DG solutions (those minor oscillations in Figure 8.13 never exceed the smooth extrema on each side of the shock wave) the resulting computed design space is smooth for all discretization orders employed. Furthermore, when employing second-order accuracy the computational cost of the DG solver is comparable to the computational cost of the finite-volume solver as shown in Figure 8.14.

The added monotonicity of the surface pressure distribution comes with the drawback of a slightly more diffused shock wave when employing a second-order or $p = 1$ discretization. However, the shock wave is not diffused enough to cause severe degradation of the computed lift and drag coefficients as seen in Figure 8.10(a) and Figure 8.10(b). Figure 8.15(a) depicts the computed Mach number contours at $M_\infty = .75$ using the second-order finite-volume solver with the most diffusive limiter settings and Figure 8.15(b) shows the computed Mach number contours using the DG solver with a discretization order of $p = 1$. The computed shock wave is spread over a larger distance in the DG solution compared to the finite-volume solution. The shock wave computed with the DG solver is thicker because the extent of the domain in which stabilization is active, is effectively larger for the DG solver than the finite-volume solver. This is demonstrated by comparing Figure 8.16(b) to Figure 8.16(a), which illustrate the extent of the regions of the domain where the respective stabilization

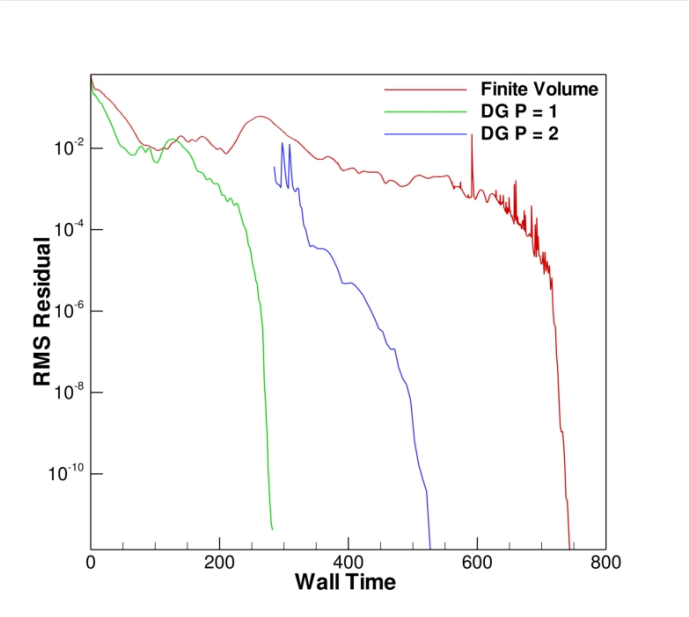
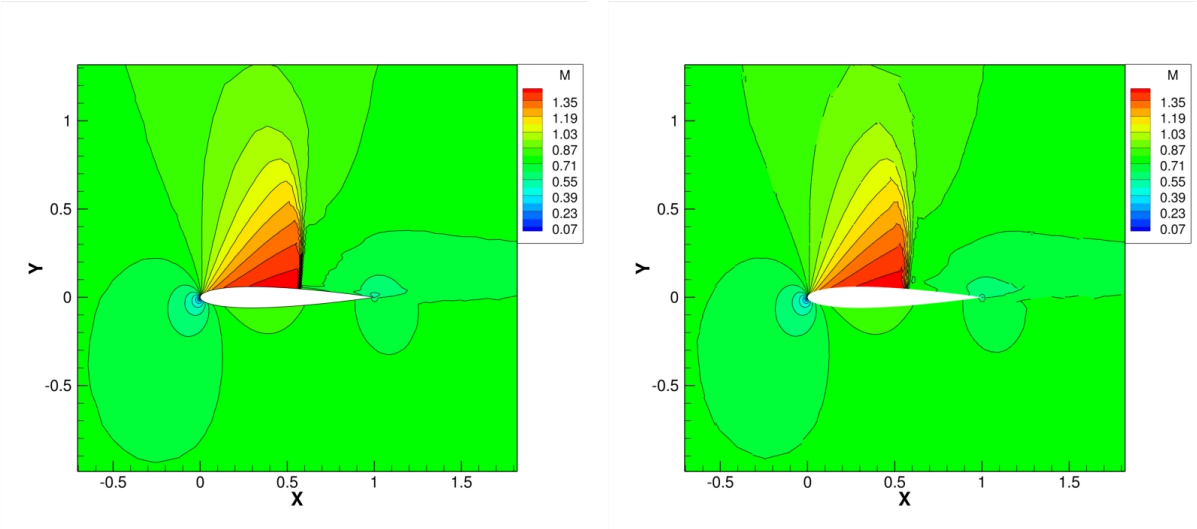


Figure 8.14: Convergence of finite-volume and DG flow solvers versus wall clock time at $\alpha = 3.5^o$ and $M_\infty = .75$.

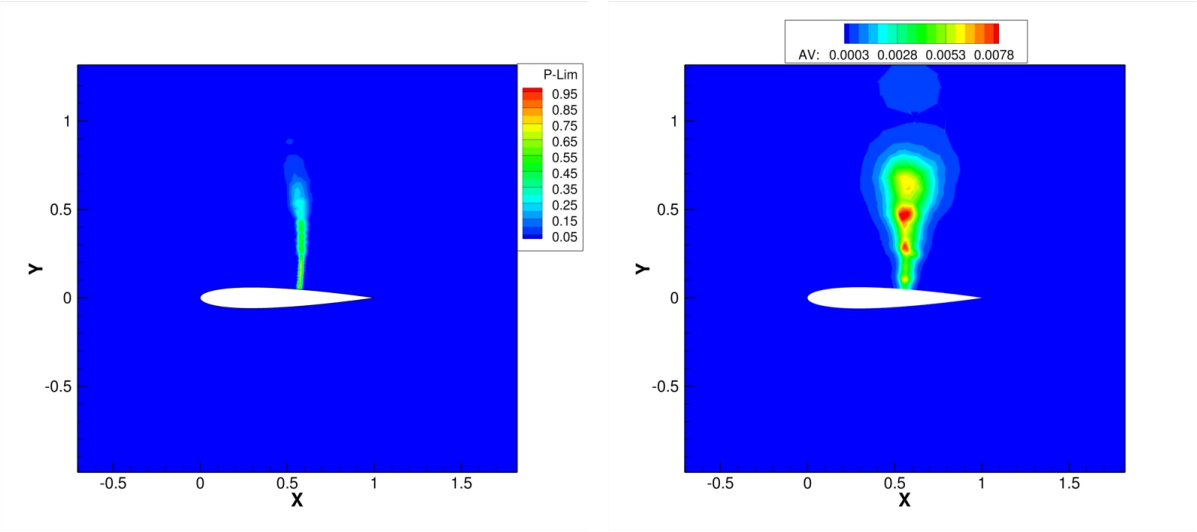


(a) finite-volume

(b) DG $p = 1$

Figure 8.15: Computed Mach number contours for inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$, $M = (.75)$ by .001 using the second-order finite-volume and DG with $p = 1$ solver.

methods are active (artificial viscosity for DG and limiter for finite-volume). The PDE-based artificial viscosity acts over a larger region than the limiter for the finite-volume solver, hence the shock wave computed with the DG solver is the thicker. However, the

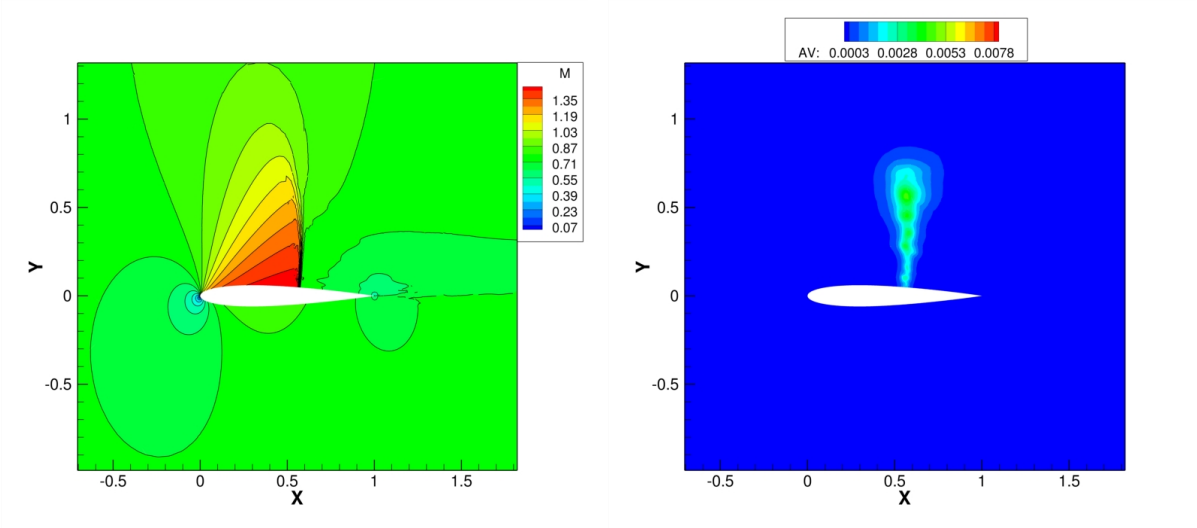


(a) finite-volume

(b) DG $p = 1$

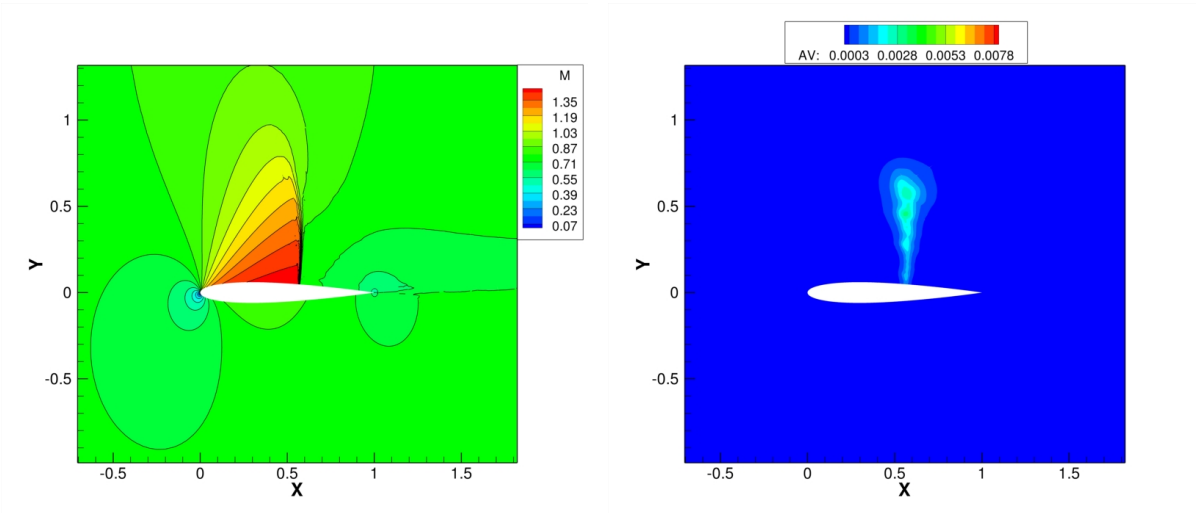
Figure 8.16: Regions of the mesh where stabilization is applied for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$, $M = (.75)$ by .001 using finite-volume and DG $p = 1$ solvers.

larger extent over which the artificial viscosity acts, results in the monotone surface pressure results produced by the DG solver. Also notice that the artificial viscosity yields monotone results without affecting the smooth flow regions in the domain. It is possible to adjust the limiter settings to obtain a monotone surface pressure profile using the finite-volume solver. However, if the limiter settings of the finite-volume solver were adjusted so that the extent over which the limiter acts is increased resulting a monotone surface pressure profile, then the limiter might become active in regions of smooth flow, which negatively impacts the accuracy of the solver. Furthermore, more diffusive limiter settings are known to degrade the iterative convergence of the solver [1]. Figure 8.17(a) and Figure 8.17(b) show the computed Mach number and artificial viscosity contours for a $p = 2$ DG discretization. While the results demonstrate that the sharper computed shock wave of the finite-volume solver is not necessarily the better result, it is possible that the DG solver requires more computational time than the finite-volume solver to compute a shock wave of this thickness. Figure 8.14 shows the computational time required to obtain a $p = 2$ solution using the DG solver. This timing result includes the computational time required to compute the $p = 1$ DG solution that initializes the $p = 2$ solution. The total computational time required to obtain the



(a) Mach $p = 2$ (b) A.V. $p = 2$

Figure 8.17: Mach number and A.V. contours for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$, $M_\infty = .75$ with the DG solver using a discretization order of $p = 2$.



(a) Mach $p = 3$ (b) A.V. $p = 3$

Figure 8.18: Mach number and A.V. contours for the inviscid flow over a NACA0012 airfoil at $\alpha = 3.5^o$, $M_\infty = .75$ with the DG solver using a discretization order of $p = 3$.

$p = 2$ DG solution is still less than the computational time required by the finite-volume solver. Furthermore, employing the DG solver with a discretization order of $p = 2$ has sharpened the computed shock wave enough that it is virtually indistinguishable from the finite-volume result without increasing the computational cost beyond what is required by

the finite-volume solver. While in practice one cycle of *hp*-adaption could have been applied to this problem, the $p = 2$ result demonstrates that the PDE-based artificial viscosity shock capturing technique is robust enough to maintain monotonicity at higher than second-order. Furthermore, the computed Mach number and artificial viscosity contours for a $p = 3$ DG discretization are shown in Figure 8.18(a) and Figure 8.18(b) respectively. These figures illustrate the reduced artificial viscosity and sharper shock wave computed with the DG solver, using a discretization order of $p = 3$ which still maintains monotone surface pressure profiles as shown in Figure 8.13.

8.4 Summary

This chapter discusses a quantitative comparison of the presented DG solver compared to a finite-volume solver. The two solvers share as much source-code as possible to facilitate an adequate comparison between the two CFD solvers. A performance test at second-order accuracy has demonstrated the increased efficiency of the DG solver compared to the finite-volume solver when employing similar resolutions at second-order accuracy. Furthermore, even at second-order accuracy the DG solver produces more accurate functionals than the finite-volume solver.

Comparison of uniform refinements between the two solvers has shown that in terms of computational time and N_{DoF}, the high-order DG solver generates more accurate functionals than the finite-volume solver. Furthermore, this comparison also demonstrated that employing a higher-order discretization and fewer elements can result in a more efficient solver, at least for the functional error levels considered in this comparison. While not immediately apparent from these results, the additional efficiency of high-order methods has significant implications for computing very large flow problems, where second-order accurate methods require very large meshes. Employing very large meshes results in significant increases in the computational costs of preforming mesh pre-processing and domain decomposition. The added flexibility of adding resolution within the elements, makes pre-processing and grid-splitting less expensive, because only the grid cells make use of these operations. For

high-order methods such as DG, the grid cells represent only a subset of the unknowns used to solve the problem and hence resolution can be increased without needing to reconsider mesh pre-processing or grid splitting.

Lastly, the presented DG solver demonstrates superior computed flight envelope design space smoothness for an inviscid transonic flow. Noisy design spaces can impede the accuracy of gradient-enhanced surrogates. For the particular noisy design space considered, the noise in the computed lift/drag coefficients versus Mach number is attributed to non-monotone behavior in the computed surface pressure distribution in the vicinity of the shock wave. This design space was computed using two different finite-volume solvers as well as the presented DG solver. For these computations several attempts were made to improve the monotonicity of the surface pressure distributions computed with the finite-volume solvers and no modifications were able to remove the surface pressure oscillations. However, the same design space was computed with the DG solver, which was able to produce a smooth design space for discretization orders $p = 1$ to $p = 3$. The robust and accurate shock capturing abilities of the PDE-based artificial viscosity method are the key component of the DG solver that results in a smooth computed design space.

Chapter 9

Conclusions and Future Work

9.1 Summary

This work has demonstrated that discontinuous Galerkin (DG) methods can be used as the basis of a robust computational fluid dynamics (CFD) solver at both low and high-orders of accuracy. The robustness of the solver is demonstrated by the application of the presented flow solver to practical two-dimensional aerodynamic problems. In the case of shocked and viscous laminar flows, the high-order DG solver delivers optimal accuracy and superior performance when compared to low-order methods. However, assessing the merits of computing turbulent flows with high-order methods has proven difficult. The difficulty is due to non-smooth behavior in the Spalart and Allmaras turbulence model working variable. This non-smooth behavior of the turbulence model working variable is not unique to DG discretizations but also affects second-order accurate finite-volume discretizations as well. Additionally, the application areas of DG discretizations have been broadened from the standard academic problems, such as isentropic vortex convection, to problems of real world interest to the aerodynamics community, such as high-lift turbulent flows and hypersonic flows.

DG methods have largely been applied to linear wave propagation problems as in references [24, 75] and others. For example, electro-magnetics and acoustics are two application areas that study linear wave propagation. There has also been a great deal of work con-

ducted in applying high-order DG methods to the inviscid Euler equations such as the work of references [23,27,33,111] and many others. The focus of this work has been high-order DG methods for non-linear Navier-Stokes problems which contain both convection and diffusion terms. It was demonstrated that DG can be as robust as low-order finite volume methods for a large variety of flow problems. Furthermore, the presented DG solver has proven to be very efficient when employing both low and high discretization orders compared to a finite-volume solver.

Several challenging application areas were considered in this work, including high speed flow applications. This work has quantitatively shown that, while from a robustness point of view a high-order solver must be able to capture shock waves using high discretization orders, capturing shock waves using this approach is not the most effective way to obtain accurate results. In terms of efficiency and robustness, it has been shown that employing low discretization order in the vicinity of the shock wave combined with mesh refinement is simultaneously more efficient and more robust. However, when considering a flow with both smooth features and shock waves, *hp*-adaptation has proven to be accurate, efficient and robust by targeting smooth features with high discretization order and shock waves with mesh refinement. Recent work [39] has advocated the use of so-called sub-cell shock wave resolution. The method of reference [34] is an example of a shock capturing technique that is capable of sub-cell shock wave resolution. This work has shown that attaining sub-cell shock wave resolution lacks robustness and is not necessarily more accurate than super-cell shock wave resolution. Therefore focus should shift from shock capturing for high-order methods to shock refinement for high-order methods. This work has presented isotropic mesh refinement as one possible refinement option. However, other mesh refinement options exist, such as anisotropic mesh movement/refinement and r-adaptation.

The subject of viscous flows leaves a somewhat less definitive conclusion. While for laminar flows DG has proven to be robust and superbly efficient at high-order accuracy, the results for turbulent flows, modeled using the RANS equation, are less encouraging. It was shown that increasing mean flow discretization order yields excellent agreement with experiment and smooth surface pressure and skin friction profiles. However, quantitative

error estimation is inaccurate due to the turbulence model working variable irregularity or non-smooth behavior. Furthermore, while this hybrid discretization has proven to be robust and allows for the possibility of functional grid convergence, mesh refinement becomes excessive due to the first-order accurate turbulence model discretization. The results demonstrated that the turbulence model discretization error has a significant impact on complex high-lift flows. Unfortunately, the current generation of turbulence models are not amenable to high-order discretization, making efficient discretization error reduction difficult. In conclusion, a new generation of turbulence models that are developed around more rigorous discretization methods such as DG will be required before demonstrable high-order accuracy will be attained for RANS flows. Furthermore, high-order DG discretizations may be more appropriately combined with Large Eddy simulations (LES) for turbulent flows.

9.2 Contributions to the Field

In summary this work has contributed the following to the field of computational fluid dynamics.

- **Non-conforming *hp*-adaptation for viscous flows**

  Although, adjoint-based refinement methods have been used in the past, this work presents the first demonstration of the application of adjoint-based refinement methods on non-conforming mixed element meshes with *hp*-adaptation for viscous flows. It was shown that this type of adaptation is very effective and does not result in a loss of accuracy for a large variety of viscous flows including airfoils and shocked flows. Furthermore, this work introduced the idea of *hp*-adaptation and appropriate refinement as both a method to increase solver efficiency and a method to increase solver robustness. This is a unique view point on *hp*-adaptation as a so called "bottom-up" limiting approach.

- **Error estimation under solution irregularity**

  This work made extensive use of adjoint based-error estimates and a great deal of effort

was focused on obtaining error estimates that were accurate and provided adequate functional corrections. The dual consistency of the artificial diffusion operator was discussed at length and constraints on the form of the artificial viscosity were obtained based on the dual consistency analysis. For shocked flows it was demonstrated that artificial viscosity was able to sufficiently regularize the solution so that functional error estimates are not corrupted by Gibbs phenomena in the fine level residual estimate. Many authors [20,45] get around the irregular residual issue by using the so-called dual error estimate (ϵ_a in equation (5.2.10)). This strategy has been avoided because this error estimate is not a true representation of functional error, instead it is a measure of duality error. It was desired that the adaptive refinement strategy target functional error directly and controlling the fine level residual estimate is a more direct way to achieve accurate error estimates and drive adaptive mesh refinement.

- **Critical study of high-order shock capturing**

  Shock capturing using high-order discretizations was discussed at length in this work. Although, there were not any new shock capturing methods presented, an even more important conclusions were drawn. While adequate techniques for capturing shock waves with high-order discretizations already exist, this work presents the first quantitative analysis of the accuracy and robustness of high-order shock capturing. It was found that employing the appropriate refinement strategy is the key to obtaining accurate results for flows with shock waves. Furthermore, it was shown that attaining sub-cell shock wave resolution is not necessarily optimal due to robustness issues. Additionally, it was shown that the shock capturing method of reference [34], which is able to obtain sub-cell shock wave resolution, did not produced grid converged functional outputs as the discretization order was increased. Hence sub-cell shock wave resolution is not necessarily more accurate than spreading shock waves over one or more elements. Additionally, this work contains the first adaptive computation of a hypersonic viscous flow above third-order accuracy and is the first to demonstrate optimal high-order convergence rates of surface heating error for a viscous hypersonic flow. This work also contains the first application of adjoint-based *hp*-adaptation to a viscous hypersonic

flow, demonstrating the accuracy, efficiency and robustness of this approach for this challenging test problem.

- **Turbulence modeling and high-order methods**

  Turbulent flows also made up a large portion of the presented work, and while the results were not optimal they were at least able to demonstrate that the DG solver is robust for the mean flow equations. In fact, one could say that DG methods are as or more robust than finite-volume methods, since no limiter or artificial diffusion was used in any of the turbulent flow computations. The subject of turbulence model non-smooth behavior was discussed at length as was the impact of this non-smooth behavior on error estimation and solver robustness. It was found that a hybrid discretization using a first-order accurate finite-volume discretization for the turbulence model equation and high-order DG discretization for the mean flow equations represents a viable option for a robust DG based turbulent flow solver. The robustness improvement was sufficient to allow for the first DG simulation of the 30P30N high-lift multi-element airfoil configuration, which is a milestone in making DG discretizations the basis of a practical aerodynamic analysis tool.

In summary, this work has shown that DG methods can be made robust and efficient for CFD applications. Furthermore, enhancing the robustness of DG methods has not adversely impacted the accuracy for most of the examples presented in this work. The robustness of the present DG solver is demonstrated by its application to large variety of challenging flow problems.

9.3 Future Work

The results of this thesis are very encouraging and show promise for applying high-order DG discretizations to enhance CFD simulation fidelity in a variety of application areas. Some potential future work directions are outlined below:

- **Three-dimensional problems**

Since the recipe for a robust high-order accurate DG discretization based flow solver is now in hand, the next area of future work will be to write a three-dimensional version of the current solver. Due to the difficulties encountered with RANS turbulence models, one is tempted to consider studying the applicability of high-order methods to Large Eddy Simulation(LES), which requires a three-dimensional solver in order to adequately capture the turbulent physics. However, there are significant challenges associated with a three-dimensional solver. At present the two-dimensional solver stores the full flux Jacobian matrix which contains large dense blocks. The block size scales as $(N_f M)^{2d}$ where d is number of spatial dimensions, N_f is the number PDEs in the system, and M is the number degrees of freedom within each element. Increasing d from 2 to 3 makes the matrices large and very expensive to factorize. Furthermore, the number of unknowns and quadrature cost scale poorly in three-dimensions as well. However, for hexahedral elements an efficient tensor product nodal basis set can be used. In this case, the residual scaling cost goes from M^d to dM which is a significant improvement. Efficient basis sets on non-hexahedral elements will be the subject of future research, in order to develop the most efficient three-dimensional implementation possible.

- **LES and hybrid RANS/LES formulations**

  Since the non-smooth behavior of RANS turbulence models causes such difficulty for computing turbulent flows, LES becomes an attractive alternative. Often times there is no model PDE to be solved for LES, which can make turbulence model non-smooth behavior a non-issue, at least for preliminary testing of LES and high-order methods. Eventually new turbulence models will need to be developed, such that the turbulence model is free of non-smooth behavior and appropriate for use with next generation numerical methods. This new turbulence model will probably take the form of a hybrid RANS/LES model to facilitate very high Reynolds number applications $Re \geq 10^6$. This area of future work is obviously very closely tied with the development of a three-dimensional solver. As far as LES is concerned, a possible immediate application of high-order methods for LES flows employs the so called implicit LES method (ILES),

which uses no turbulence model. A sample two-dimensional ILES calculation is shown in Figure 9.1, which shows that high-order DG discretizations can capture multiple scales of the flow.

- **Rotorcraft and Wind Energy**

  Two of the most challenging aerospace CFD problems currently under consideration are rotorcraft and wind turbine aerodynamics. These application areas are true multi-physics problems because fluids, structures, and dynamics are all inherently coupled. These types of flows also require very high resolution, and adaptive refinement has been shown to be beneficial for the off-body wake capturing [112]. Future work will consist of using DG for these types of computations especially for the off-body wake capturing, where currently hundreds of millions of unknowns are used to capture the wakes from these complex applications. Additionally, these applications are an example where current turbulence models do not perform adequately. These application areas provide motivating problems for the investigation of the relationship between discretization error and turbulence modeling

- **Real Gas Hypersonic Flows**

  Based on the initial success with computing perfect gas hypersonic flows presented in this work, the extension to real gas hypersonic flows is a another area of future research. In hypersonic flows the temperature is sufficiently high, that gases dissociate and react with one another in the shock layer. The extension of the present method to include these effects represents another area of potential research. Real gas flows involve solving scalar transport equations for the various species that are generated by the chemical reactions, which results in significantly more equations to solve. However the reaction zones add additional smooth phenomena that can benefit from higher-order methods.

- **Time-accurate *hp*-adaptation**

  An avenue of additional fundamental research consists of extending the steady-state *hp*-adaptation method to a time-accurate *hp*-adaptation method, allowing for refining

and de-refining in time as necessary. When considering time-accurate flows one should also consider temporal refinement and variable algebraic error levels. These sources of error have been considered in a finite-volume setting in reference [19]. Temporal refinement for high-order DG methods has also been considered in [59]. Future work will build on the work of these references to develop an adjoint-based *hp*-adaptation strategy for time-accurate viscous flows. The *hp*-adaptation strategy will include: spatial *hp*-adaptation in a time-accurate setting as well as temporal *hp*-adaptation within the same framework. This type of approach should provide significant efficiency and robustness for computing time-accurate viscous flows and LES computations.

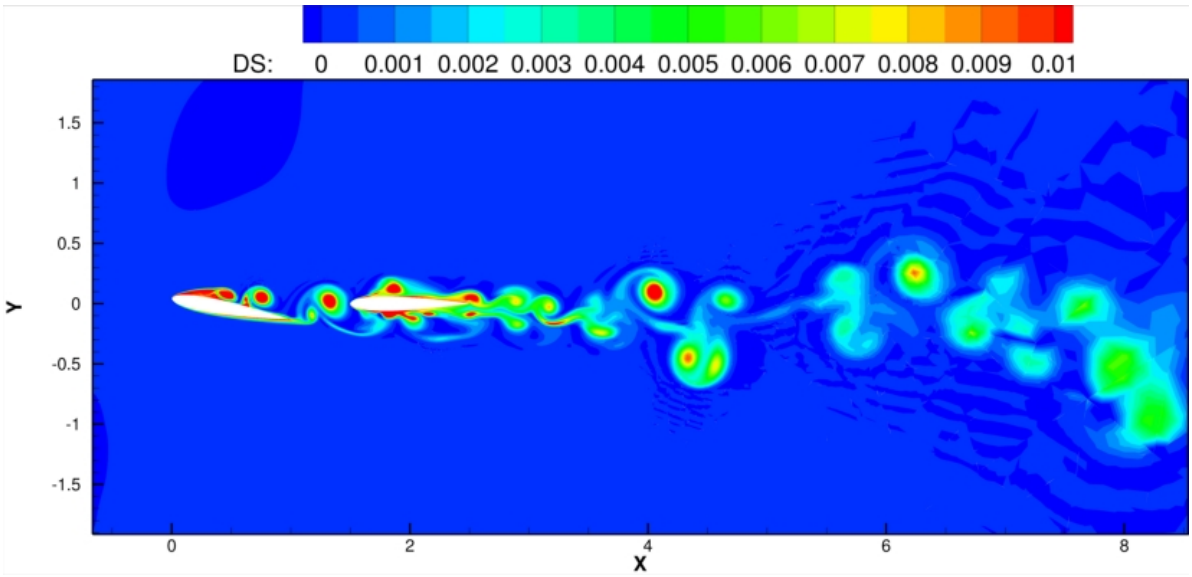


Figure 9.1: Entropy contours for the ILES flow over tandem NACA0012 airfoils using a DG discretization of order $p = 4$, $M = .2$, $Re = 10,000$

Appendix A

Derivation of the Symmetric Interior Penalty (SIP) Method

This appendix gives a detailed derivation of the symmetric interior penalty method. The derivation illustrates the origin of the penalty and symmetry terms which are often deemed ad-hoc. In particular, it is shown that the symmetry term is a necessary part of the diffusion discretization.

A.1 Model Problem

Herein the Symmetric Interior Penalty(SIP) discontinuous Galerkin(DG) method for diffusion problems is derived. As a model problem consider a Poisson equation with a non constant diffusion coefficient given by:

$$\nabla \cdot (D(x,u)\nabla u) = -f \quad \vec{x} \in \Omega \tag{A.1.1}$$

where $D(x,u)$ is the diffusion coefficient. For a system of equations such as the Navier-Stokes equations things become a bit more complicated to write down, however the basic formula is the same. In fact, the viscous fluxes of the Navier stokes equations are the same as the model equation, with the exception that $D(x,u)$ becomes a block matrix that is only a function of u.

A.1.1 Mixed Finite-Element Method

Let the domain Ω be divided in N non-overlapping elements denoted as $\mathcal{T}_h$ such that

$$\Omega = \bigcup_{k=1}^{N} \Omega_k \quad k \in \mathcal{T}_h \tag{A.1.2}$$

On each element in $\mathcal{T}_h$ define the following discontinuous function spaces, in which solutions and gradients reside.

$$\begin{aligned} \mathcal{V}_h^p &:= \left\{ v_h \in L^2(\Omega_h) : v|_e \in P^n(k) \, \forall k \in \mathcal{T}_h \right\} \\ \vec{\mathcal{G}}_h^p &:= \left\{ \vec{g}_h \in \left[L^2(\Omega_h) \right]^d : \vec{g}|_e \in P^n(k) \, \forall k \in \mathcal{T}_h \right\} \end{aligned} \tag{A.1.3}$$

The discretization is given by introducing a test function $v_h \in \mathcal{V}_p^h$ and taking the inner product of this test function and equation (A.1.1). The DG discretization of equation A.1.1 in the domain Ω is given by: find $u_h \in V_h^p$ such that

$$\sum_{k \in \mathcal{T}_h} \int_{\Omega_k} v_h \nabla \cdot (D(x, u_h) \nabla u_h) \, d\Omega_k = - \sum_{k \in \mathcal{T}_h} \int_{\Omega_k} v_h f \, d\Omega_k \quad \forall v_h \in \mathcal{V}_h^p \tag{A.1.4}$$

It is not straight forward to approximate the diffusion operator in a discontinuous function space such as $\mathcal{V}_h^p$. However, it is clear how to discretize the advection operator in a discontinuous space and thus an auxiliary variable for the gradient will be introduced, which effectively recasts the diffusion equation in equation (A.1.1) as two advection equations. While it is possible solve for the auxiliary variable and use it to compute the viscous fluxes, doing so is not an optimal method, since this method turns one equation into two equations. Therefore, the auxiliary variable is introduced solely as a method of manipulating the equations and is never explicitly computed in the SIP method. However, significant insight into how the SIP method is derived and how SIP compares to other DG diffusion discretizations is gained by considering the auxiliary variable formulation introduced above. Introduction of the auxiliary variable $\vec{z} = \nabla u$ gives the finite-element problem as: find $u_h \in \mathcal{V}_h^p$ and $\vec{z}_h \in \vec{\mathcal{G}}_h^p$ such that

$$\begin{aligned} \sum_{k \in \mathcal{T}_h} \int_{\Omega_k} v_h \nabla \cdot (D(x, u_h) \vec{z}_h) \, d\Omega_k &= - \sum_{k \in \mathcal{T}_h} \int_{\Omega_k} v_h f \, d\Omega_k \quad \forall v_h \in \mathcal{V}_h^p \\ \sum_{k \in \mathcal{T}_h} \int_{\Omega_k} \vec{g}_h \cdot \vec{z}_h d\Omega_k &= \sum_{k \in \mathcal{T}_h} \int_{\Omega_k} \vec{g}_h \cdot \nabla u_h d\Omega_k \quad \forall \vec{g}_h \in \vec{\mathcal{G}}_h^p \end{aligned} \tag{A.1.5}$$

where the $\widehat{(\cdot)}$ denotes a numerical flux of the given quantity. Integration by parts results in a weak form of the governing equations, which is used to derive a fully discrete system. The weak form is given by:

$$\begin{aligned}
&\sum_{k\in\mathcal{T}_h}\left(-\int_{\Omega_k}\nabla v_h\cdot(D(x,u_h)\vec{z})\,dx+\oint_{\partial\Omega_k}v_h(\widehat{D(x,u_h)\vec{z}})\cdot\vec{n}ds\right)=\\
&-\sum_{k\in\mathcal{T}_h}\int_{\Omega}v_h f dx\quad\forall v_h\in\mathcal{V}_h^p\\
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\vec{z}dx=\sum_{k\in\mathcal{T}_h}\left(-\int_{\Omega_k}\nabla\cdot\vec{g}_h u_h dx+\oint_{\partial\Omega_k}\vec{g}_h\widehat{u_h}\cdot\vec{n}ds\right)\quad\forall\vec{g}_h\in\vec{\mathcal{G}}_h^p
\end{aligned}\tag{A.1.6}$$

It is now convenient to introduce the following average and jump operators for both vector and scalar quantities. The average operator for a scalar ϕ and vector $\vec{\chi}$ is defined by:

$$\begin{aligned}
\{\phi\}&=\frac{1}{2}\left(\phi^+ +\phi^-\right)\\
\{\vec{x}\}&=\frac{1}{2}\left(\vec{\chi}^+ +\vec{\chi}^-\right)
\end{aligned}\tag{A.1.7}$$

with the scalar and vector jump operators given by:

$$\begin{aligned}
\phi&=\left(\phi^+ -\phi^-\right)\vec{n}\\
[\![\vec{x}]\!]&=\left(\vec{\chi}^+ -\vec{\chi}^-\right)\cdot\vec{n}
\end{aligned}\tag{A.1.8}$$

respectively. Note that the jump in a scalar quantity is a vector and the jump in a vector quantity is a scalar. Using these operators in equation (A.1.6) is re-written in a face-based integration form, given by:

$$\begin{aligned}
&-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla v_h\cdot(D(x,u)\vec{z})\,d\Omega_k+\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![v_h]\!]\cdot(\widehat{D(x,u_h)\vec{z}})ds+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}v_h(\widehat{D(x,u_b)\vec{z}})_b\cdot\vec{n}ds=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}v_h f d\Omega_k\quad\forall v_h\in\mathcal{V}_h^p\\
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\vec{z}_h d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\cdot\vec{g}_h u_h d\Omega_k+\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\vec{g}_h]\!]\widehat{u_h}ds+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{g}_h\widehat{u_b}\cdot\vec{n}ds\quad\forall\vec{g}_h\in\vec{\mathcal{G}}_h^p
\end{aligned}\tag{A.1.9}$$

Before proceeding with the SIP derivation two identities are required:

Identity 1.

$$\left(\alpha_h\vec{\beta}_h\cdot\vec{n}\right)^+ + \left(\alpha_h\vec{\beta}_h\cdot\vec{n}\right)^- = \{\alpha_h\}\left[\!\left[\vec{\beta}_h\right]\!\right] + \left\{\vec{\beta}_h\right\}\left[\!\left[\alpha_h\right]\!\right]$$

Proof: Let the average and jump operators be defined as previously. Then

$$\{\alpha_h\}\left[\!\left[\vec{\beta}_h\right]\!\right] + \left\{\vec{\beta}_h\right\}\left[\!\left[\alpha_h\right]\!\right] =$$
$$\frac{\alpha_h^+ + \alpha_h^-}{2}\left[\left(\vec{\beta}_h\cdot\vec{n}\right)^+ + \left(\vec{\beta}_h\cdot\vec{n}\right)^-\right] + \frac{\vec{\beta}_h^+ + \vec{\beta}_h^-}{2}\cdot\left[(\alpha_h\vec{n})^+ + (\alpha_h\vec{n})^-\right]$$

Then $\vec{n}^- = -\vec{n}^+ = -\vec{n}$.

$$\frac{\alpha_h^+ + \alpha_h^-}{2}\left[\left(\vec{\beta}_h\cdot\vec{n}\right)^+ + \left(\vec{\beta}_h\cdot\vec{n}\right)^-\right] + \frac{\vec{\beta}_h^+ + \vec{\beta}_h^-}{2}\left[(\alpha_h\vec{n})^+ + (\alpha_h\vec{n})^-\right] =$$
$$\frac{\alpha_h^+\vec{\beta}_h^+\cdot\vec{n}}{2} - \frac{\alpha_h^+\vec{\beta}_h^-\cdot\vec{n}}{2} + \frac{\alpha_h^-\vec{\beta}_h^+\cdot\vec{n}}{2} - \frac{\alpha_h^-\vec{\beta}_h^-\cdot\vec{n}}{2} + \frac{\vec{\beta}_h^+\cdot\alpha_h^+\vec{n}}{2} - \frac{\vec{\beta}_h^+\cdot\alpha_h^-\vec{n}}{2} +$$
$$\frac{\vec{\beta}_h^-\cdot\alpha_h^+\vec{n}}{2} - \frac{\vec{\beta}_h^-\cdot\alpha_h^-\vec{n}}{2} = \alpha_h^+\vec{\beta}_h^+\cdot\vec{n} - \alpha_h^-\vec{\beta}_h^-\cdot\vec{n} = \left(\alpha_h\vec{\beta}_h\cdot\vec{n}\right)^+ + \left(\alpha_h\vec{\beta}_h\cdot\vec{n}\right)^-$$

□

Identity 1 is used to prove the following identity, which is a key component of the SIP derivation.

Identity 2.

$$\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\alpha_h\nabla\cdot\vec{\beta}_h + \vec{\beta}_h\cdot\nabla\alpha_h d\Omega_k = \sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{\alpha_h\}\left[\!\left[\vec{\beta}_h\right]\!\right] + \left\{\vec{\beta}_h\right\}\left[\!\left[\alpha_h\right]\!\right]ds + \sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\alpha_h\vec{\beta}_h\cdot\vec{n}ds$$

Proof: Begin by considering

$$\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\alpha_h\left(\nabla\cdot\vec{\beta}_h\right) + \vec{\beta}_h\cdot\nabla\alpha_h d\Omega_k = \sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\cdot\left(\alpha\vec{\beta}_h\right)d\Omega_k$$

Integrating by parts

$$\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\left[\left(\alpha_h\vec{\beta}_h\cdot\vec{n}\right)^+ + \left(\alpha_h\vec{\beta}_h\cdot\vec{n}\right)^-\right]ds + \sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\alpha_h\vec{\beta}_h\cdot\vec{n}ds$$

which by employing Identity 1 is

$$= \sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{\alpha_h\}\left[\!\left[\vec{\beta}_h\right]\!\right] + \left\{\vec{\beta}_h\right\}\left[\!\left[\alpha_h\right]\!\right]ds + \sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\alpha_h\vec{\beta}_h\cdot\vec{n}ds$$

This completes the proof. □

Now the flux for the solution u_h must be specified. For the SIP method the flux of $\widehat{u_h}$ is given by:

$$\begin{aligned} \widehat{u_h} &= \{u_h\} \\ \widehat{u_b} &= u_b \end{aligned} \tag{A.1.10}$$

which leaves the flux of $\vec{z}_h$ to be specified. The definition of the flux of $\vec{z}_h$ is the term that generates various DG diffusion schemes. Inserting the definition of $\widehat{u_h}$ in to equation (A.1.9) results in the following.

$$\begin{aligned} &-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla v_h\cdot(D(x,u)\vec{z})\,d\Omega_k+\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![v_h]\!]\cdot(\widehat{D(x,u_h)\vec{z}})ds+ \\ &\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}v_h(\widehat{D(x,u_b)\vec{z}})_b\cdot\vec{n}ds=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}v_h f d\Omega_k\quad\forall v_h\in\mathcal{V}_h^p \\ &\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\vec{z}_h d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\cdot\vec{g}_h u_h d\Omega_k+\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\vec{g}_h]\!]\{u_h\}\,ds+ \\ &\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{g}_h u_b\cdot\vec{n}ds\quad\forall\vec{g}_h\in\vec{\mathcal{G}}_h^p \end{aligned} \tag{A.1.11}$$

Using Identity 2 one can re-write the second of equation (A.1.11) in a more convenient form,

$$\begin{aligned} &\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\vec{z}_h d\Omega_k=\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\nabla u_h d\Omega_k-\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{\vec{g}_h\}\cdot[\![u_h]\!]ds- \\ &\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{g}_h\cdot(u_h-u_b)\,\vec{n}ds\quad\forall\vec{g}_h\in\vec{\mathcal{G}}_h^p \end{aligned} \tag{A.1.12}$$

which gives the final system of equations before the flux of $\vec{z}_h$ is specified.

$$\begin{aligned} &-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla v_h\cdot(D(x,u_h)\vec{z}_h)\,d\Omega_k+\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![v_h]\!]\cdot(\widehat{D(x,u_h)\vec{z}_h})ds+ \\ &\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}v_h(\widehat{D(x,u_h)}(\vec{z})_b)\cdot\vec{n}ds=\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}v_h f d\Omega_k\quad\forall v_h\in\mathcal{V}_h^p \\ &\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\vec{z}_h d\Omega_k=\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\nabla u_h d\Omega_k-\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{\vec{g}_h\}\cdot[\![u_h]\!]ds- \\ &\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{g}_h\cdot(u_h-u_b)\,\vec{n}ds\quad\forall\vec{g}_h\in\vec{\mathcal{G}}_h^p \end{aligned} \tag{A.1.13}$$

From this point many diffusion schemes such as BR1 [113], BR2 [73] and SIP [54–56] can be derived.

A.1.2 Symmetric Interior Penalty Method

The Symmetric Interior Penalty (SIP) method is a special case of the general mixed finite-element method. The SIP method is given by first specifying the numerical flux terms as

$$\begin{aligned}
&\widehat{(D(x,u_h)\vec{z})} = \{D(x,u_h)\nabla u_h\} - \{D(x,u_h)\nu[\![u_h]\!]\} = \{D(x,u_h)\nabla u_h\} - \nu\{D(x,u_h)\}[\![u_h]\!] \\
&\widehat{u_h} = \{u_h\} \\
&\widehat{(D(x,u_b)\mathbf{z}_b)} = D(x,u_b)\nabla u_h^+ - \nu D(x,u_b)(u_h - u_b)\vec{n}
\end{aligned} \tag{A.1.14}$$

where $\widehat{(\cdot)}_b$ denotes a numerical flux on the boundary and ν denotes the penalty parameter.

The auxiliary variable $\vec{z}_h$ is still present in the volume integral and must be eliminated from the system of equations. In order to eliminate the auxiliary variable $\vec{z}_h$ from the volume integral, substitute $\vec{g}_h = D(x,u)\nabla v_h$ into equation (A.1.13). Upon making this substitution one can see that the term involving the auxiliary variable in the volume integral can be replaced by the term in the second of equation (A.1.13).

$$\begin{aligned}
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\vec{z}_h d\Omega_k = \sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{g}_h\cdot\nabla u_h d\Omega_k - \sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{\vec{g}_h\}\cdot[\![u_h]\!]ds - \\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{g}_h\cdot(u_h-u_b)\vec{n}ds = \sum_{k\in\mathcal{T}_h}\int_{\Omega_k}D(x,u)\nabla v_h\cdot\vec{z}_h d\Omega_k = \\
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}D(x,u)\nabla v_h\cdot\nabla u_h d\Omega_k - \sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{D(x,u)\nabla v_h\}\cdot[\![u_h]\!]ds - \\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}D(x,u)\nabla v_h\cdot(u_h-u_b)\vec{n}ds \quad \forall\vec{g}_h\in\vec{\mathcal{G}}_h^p \\
&-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}D(x,u)\nabla v_h\cdot\vec{z}_h d\Omega_k = \\
&-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}D(x,u)\nabla v_h\cdot\nabla u_h d\Omega_k + \sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{D(x,u)\nabla v_h\}\cdot[\![u_h]\!]ds + \\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}D(x,u)\nabla v_h\cdot(u_h-u_b)\vec{n}ds \quad \forall\vec{g}_h\in\vec{\mathcal{G}}_h^p
\end{aligned} \tag{A.1.15}$$

The resulting additional terms in the surface integrations are known as the symmetrizing or symmetry terms. Although many methods are given in the literature that do not contain these terms, it is clear from this derivation that in order for the discretization to correspond to

a mixed finite-element method this term is required. Furthermore, it can be shown that the method will only be dual consistent if the symmetry terms are present [49]. Dual consistency is a requirement if the method is to yield optimal output functional accuracy as shown in equation (3.3.12). The final expression for the SIP method is given by:

$$
\begin{aligned}
&-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla v_h\cdot\left(D(x,u_h)\nabla u_h\right)d\Omega_k+\\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![v_h]\!]\cdot\left\{D(x,u_h)\nabla u_h\right\}+\left\{D(x,u_h)\nabla v_h\right\}\cdot[\![u_h]\!]-[\![v_h]\!]\cdot\nu\left\{D(x,u_h)\right\}[\![u_h]\!]ds+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}v_h^+D(x,u_b)\nabla u_h^+\cdot\vec{n}+D(x,u_b)\nabla v_h^+\cdot\left(u_h-u_b\right)\vec{n}-\\
&v_h^+\nu D(x,u_b)\left(u_h-u_b\right)\vec{n}\cdot\vec{n}ds=-\sum_{k\in\mathcal{T}_h}\int_{\Omega}v_h f dx\quad\forall v_h\in\mathcal{V}_h^p
\end{aligned}
\tag{A.1.16}
$$

The value of the penalty parameter ν is only required to be large enough to enforce the coercivity of the bi-linear form. The following expression has been used for all examples in this work. The derivation of this value of penalty parameter was conducted by K. Shahbazi in reference [54]. In short the expression for ν on an interface is

$$
\nu=max\left(M^+\frac{|\partial\Omega_k^+|}{|\Omega_k^+|},M^-\frac{|\partial\Omega_k^-|}{|\Omega_k^-|}\right)
\tag{A.1.17}
$$

Where $M^{\pm}$ is the number of modes, $|\Omega|^{\pm}$ is the element area, and $|\partial\Omega|^{\pm}$ is the element perimeter on each side of the interface.

This method is used successfully for Laplace equations, the Navier-Stokes equations, non-linear Laplace like equations (from RANS models) and artificial diffusion methods for shock capturing. All diffusion discretizations in this work use the SIP method with the value of penalty parameter in equation (A.1.17) for the discretization of diffusion operators.

A.2 SIP for Navier-Stokes

The derivation of the SIP method for the Navier-Stokes equations is straight forward. Consider the Navier-Stokes flux from equation (2.1.3), which is linear in gradients.

$$
\vec{\mathbf{F}}_v\left(\mathbf{u},\nabla\mathbf{u}\right)=\left[\mathbf{G}\left(\mathbf{u}\right)\right]\nabla\mathbf{u}
\tag{A.2.1}
$$

This notation can be a bit confusing so the terms on the right hand side of equation (A.2.1) are explicitly written out. Consider the viscous flux in the i^{th} coordinate direction, which is written as

$$\left(\left[\mathbf{G}\left(\mathbf{u}\right)\right]\nabla\mathbf{u}\right)_i = \sum_{j=1..3}\left[G_{ij}\left(\mathbf{u}\right)\right]\frac{\partial\mathbf{u}}{\partial x_j} \tag{A.2.2}$$

where $\left[G_{ij}\left(\mathbf{u}\right)\right]$ is an $N_f \times N_f$ matrix(5×5 for three-dimensional laminar Navier-stokes) where N_f is the number of PDEs in the system. Thus the full matrix $\left[\mathbf{G}\left(\mathbf{u}\right)\right]$ for a three-dimensional flow is given by:

$$\left[\mathbf{G}\left(\mathbf{u}\right)\right] = \begin{bmatrix} \left[G_{11}\left(\mathbf{u}\right)\right] & \left[G_{12}\left(\mathbf{u}\right)\right] & \left[G_{13}\left(\mathbf{u}\right)\right] \\ \left[G_{21}\left(\mathbf{u}\right)\right] & \left[G_{22}\left(\mathbf{u}\right)\right] & \left[G_{23}\left(\mathbf{u}\right)\right] \\ \left[G_{31}\left(\mathbf{u}\right)\right] & \left[G_{32}\left(\mathbf{u}\right)\right] & \left[G_{33}\left(\mathbf{u}\right)\right] \end{bmatrix} \tag{A.2.3}$$

Now that some clarity has been shed on the notation, consider the discretization of just the viscous fluxes of the Navier-Stokes equations. Note that the flux terms are part of a set of equations and thus = signs denote equality of one term to another. New discontinuous function spaces that are defined for a system of equations are given by:

$$\begin{aligned} \mathbf{V}_h^p &:= \left\{\mathbf{v}_h \in L^2\left(\Omega_h\right) : v|_e \in P^n\left(k\right)\forall k \in \mathcal{T}_h\right\} \\ \vec{\mathbf{G}}_h^p &:= \left\{\vec{\mathbf{g}}_h \in \left[L^2\left(\Omega_h\right)\right]^d : \vec{g}|_e \in P^n\left(k\right)\forall k \in \mathcal{T}_h\right\} \end{aligned} \tag{A.2.4}$$

The discretized viscous flux is then

$$\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\vec{\mathbf{F}}_v\left(\mathbf{u}_h,\nabla\mathbf{u}_h\right)d\Omega_k = \sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\nabla\mathbf{u}_h\right)d\Omega_k \quad \forall\mathbf{v}_h \in \mathbf{V}_h^p \tag{A.2.5}$$

which is recast as a mixed finite-element method.

$$\begin{aligned} &\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k \quad \forall\mathbf{v}_h \in \mathbf{V}_h^p \\ &\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{\mathbf{g}}_h^T\cdot\vec{\mathbf{z}}_h d\Omega_k = \sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{\mathbf{g}}_h^T\cdot\nabla\mathbf{u}_h d\Omega_k \quad \forall\vec{\mathbf{g}}_h \in \vec{\mathbf{G}}_h^p \end{aligned} \tag{A.2.6}$$

This new system of equations is integrated by parts to yield:

$$\begin{aligned}
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\mathbf{v}_h^T\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k+\\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\mathbf{v}_h]\!]^T\cdot\left(\widehat{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h}\right)ds+\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathbf{v}_h^{+T}\vec{n}\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\vec{\mathbf{z}}_b\right)ds\quad\forall\mathbf{v}_h\in\mathbf{V}_h^p\\
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{\mathbf{g}}_h^T\cdot\vec{\mathbf{z}}_h d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\cdot\vec{\mathbf{g}}_h^T\mathbf{u}_h d\Omega_k+\\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\vec{\mathbf{g}}_h]\!]^T\widehat{\mathbf{u}}_h ds+\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\widehat{\mathbf{u}_n\vec{\mathbf{g}}_h^T}\cdot\vec{n}ds\quad\forall\vec{\mathbf{g}}_h\in\vec{\mathbf{G}}_h^p
\end{aligned}\tag{A.2.7}$$

The second of equations A.2.7 is manipulated in exactly the same way as the Poisson equation. In fact, Identity 2 holds for each equation in the system independently. Thus defining

$$\begin{aligned}
\widehat{\mathbf{u}_h}&=\{\mathbf{u}_h\}\\
\widehat{\mathbf{u}}_b&=\mathbf{u}_b
\end{aligned}\tag{A.2.8}$$

and using Identity 2, the following mixed FEM system is obtained.

$$\begin{aligned}
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\mathbf{v}_h^T\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k+\\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\mathbf{v}_h]\!]^T\cdot\left(\widehat{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h}\right)ds+\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathbf{v}_h^{+T}\vec{n}\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\vec{\mathbf{z}}_b\right)ds\quad\forall\mathbf{v}_h\in\mathbf{V}_h^p\\
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{\mathbf{g}}_h^T\cdot\vec{\mathbf{z}}_h d\Omega_k=\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\cdot\vec{\mathbf{g}}_h^T\mathbf{u}_h d\Omega_k-\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\{\vec{\mathbf{g}}_h\}^T[\![\mathbf{u}]\!]_{\mathbf{h}}\mathbf{ds}-\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{\mathbf{g}}_h^T\left(\mathbf{u}_h-\mathbf{u}_b\right)\cdot\vec{n}ds\quad\forall\vec{\mathbf{g}}_h\in\vec{\mathbf{G}}_h^p
\end{aligned}\tag{A.2.9}$$

As with the Poisson equation the numerical fluxes for the terms involving $\vec{\mathbf{z}}_h$ are defined by:

$$\begin{aligned}
&\widehat{\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)}=\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\nabla\mathbf{u}_h\right\}-\nu\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right][\![\mathbf{u}_h]\!]\right\}=\\
&\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\nabla\mathbf{u}_h\right\}-\nu\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\right\}[\![\mathbf{u}_h]\!]\\
&\widehat{\mathbf{u}_h}=\{\mathbf{u}_h\}\\
&\widehat{\left(\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\mathbf{z}_b\right)}=\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\nabla\mathbf{u}_h^+-\nu\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\left(\mathbf{u}_h-\mathbf{u}_b\right)\vec{n}
\end{aligned}\tag{A.2.10}$$

With the fluxes of the auxiliary variables defined the discretization of the Navier-Stokes flux can be written explicitly as:

$$\begin{aligned}
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\mathbf{v}_h^T\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k+\\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\mathbf{v}_h]\!]^T\cdot\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\nabla\mathbf{u}_h\right\}-[\![\mathbf{v}_h]\!]^T\cdot\nu\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\right\}[\![\mathbf{u}_h]\!]ds+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathbf{v}_h^{+T}\vec{n}\cdot\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\nabla\mathbf{u}_h^+-\mathbf{v}_h^{+T}\nu\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\left(\mathbf{u}_h-\mathbf{u}_b\right)\vec{n}\cdot\vec{n}ds\quad\forall\mathbf{v}_h\in\mathbf{V}_h^p\\
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\vec{\mathbf{g}}_h^T\cdot\vec{\mathbf{z}}_h d\Omega_k=\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\cdot\vec{\mathbf{g}}_h^T\mathbf{u}_h d\Omega_k-\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}\left\{\vec{\mathbf{g}}_h\right\}^T[\![\mathbf{u}_h]\!]ds-\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\vec{\mathbf{g}}_h^T\left(\mathbf{u}_h-\mathbf{u}_b\right)\cdot\vec{n}ds\quad\forall\vec{\mathbf{g}}_h\in\vec{\mathbf{G}}_h^p
\end{aligned}\tag{A.2.11}$$

As with the Poisson equation the auxiliary variable $\vec{\mathbf{z}}_h$ is eliminated by using the auxiliary equation and substituting $\vec{\mathbf{g}}_h^T=\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]^{T_{block}}\nabla\mathbf{v}_h$. Applying this substitution results in the final from of the SIP numerical flux for the Navier-Stokes equations.

$$\begin{aligned}
&\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\mathbf{v}_h^T\nabla\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{z}}_h\right)d\Omega_k=-\sum_{k\in\mathcal{T}_h}\int_{\Omega_k}\nabla\mathbf{v}_h^T\cdot\left(\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\vec{\mathbf{u}}_h\right)d\Omega_k+\\
&\sum_{i\in\mathcal{I}_h}\int_{\Gamma^i}[\![\mathbf{v}_h]\!]^T\cdot\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\nabla\mathbf{u}_h\right\}+\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]^{T_{block}}\nabla\mathbf{v}_h\right\}\cdot[\![\mathbf{u}_h]\!]-\\
&[\![\mathbf{v}_h]\!]^T\cdot\nu\left\{\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]\right\}[\![\mathbf{u}_h]\!]ds+\\
&\sum_{b\in\mathcal{B}_h}\int_{\Gamma^b}\mathbf{v}_h^{+T}\vec{n}\cdot\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\nabla\mathbf{u}_h^++\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]^{T_{block}}\nabla\mathbf{v}_h^+\cdot\left(\mathbf{u}_h-\mathbf{u}_b\right)\vec{n}-\\
&\mathbf{v}_h^{+T}\nu\left[\mathbf{G}\left(\mathbf{u}_b\right)\right]\left(\mathbf{u}_h-\mathbf{u}_b\right)\vec{n}\cdot\vec{n}ds\quad\forall\mathbf{v}_h\in\mathbf{V}_h^p
\end{aligned}\tag{A.2.12}$$

This defines the SIP flux for the Navier-Stokes equations. Notice that the symmetry term is not at all ad-hoc. In fact, it comes from a mixed finite-element method, which is where other DG diffusion schemes such as LDG [114],BR1 [113], and BR2 [73] have their roots.

As a final note the $[]^{T_{block}}$ is often a source of confusion for DG diffusion discretizations. Consider the $\left[\mathbf{G}\left(\mathbf{u}_h\right)\right]$ for the three-dimensional Navier-Stokes equations given in equation

(A.2.3), the block transpose of this is

$$[\mathbf{G}(\mathbf{u})]^{T_{block}} = \begin{bmatrix} [G_{11}(\mathbf{u})] & [G_{21}(\mathbf{u})] & [G_{31}(\mathbf{u})] \\ [G_{12}(\mathbf{u})] & [G_{22}(\mathbf{u})] & [G_{32}(\mathbf{u})] \\ [G_{13}(\mathbf{u})] & [G_{23}(\mathbf{u})] & [G_{33}(\mathbf{u})] \end{bmatrix} \tag{A.2.13}$$

with no transpose of the blocks themselves.

A.2.1 Implementing SIP for Navier-Stokes

While the presented formulas are general, these formulas can be confusing for a first time reader. It is helpful to clarify the formulas by giving an example of the full detailed expansion of the SIP flux.

The flux on the interior interfaces is given by:

$$\mathcal{H}_v = [\![\mathbf{v}_h]\!]^T \cdot \{[\mathbf{G}(\mathbf{u}_h)]\nabla\mathbf{u}_h\} + \left\{[\mathbf{G}(\mathbf{u}_h)]^{T_{block}}\nabla\mathbf{v}_h\right\} \cdot [\![\mathbf{u}_h]\!] - [\![\mathbf{v}_h]\!]^T \cdot \nu\{[\mathbf{G}(\mathbf{u}_h)]\}[\![\mathbf{u}_h]\!] \tag{A.2.14}$$

For two dimensional flow this can be written as

$$\begin{aligned} \mathcal{H}_v = {} & [\![\mathbf{v}_h]\!]^T \cdot \{(\mathbf{F}_v^x(\mathbf{u}_h, \nabla\mathbf{u}_h), \mathbf{F}_v^y(\mathbf{u}_h, \nabla\mathbf{u}_h))\} + \\ & \left\{\left([G_{11}]\frac{\partial\mathbf{v}_h}{\partial x} + [G_{21}]\frac{\partial\mathbf{v}_h}{\partial y}, [G_{21}]\frac{\partial\mathbf{v}_h}{\partial x} + [G_{22}]\frac{\partial\mathbf{v}_h}{\partial y}\right)\right\} \cdot [\![\mathbf{u}_h]\!] + \\ & [\![\mathbf{v}_h]\!]^T \cdot \nu\{([G_{11}]\Delta\mathbf{u}_h n_x + [G_{12}]\Delta\mathbf{u}_h n_y, [G_{21}]\Delta\mathbf{u}_h n_x + [G_{22}]\Delta\mathbf{u}_h n_y)\} \\ & \Delta\mathbf{u}_h = \left(\mathbf{u}_h^+ - \mathbf{u}_h^-\right) \end{aligned} \tag{A.2.15}$$

This expression defines the SIP flux on an interior boundary. However, this is not how the SIP flux is implemented within the solver. Careful inspection of equation (A.2.15) reveals that this expansion is more complicated than necessary. In fact, the SIP flux can be written without any reference to the [**G**] matrix. Recall equation (A.2.1), which shows that the viscous flux is linear in solution gradients hence equation (A.2.15) is written in a very simple form as:

$$\begin{aligned} \mathcal{H}_v = {} & [\![\mathbf{v}_h]\!]^T \cdot \{(\mathbf{F}_v^x(\mathbf{u_h}, \nabla\mathbf{u_h}), \mathbf{F}_v^y(\mathbf{u}_h, \nabla\mathbf{u}_h))\} + \\ & \{(\mathbf{F}_v^x(\mathbf{u}_h, [\![\mathbf{u}_h]\!]), \mathbf{F}_v^y(\mathbf{u}_h, [\![\mathbf{u}_h]\!])) \cdot \nabla\mathbf{v}_h\} \\ & [\![\mathbf{v}_h]\!]^T \cdot \{(\mathbf{F}_v^x(\mathbf{u}_h, \nu[\![\mathbf{u}_h]\!]), \mathbf{F}_v^y(\mathbf{u}_h, \nu[\![\mathbf{u}_h]\!]))\} \end{aligned} \tag{A.2.16}$$

which is further compacted into

$$\mathcal{H}_v = [\![\mathbf{v}_h]\!]^T \cdot \left\{ \vec{\mathbf{F}}_v \left(\mathbf{u}_h, \nabla \mathbf{u}_h \right) \right\} + \left\{ \vec{\mathbf{F}}_v \left(\mathbf{u}_h, [\![\mathbf{u}_h]\!] \right) \cdot \nabla \mathbf{v}_h \right\} + [\![\mathbf{v}_h]\!]^T \cdot \left\{ \vec{\mathbf{F}}_v \left(\mathbf{u}_h, \nu [\![\mathbf{u}_h]\!] \right) \right\} \quad \text{(A.2.17)}$$

which is an expression for any number of equations and dimensions. This is how the SIP flux is implemented in the most efficient and straight forward manner. This form makes it easy for one to compute the flux and also to linearize the flux to obtain the flux Jacobian used in implicit solvers.

The boundary viscous flux is written in an analogous form

$$\mathcal{H}_v^b = \mathbf{v}_h^{+T} \vec{\mathbf{F}}_v \left(\mathbf{u}_b, \nabla \mathbf{u}_h^+ \right) \cdot \vec{n} + \vec{\mathbf{F}}_v \left(\mathbf{u}_b, \left(\mathbf{u}_h - \mathbf{u}_b \right) \vec{n} \right) \cdot \nabla \mathbf{v}_h^+ + \mathbf{v}_h^{+T} \vec{\mathbf{F}}_v \left(\mathbf{u}_b, \nu \left(\mathbf{u}_h - \mathbf{u}_b \right) \vec{n} \right) \cdot \vec{n} \quad \text{(A.2.18)}$$

which is derived via an analogous proceedure to the one used for the interior flux. Notice that this form of the SIP flux is especially simple and shows the elegance of the SIP method. The SIP method is not as ad-hoc as it might seem upon initial inspection. Rather, the SIP method is the result of very clever choices of the numerical fluxes of a mixed finite-element method. Furthermore, the symmetry terms are not an arbitrary addition to the SIP flux, but rather a necessary component of the SIP flux based on the derivation presented. While the derivation of the SIP method is detailed, the stability properties of the SIP method have not been discussed. Reference [54] presents an excellent discussion of stability and coercivity of the SIP discretization for Poisson's equation.

Appendix B

Spalart Allmaras Modifications for Negative Values

In this work the Reynolds Averaged Navier-Stokes (RANS) equations are closed using the turbulence model of Spalart and Allmaras (SA turbulence model) [41]. Due to the non-smooth behavior of high-order DG discretizations of the SA turbulence model, this work utilizes modifications of the SA turbulence model source terms. These modifications were initially presented in reference [20] and are designed to help stabilize the turbulence model when negative values of the turbulence model working variable $\tilde{\nu}$ are encountered. Reference [20] has proven, in an energy norm sense, that these modifications represent an energy stable turbulence model. However, even with the modifications, negative values of $\tilde{\nu}$ induce strong transients that can easily cause the steady-state Newton solver to fail.

In the experiences of this work, the solver will often fail if DG discretizations of the SA turbulence model equation are employed for more challenging flows even with the modified source terms. The experience gained in this work has demonstrated that the turbulence model working variable values must remain positive to prevent solver failure. While the modifications to the source terms aided a small amount in preventing solver failure for flat plate and simple airfoil flows, the modifications are insufficient to make high-order DG discretizations of the turbulence model equation truly robust. In this section the modifications of the turbulence model source terms are discussed and it is shown what the behavior of the

source terms are for negative $\tilde{\nu}$ both with and without the modifications. Full details of the original form of the SA model are given in reference [41].

B.1 Modified Spalart Allmaras Model

The equation for the modified SA turbulence model is given by:

$$\frac{\partial \rho\tilde{\nu}}{\partial t} + \nabla \cdot (\rho\tilde{\nu}\vec{u}) = \mathcal{P}(\mathbf{u}, \nabla\mathbf{u}) + \frac{1}{\sigma}\left[\nabla \cdot (\eta\nabla\tilde{\nu}) + c_{b_2}\rho\nabla\tilde{\nu} \cdot \nabla\tilde{\nu}\right] - \mathcal{D}(\mathbf{u}, \nabla\mathbf{u}) \tag{B.1.1}$$

where $\tilde{\nu}$ is the SA or turbulence model working variable, ρ is the density, $\vec{u}$ is the velocity field vector, η is modified diffusion coefficient, $\mathcal{P}(\mathbf{u}, \nabla\mathbf{u})$ is the production source term, $\mathcal{D}(\mathbf{u}, \nabla\mathbf{u})$ is the destruction source term, d is the distance to the closest wall and the term multiplied by $\frac{1}{\sigma}$ is the diffusion term. Additionally, σ and c_{b_2} are turbulence model constants with the values of these constants specified below. The modified diffusion coefficient is given by:

$$\eta = \begin{cases} (\mu + \rho\tilde{\nu}) & \rho\tilde{\nu} \geq 0 \\ \left(\mu + \rho\tilde{\nu} + \frac{(\rho\tilde{\nu})^2}{\mu}\right) & \rho\tilde{\nu} < 0 \end{cases} \tag{B.1.2}$$

which is designed to remain positive regardless of the sign of $\tilde{\nu}$. The modified production term $\mathcal{P}(\mathbf{u}, \nabla\mathbf{u})$ is given by:

$$\begin{aligned} \mathcal{P}(\mathbf{u}, \nabla\mathbf{u}) &= \begin{cases} c_{b_1}\tilde{S}\rho\tilde{\nu} & \chi \geq 0 \\ c_{b_1}S\rho\tilde{\nu}g & \chi < 0 \end{cases} \\ g_n &= 1 - \frac{1000\chi^2}{1+\chi^2} \\ \chi &= \frac{\rho\tilde{\nu}}{\mu} \end{aligned} \tag{B.1.3}$$

with $\tilde{S}$ given according to

$$\tilde{S} = \begin{cases} S + \bar{S} & \bar{S} \geq -c_{v_2}S \\ S + \frac{S\left(c_{v_2}^2 S + c_{v_3}\bar{S}\right)}{\left(c_{v_3} - 2c_{v_2}\right)S - \bar{S}} & \bar{S} < -c_{v_2}S \end{cases} \tag{B.1.4}$$

$$
\begin{aligned}
S &= \sqrt{\vec{\omega} \cdot \vec{\omega}} \\
\bar{S} &= \frac{\tilde{\nu}^2 f_{v_2}}{\kappa^2 d^2} \\
f_{v_1} &= \frac{\chi^3}{\chi^3 + c_{v_1}^3} \\
f_{v_2} &= 1 - \frac{\chi}{1 + \chi f_{v_1}}
\end{aligned}
\tag{B.1.5}
$$

where $\vec{\omega}$ is the vorticity vector. The modified destruction term is

$$
\begin{aligned}
\mathcal{D}\left(\mathbf{u}, \nabla \mathbf{u}\right) &= \begin{cases} c_{w_1} \rho f_w \left(\frac{\tilde{\nu}}{d}\right)^2 & \chi \geq 0 \\ -c_{w_1} \rho \left(\frac{\tilde{\nu}}{d}\right)^2 & \chi < 0 \end{cases} \\
r &= \frac{\tilde{\nu}}{\tilde{S} \kappa^2 d^2} \\
g &= r + c_{w_2}\left(r^6 - r\right) \\
f_w &= g \left[\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right]^{1/6}
\end{aligned}
\tag{B.1.6}
$$

where the turbulence model constants are $c_{b_1} = 0.135$, $c_{b_2} = 0.622$, $\sigma = 2/3$, $\kappa = 0.41$, $c_{w_1} = \frac{c_{b_1}}{\kappa^2} + \frac{1 + c_{b_2}}{\sigma}$, $c_{w_2} = 0.3$, $c_{w_3} = 2$, $c_{v_1} = 7.1$, $c_{v_2} = 0.7$, and $c_{v_3} = 0.9$. Lastly the turbulent eddy viscosity μ_T is given by:

$$
\mu_T = \begin{cases} \rho \tilde{\nu} f_{v_1} & \tilde{\nu} \geq 0 \\ 0 & \tilde{\nu} < 0 \end{cases}
\tag{B.1.7}
$$

For reference the original form of the turbulence model from reference [41] is given by:

$$
\mathcal{S}_o\left(\mathbf{u}, \nabla \mathbf{u}\right) = \mathcal{P}_o - \mathcal{D}_o
\tag{B.1.8}
$$

The source terms $\mathcal{P}_o$ and $\mathcal{D}_o$ are the original production and destruction source terms respectively and are given by:

$$
\begin{aligned}
\mathcal{P}_o &= c_{b_1} \tilde{S} \rho \tilde{\nu} \\
\mathcal{D}_o &= c_{w_1} \rho f_w \left(\frac{\tilde{\nu}}{d}\right)^2
\end{aligned}
\tag{B.1.9}
$$

Since this form of the SA turbulence model represents the original form found in [41] the terms are denoted by $()_o$. The production and destruction terms are analyzed individually and then added together to show the overall behavior of the source term under negative $\tilde{\nu}$ conditions. Much of this analysis relies on graphical results, which are used to illustrate the behavior of the source terms and are generated using non-dimensional groups.

B.2 Production Modifications

The original production term is

$$\mathcal{P}_o = c_{b_1}\tilde{S}\rho\tilde{\nu} \tag{B.2.1}$$

where the $\tilde{S}$ is a vorticity like term, which is modified to be positive for all values of $\tilde{\nu}$ and C^1 continuous.

$$\begin{aligned}
\tilde{S}_o &= S + \bar{S} \\
\tilde{S} &= \begin{cases} S + \bar{S} & \bar{S} \geq -c_{v_2}S \\ S + \frac{S\left(c_{v_2}^2 S + c_{v_3}\bar{S}\right)}{\left(c_{v_3} - 2c_{v_2}\right)S - \bar{S}} & \bar{S} < -c_{v_2}S \end{cases} \\
\bar{S} &= \frac{\tilde{\nu}^2 f_{v_2}}{\kappa^2 d^2} \\
S &= \left|\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right|
\end{aligned} \tag{B.2.2}$$

Where the $\bar{S}$ is given by:

$$\begin{aligned}
\bar{S} &= \frac{\tilde{\nu}^2 f_{v_2}}{\kappa^2 d^2} \\
f_{v_2} &= 1 - \frac{\chi}{1 + \chi f v_{v_1}} \\
f_{v_1} &= \frac{\chi^3}{\chi^3 + c_{v_1}^3} \\
\chi &= \frac{\rho\tilde{\nu}}{\mu}
\end{aligned} \tag{B.2.3}$$

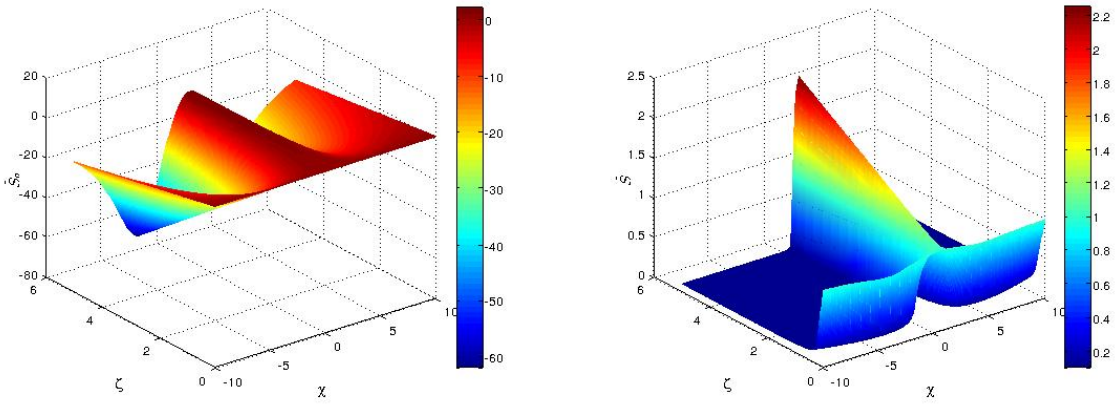
The only component of $\tilde{S}_o$ that can be negative for $\tilde{\nu} < 0$ is $\bar{S}$. In order to conduct a parameter study of the source terms and the associated modifications, non-dimensional groupings of the turbulence model inputs are devised such that the number of non-dimensional parameters is two. These non-dimensional groups are chosen in order to isolate the effects of the SA turbulence model working variable $\tilde{\nu}$ from the other turbulence model inputs. The $\tilde{S}$ term has units of vorticity and is non-dimensionalized by vorticity magnitude S to give

$$\begin{aligned}
\frac{\tilde{S}_o}{S} &= 1 + \frac{\bar{S}}{S} \\
\frac{\tilde{S}}{S} &= \begin{cases} 1 + \frac{\bar{S}}{S} & \frac{\bar{S}}{S} \geq -c_{v_2} \\ 1 + \frac{\left(c_{v_2}^2 + c_{v_3}\frac{\bar{S}}{S}\right)}{\left(c_{v_3} - 2c_{v_2}\right) - \frac{\bar{S}}{S}} & \frac{\bar{S}}{S} < -c_{v_2} \end{cases}
\end{aligned} \tag{B.2.4}$$

for both the original and modified terms. The result of non-dimesionalizing $\bar{S}$ with S is

$$\begin{aligned}
\frac{\bar{S}}{S} &= \frac{\tilde{\nu} f_{v_2}}{\kappa^2 d^2 S} = \frac{\nu \chi f_{v_2}}{\kappa^2 d^2 S} = \frac{\zeta \chi f_{v_2}}{\kappa^2} \\
\zeta &= \frac{\nu}{d^2 S} \\
\nu &= \frac{\mu}{\rho}
\end{aligned} \tag{B.2.5}$$

which gives the two non-dimensional parameters χ and ζ. χ is the non-dimensional turbulence model working variable, which gives the effect of the turbulence model working variable on the source terms. ζ is a non-dimensional grouping of the remaining inputs to the turbulence model, including velocity field (S), fluid (μ), and geometric (d) inputs. The $\tilde{S}$ terms



(a) $\frac{\tilde{S}_o}{S}$ (b) $\frac{\tilde{S}}{S}$

Figure B.1: Original and modified $\frac{\tilde{S}}{S}$ across the parameter space.

are shown in Figure B.1. The effect of the modification is to keep $\tilde{S}$ positive for all values of χ. One should note that the original form of the turbulence model does not keep $\tilde{S}$ positive for all positive χ, whereas the modification yields a positive $\tilde{S}$ for all χ, both positive and negative. The original and modified non-dimensional production terms are given by:

$$\begin{aligned}
\frac{\mathcal{P}_o}{\mu S} &= c_{b_1} \frac{\tilde{S}_o}{S} \chi \\
\frac{\mathcal{P}}{\mu S} &= \begin{cases} c_{b_1} \frac{\tilde{S}}{S} \chi & \chi \geq 0 \\ c_{b_1} \chi g_n & \chi < 0 \end{cases} \\
g_n &= 1 - \frac{1000 \chi^2}{1 + \chi^2}
\end{aligned} \tag{B.2.6}$$

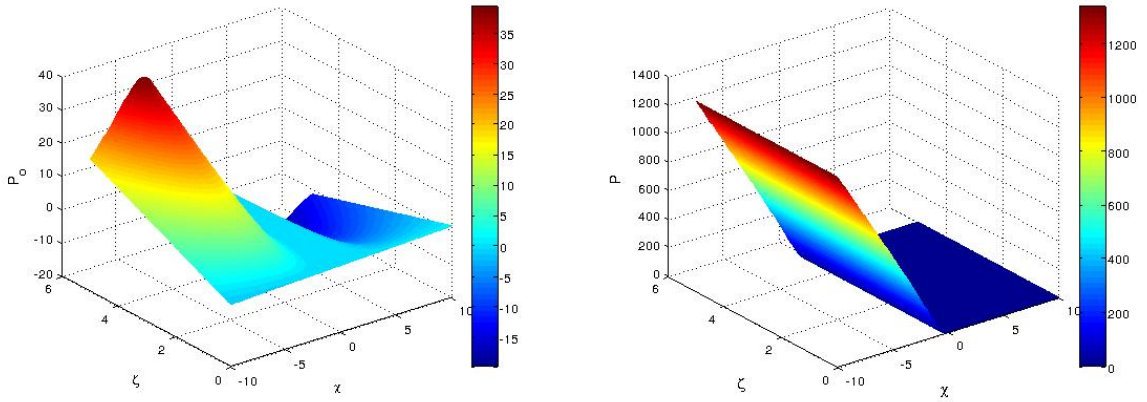
which are depicted graphically in Figure B.2.

As shown in Figure B.2, the modified production term becomes very strong in the negative χ region, which results in the modified production term attempting to increase the values of $\tilde{\nu}$. While this modification may be mathematically stable, it is destabilizing in a practical sense, which results from the fact that the negative values of χ occur when the flow is starting to converge after the initial turbulence model transient. Hence when the production term suddenly becomes very strong it induces a sudden and strong transient in the Newton solver and causes the residual to grow. These parameter space plots in Figure B.2 show that the presented modification to the production term is not a modification that allows for the negative values of $\tilde{\nu}$ to be a valid solution of the turbulence model equation. Rather, the modification to the production term is an attempt to force $\tilde{\nu}$ to be positive everywhere. Unfortunately once positivity is achieved the turbulence model equation is back in the original form, which initially caused the negative values of $\tilde{\nu}$. While trying to force the turbulence model working variable to be positive everywhere should be an appropriate strategy, this strategy must be complimented by a modification to eliminate the artificial source of the negative $\tilde{\nu}$ values. Thus in practice the result of this modification is a very strong transient that is overly difficult to overcome and in many cases causes solver failure for the turbulence model discretization. This work has found this modification to the production term insufficiently robust for the solution of complex flow problems.

B.3 Destruction Modifications

Using the same non-dimensional groupings the destruction term is defined non-dimensionally. The only additional intermediate term that needs to be non-dimensionalized is the input r to the destruction coefficient f_w:

$$r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} = \frac{\chi\zeta}{\kappa^2 \frac{\tilde{S}}{S}} \tag{B.3.1}$$



(a) $\frac{\mathcal{P}_o}{\mu S}$ (b) $\frac{\mathcal{P}}{\mu S}$

Figure B.2: Original and modified non-dimensional production source term across the parameter space defined by ζ and χ.

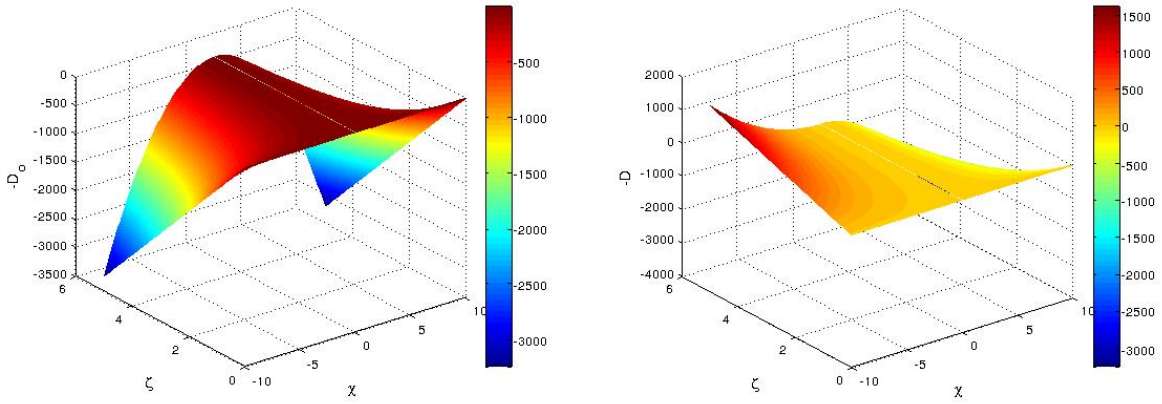
which gives the coefficient of the destruction term as:

$$\begin{aligned} g &= r + c_{w_2}(r^6 - r) \\ f_w &= g\left(\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right)^{1/6} \end{aligned} \tag{B.3.2}$$

and the scaled destruction term as:

$$\begin{aligned} \frac{\mathcal{D}_o}{\mu S} &= c_{w_1} f_w \frac{\rho}{\mu S}\left(\frac{\tilde{\nu}^2}{d^2}\right) = c_{w_1} f_w \chi^2 \zeta \\ \frac{\mathcal{D}}{\mu S} &= \begin{cases} c_{w_1} f_w \chi^2 \zeta & \chi \geq 0 \\ -c_{w_1} \chi^2 \zeta & \chi < 0 \end{cases} \end{aligned} \tag{B.3.3}$$

Figure B.3 shows the non-dimensional destruction term across the parameter space. As with the production term, the modification for negative χ has caused $-\mathcal{D}$ to become a positive source or production term. Since positive source terms tend to try and grow the values of the turbulence model working variable, this modification induces a strong transient in the Newton solver and will also push the turbulence model further from convergence. Similarly to the modified production term, when negative values of χ start to occur in the solution, this modification destabilizes the solver due to sudden production term transient behavior.



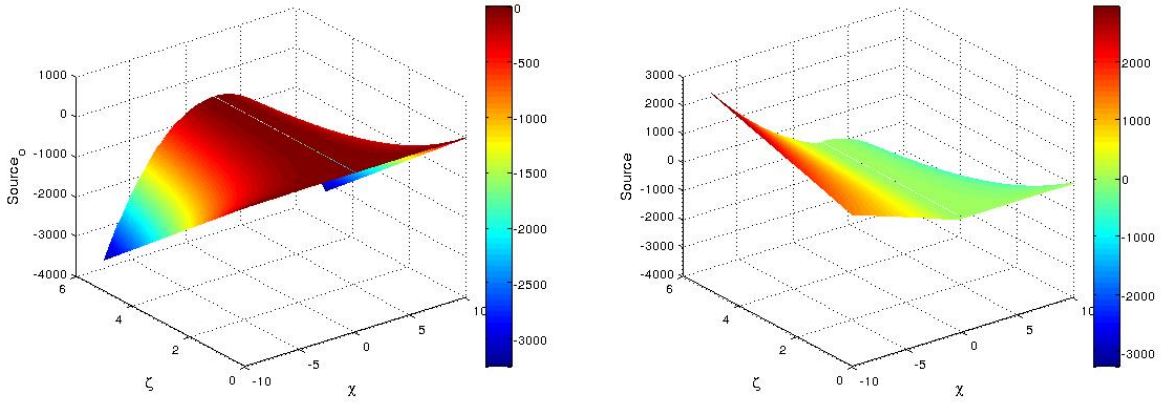
(a) $\frac{\mathcal{D}_o}{\mu S}$ (b) $\frac{\mathcal{D}}{\mu S}$

Figure B.3: Original and modified destruction source terms across the parameter space defined by ζ and χ.

B.4 Full Source Term

The original and modified non-dimensional source term is depicted in Figure B.4. Compared to the original source term the modified one is strongly productive in the negative χ region. This production transient often causes the implicit solver to fail when trying to solve complex flows with strong turbulence model discontinuities, such as high-lift cases. While these modifications are designed to stabilize the turbulence model for negative χ, the result is that the modified turbulence model is no more amenable high-order discretization than the original version, though for a fundamentally different reason. While the negative χ region for the original turbulence model is energy unstable the modified turbulence model is energy stable due to the presented modifications [20]. However, the transients induced by the modified source terms are strong enough to cause the implicit solver to fail. Furthermore, should the modifications succeed in producing positive χ everywhere, the turbulence model source term takes the original form. However, this original form of the turbulence model initially generated the negative values of the turbulence model working variable. Hence the analysis of the modifications to the production and destruction source terms does not clearly demonstrate how these modification are increasing the robustness of the turbulence model equation for negative values of $\tilde{\nu}$. Rather the analysis has demonstrated that these

modifications have the potential to induce very strong transients during the solution of the turbulence model equation, which can cause the implicit solver to fail. In order to have a truly robust high-order amenable turbulence model, the source of the non-smooth behavior of the turbulence model discrete equations needs to be addressed directly.



(a) Original Full Source Term

(b) Full Source Term

Figure B.4: Original and modified SA model source terms across the parameter space defined by ζ and χ.

Appendix C

Non-dimensionalization

Though most computational fluid dynamics (CFD) solvers utilize non-dimensional scalings of the governing variables in order to keep the magnitudes of the computed results relatively close together, all the equations in this work are presented in dimensional form. This form is presented to allow the reader to apply their own non-dimensional groupings when implementing the methods described within the body of the dissertation. Furthermore, showing non-dimensional equations can sometimes add additional parameters that can become confusing and lead to mistakes when implementing a particular formula. This section specifies the non-dimensional parameters used in the CFD solver written for this work. The non-dimensional parameters are the same as the CFL3D flow solver [66].

C.1 Non-dimensional Variables for the RANS equations

Recall the conserved variable vector $\mathbf{u}$ and flux vectors from equation (2.1.3). All of the flow field variables in these equations need to be suitable non-dimensionalized. The non-dimensional flow field variables are given by defining a suitable reference state, which is used to scale the flow field quantities. The reference state in this work consists of defining the reference velocity u_{ref}, density ρ_{ref}, temperature T_{ref}, viscosity μ_{ref} at the free-stream $()_\infty$

state as:

$$\begin{aligned} \rho_{ref} &= \rho_\infty \\ u_{ref} &= a_\infty \\ T_{ref} &= T_\infty \\ \mu_{ref} &= \mu_\infty \end{aligned} \tag{C.1.1}$$

Additionally, a reference length L_{ref} is defined as an input to the solver. Let $\bar{()}$ denote a non-dimensional flow field quantity. The non-dimensional flow field and free-stream quantities are given as:

$$\begin{aligned} \bar{\rho} &= \frac{\rho}{\rho_\infty} & \bar{\rho}_\infty &= 1 \\ \bar{u} &= \frac{u}{a_\infty} & \bar{u}_\infty &= M_\infty \cos\alpha \\ \bar{v} &= \frac{v}{a_\infty} & \bar{v}_\infty &= M_\infty \sin\alpha \\ \bar{P} &= \frac{P}{\rho_\infty a_\infty^2} & \bar{P}_\infty &= \frac{1}{\gamma} \\ \bar{E}_t &= \frac{E_t}{\rho_\infty a_\infty^2} & \left(\bar{E}_t\right)_\infty &= \frac{1}{\gamma\left(\gamma-1\right)} + \frac{M_\infty^2}{2} \\ \bar{a} &= \frac{a}{a_\infty} & \bar{a}_\infty &= 1 \\ \bar{T} &= \frac{T}{T_\infty} & \bar{T}_\infty &= 1 \end{aligned} \tag{C.1.2}$$

The non-dimensional spatial and temporal coordinates are given as:

$$\begin{aligned} \bar{x} &= \frac{x}{L_{ref}} & \bar{y} &= \frac{y}{L_{ref}} \\ \bar{t} &= \frac{t a_\infty}{L_{ref}} \end{aligned} \tag{C.1.3}$$

The non-dimensional viscosity is given by Sutherland's law as:

$$\bar{\mu} = \frac{\mu}{\mu_\infty} = \bar{T}^{\frac{3}{2}} \left[\frac{1+\frac{c}{T_\infty}}{\bar{T}+\frac{c}{T_\infty}}\right] \tag{C.1.4}$$

and the non-dimensional Spalart Allmaras turbulence model working variable and turbulent eddy viscosity are given as:

$$\begin{aligned} \bar{\rho\tilde{\nu}} &= \frac{\rho\tilde{\nu}}{\mu_\infty} & \bar{\mu}_T &= \frac{\mu_T}{\mu_\infty} \end{aligned} \tag{C.1.5}$$

Additionally, the non-dimensional artificial viscosity $\bar{\hat{\epsilon}}$ is given by:

$$\bar{\hat{\epsilon}} = \frac{\hat{\epsilon}}{a_\infty L_{ref}} \tag{C.1.6}$$

Appendix D

Roe Flux Function for Turbulence Models

Solving turbulent flows using the Reynolds Averaged Navier-Stokes (RANS) equations requires the discretization of one or more turbulence model equations. In this work, the one-equation turbulence model of Spalart and Allmaras (SA) [41] is used to close the RANS equations. Many production level solvers [5,64–66] treat the discretization of the turbulence model in a decoupled fashion, which results in treating the convection term as though it were a scalar transport equation evolving with a prescribed velocity field. However, this treatment neglects the fact that the velocity field, which convects the turbulence model quantities, is heavily influence by the turbulence model solution. Therefore, in this work the turbulence model used to close the equations is fully coupled to the mean flow equations and the RANS-SA system is considered as a complete system of equations. This treatment necessitates re-deriving the convective numerical flux function to include the turbulence model equation.

D.1 Roe's Riemann Solver for RANS-SA System

The RANS equations closed with the SA turbulence model results in a total of five PDEs. In order to produce a stable and accurate convective discretization, the numerical flux func-

tion must be derived for the convective terms of the five coupled equations. Consider the convective flux vectors, originally given in equation (2.1.3), repeated here as:

$$\mathbf{F_c}^x = \begin{Bmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ u(E_t + P) \\ \rho u\tilde{\nu} \end{Bmatrix}, \quad \mathbf{F_c}^y = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ v(E_t + P) \\ \rho v\tilde{\nu} \end{Bmatrix}, \tag{D.1.1}$$

Considering these flux vectors, the method of Roe and Pike [50] is used to derive a numerical flux function for the coupled RANS-SA system. The method of Roe and Pike requires the eigenvalues and eigenvectors of the Jacobian of the convective flux normal to an interface. The normal flux $\mathbf{F_c}^n$ is given by:

$$\mathbf{F_c}^n = \begin{Bmatrix} \rho u n_x + \rho v n_y \\ (\rho u^2 + P)\, n_x + \rho uv n_y \\ \rho uv n_x + (\rho v^2 + P)\, n_y \\ u(E_t + P)n_x + v(E_t + P)n_y \\ \rho u\tilde{\nu} n_x + \rho v\tilde{\nu} n_y \end{Bmatrix} \tag{D.1.2}$$

The eigenvalues λ and eigenvectors K of the fully coupled RANS-SA convective flux Jacobian are:

$$\lambda_1 = \vec{u}\cdot\vec{n} - a \quad \lambda_2 = \vec{u}\cdot\vec{n} \qquad \lambda_3 = \vec{u}\cdot\vec{n} \quad \lambda_4 = \vec{u}\cdot\vec{n} \quad \lambda_5 = \vec{u}\cdot\vec{n} + a$$

$$K_1 = \begin{Bmatrix} 1 \\ u - an_x \\ v - an_y \\ H - \vec{u}\cdot\vec{n}a \\ s \end{Bmatrix}, K_2 = \begin{Bmatrix} 1 \\ u \\ v \\ \frac{1}{2}(u^2 + v^2) \\ 0 \end{Bmatrix}, K_3 = \begin{Bmatrix} 0 \\ -n_y \\ n_x \\ -un_y + vn_x \\ 0 \end{Bmatrix}, \tag{D.1.3}$$

$$K_4 = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix}, K_5 = \begin{Bmatrix} 1 \\ u + an_x \\ v + an_y \\ H + \vec{u}\cdot\vec{n}a \\ s \end{Bmatrix},$$

where $\vec{u} = (u, v)$ are the Cartesian velocity components, $\vec{n} = (n_x, n_y)$ is the surface normal vector, a is the sound speed, H is the total enthalpy, and s is the scalar, which would be $\tilde{nu}$ for the SA turbulence model equation.

Following reference [50] the numerical flux at the interface, given by the Roe and Pike method, $\mathcal{H}_c$ is written as:

$$\mathcal{H}_c\left(\mathbf{u}^+, \mathbf{u}^-\right) = \frac{1}{2}\left(\mathbf{F_c}^n\left(\mathbf{u}^+\right) + \mathbf{F_c}^n\left(\mathbf{u}^-\right) + \sum_i \tilde{\alpha}_i \left|\tilde{\lambda}_i\right| \tilde{K}_i \left(\mathbf{u}^+ - \mathbf{u}^-\right)\right) \tag{D.1.4}$$

This expression requires the determination of the wave strength coefficients α_i and the Roe state $\tilde{()}$, which are determined using the following formulas:

$$\begin{aligned} \Delta\mathbf{u} &= \left(\mathbf{u}^+ - \mathbf{u}^-\right) = \sum_i \tilde{\alpha}_i \tilde{K}_i \\ \Delta\mathbf{F_c}^n &= \mathbf{F_c}^n\left(\mathbf{u}^+\right) - \mathbf{F_c}^n\left(\mathbf{u}^-\right) = \sum_i \tilde{\alpha}_i \tilde{\lambda}_i \tilde{K}_i \end{aligned} \tag{D.1.5}$$

evaluated to $\mathcal{O}\left(\Delta^2\right)$. For example, it is easy to see that

$$\Delta\left(\rho u\right) = \tilde{u}\Delta\rho + \tilde{\rho}\Delta u + \mathcal{O}\left(\Delta^2\right) \tag{D.1.6}$$

To $\mathcal{O}\left(\Delta^2\right)$, the wave strength coefficients are determined by solving the following system of equations:

$$\left\{\begin{array}{c} \tilde{\alpha}_1 + \tilde{\alpha}_2 + \tilde{\alpha}_5 = \Delta\rho \\ \tilde{\alpha}_1\left(\tilde{u} - \tilde{a}n_x\right) + \tilde{\alpha}_2 u - \tilde{\alpha}_3 n_y + \tilde{\alpha}_5\left(\tilde{u} + \tilde{a}n_x\right) = \tilde{u}\Delta\rho + \tilde{\rho}\Delta u \\ \tilde{\alpha}_1\left(\tilde{v} - \tilde{a}n_y\right) + \tilde{\alpha}_2 v - \tilde{\alpha}_3 n_x + \tilde{\alpha}_5\left(\tilde{v} + \tilde{a}n_y\right) = \tilde{v}\Delta\rho + \tilde{\rho}\Delta v \\ \tilde{\alpha}_1\left(H - \left(\tilde{u}n_x + \tilde{v}n_y\right)\tilde{a}\right) + \frac{1}{2}\tilde{\alpha}_2\left(\tilde{u}^2 + \tilde{v}^2\right) + \tilde{\alpha}_3\left(-\tilde{u}n_y + \tilde{v}n_x\right) + \\ \tilde{\alpha}_5\left(H + \left(\tilde{u}n_x + \tilde{v}n_y\right)\tilde{a}\right) = \tilde{E}\Delta\rho + \tilde{\rho}\Delta E \\ \tilde{\alpha}_1\tilde{s} + \tilde{\alpha}_4 + \tilde{\alpha}_5\tilde{s} = \tilde{s}\Delta\rho + \tilde{\rho}\Delta s \end{array}\right\} \tag{D.1.7}$$

The second of equation (D.1.5), which is the jump in fluxes, is evaluated using a similar approach, which results in a lengthy set of equations to solve for the $\tilde{()}$ state. The result of solving these equations is a fully specified Roe state, as well as a definition of the numerical flux function for the fully coupled RANS-SA system.

The specification of the numerical flux first requires specifying the Roe state for the RANS-SA system as:

$$\begin{aligned}
\tilde{\rho} &= \sqrt{\rho^+ \rho^-} \\
\tilde{u} &= \frac{\sqrt{\rho^+} u^+ + \sqrt{\rho^-} u^-}{\sqrt{\rho^+} + \sqrt{\rho^-}} \\
\tilde{v} &= \frac{\sqrt{\rho^+} v^+ + \sqrt{\rho^-} v^-}{\sqrt{\rho^+} + \sqrt{\rho^-}} \\
\tilde{H} &= \frac{\sqrt{\rho^+} H^+ + \sqrt{\rho^-} H^-}{\sqrt{\rho^+} + \sqrt{\rho^-}} \\
\tilde{s} &= \frac{\sqrt{\rho^+} s^+ + \sqrt{\rho^-} s^-}{\sqrt{\rho^+} + \sqrt{\rho^-}} \\
\tilde{a} &= \sqrt{(\gamma - 1)\left(\tilde{H} - \frac{1}{2}\left(\tilde{u}^2 + \tilde{v}^2\right)\right)}
\end{aligned} \tag{D.1.8}$$

The numerical flux on the boundary can be written as:

$$\mathcal{H}_c\left(\mathbf{u}^+, \mathbf{u}^-\right) = \frac{1}{2}\left(\mathbf{F_c}^n\left(\mathbf{u}^+\right) + \mathbf{F_c}^n\left(\mathbf{u}^-\right) + \mathbf{D}\right) \tag{D.1.9}$$

where $\mathbf{D}$ is the dissipative component of the numerical flux, which is given as:

$$\begin{aligned}
\mathbf{D} &= \left\{ \begin{array}{c}
\left|\tilde{\lambda}_2\right|\left(\rho^+ - \rho^-\right) + \delta_1 \\
\left|\tilde{\lambda}_2\right|\left(\rho u^+ - \rho u^-\right) + \delta_1 \tilde{u} + \delta_2 n_x \\
\left|\tilde{\lambda}_2\right|\left(\rho v^+ - \rho v^-\right) + \delta_1 \tilde{v} + \delta_2 n_y \\
\left|\tilde{\lambda}_2\right|\left(E_t^+ - E_t^-\right) + \delta_1 \tilde{H} + \delta_2\left(\tilde{u} n_x + \tilde{v} n_y\right) \\
\left|\tilde{\lambda}_2\right|\left(\rho s^+ - \rho s^-\right) + \delta_1 \tilde{s}
\end{array} \right\} \\
\delta_1 &= -\left|\tilde{\lambda}_2\right| + \frac{1}{2}\left(\left|\tilde{\lambda}_1\right| + \left|\tilde{\lambda}_3\right|\right)\frac{\Delta P}{\tilde{a}^2} + \left(\left|\tilde{\lambda}_3\right| - \left|\tilde{\lambda}_1\right|\right)\frac{1}{2}\frac{\tilde{\rho}}{\tilde{a}}\left(n_x \Delta u + n_y \Delta v\right) \\
\delta_2 &= -\left|\tilde{\lambda}_2\right| + \frac{1}{2}\left(\left|\tilde{\lambda}_1\right| + \left|\tilde{\lambda}_3\right|\right)\tilde{\rho}\left(n_x \Delta u + n_y \Delta v\right) + \left(\left|\tilde{\lambda}_3\right| - \left|\tilde{\lambda}_1\right|\right)\frac{\Delta P}{\tilde{a}}
\end{aligned} \tag{D.1.10}$$

where the term $\Delta\left(\right)$ is given by:

$$\Delta\left(\right) = \left(\right)^+ - \left(\right)^- \tag{D.1.11}$$

Comparison of the dissipation terms in equation (D.1.10) with the standard formulas given in reference [50], shows that coupling the SA turbulence model equation to the RANS system does not change the numerical flux expressions for the mean flow equations. Contrasting

equation (D.1.10) with equation (2.6.3)(which is the standard approach) shows that for the decoupled numerical flux treatment the dissipation term of the numerical flux for the turbulence model equation involves only the velocity normal to the surface. However, one should note that in the fully coupled approach in equation (D.1.10), the sound speed and pressure influence the dissipation of the turbulence model discretization. Therefore, the formulation presented in this section is significantly different from the standard approach. Numerical experiments with convecting scalar quantities with a variable velocity field have shown that this Roe formulation yields smooth scalar quantities for smooth initial conditions, whereas the decoupled approach in equation (2.6.3) does not. Furthermore, numerical experiments with various combinations of numerical fluxes for the RANS and SA turbulence model equations, demonstrated that using the same numerical flux for both the RANS and turbulence model equations results in a significantly more robust solver.

References

[1] Jiri Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elseveir, Kidlington, Oxford, 2001.

[2] Charles Hirsch. *Numerical Computation of Internal and External Flows, Volume 1*. Wiley Series in Numerical Methods in Engineering. Wiley InterScience, Chichester, West Sussex, UK, 1984.

[3] Charles Hirsch. *Numerical Computation of Internal and External Flows, Volume 2*. Wiley Series in Numerical Methods in Engineering. Wiley InterScience, Chichester, West Sussex, UK, 1984.

[4] Antony Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the euler equations by finite volume methods using runge-kutta time-stepping schemes. In *Proceeding of the 14th Fluid and Plasma Dynamic Conference, Palo Alto, CA*, Jun 1981. AIAA Paper 1981-1259.

[5] Walter O. Valarezo and Dimitri J. Mavriplis. Navier-stokes applications to high-lift airfoil analysis. *Journal of Aircraft*, 32(3):618–624, May 1995.

[6] D. J. Mavriplis. Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes. *AIAA J.*, 28(2):213–221, 1990.

[7] Harvard Lomax, Thomas H. Pulliam, and David W. Zingg. *Fundamentals of Computational Fluid Dynamics*. Scientific Computation. Springer-Verlag, 2001.

[8] Dale A. Anderson, John C. Tannehill, and R. H. Petcher. *Computational Fluid Dynamics and Heat Transfer*. Hemisphere Publishing, New York, NY, 1984.

[9] James William Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Texts in Applied Mathematics. Springer, New York, NY, 1995.

[10] A Harten, P. D. Lax, and B Van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61, 1983.

[11] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method Volume 3 Fluid Dynamics*. Butterworth-Heinemann, Oxford, UK, fifth edition, 2000.

[12] T. J. R. Hughes and A. Brooks. Streamline upwind-Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Meth. Appl. Mech. Eng.*, 32:199–259, 1982.

[13] K. Laflin, O. Brodersen, M. Rakowitz, J. Vassberg, R. Wahls, and J. Morrison. Summary of data from the second AIAA CFD drag prediction workshop. AIAA Paper 2004-0555, Jan 2004.

[14] J. C. Vassberg, E. N. Tinoco, M. Mani, O. P. Brodersen, B. Eisfeld, R. A. Wahls, J. H. Morrison, T. Zickuhr, K. R. Laflin, and D. J. Mavriplis. Summary of the third AIAA CFD drag prediction workshop. AIAA Paper 2007-0260, Jan 2007.

[15] Nicholas K. Burgess, Cristian R. Nastase, and Dimitri J. Mavriplis. Efficient solution techniques for discontinuous galerkin discretizations of the navier-stokes equations on hybrid anisotropic meshes. In *Proceeding of the 48th Aerospace Sciences Meeting, Orlando, FL*, Jan 2010. AIAA Paper 2010-1448.

[16] Ralf Hartmann and Paul Houston. Adaptive discontinuous galerkin finite element methods for the compressible euler equations. *Journal of Computational Physics*, 183(2):508–532, Dec 2002.

[17] Li Wang and Dimitri J. Mavriplis. Adjoint-based h-p adaptive discontinuous galerkin methods for the 2d euler equations. *Journal of Computational Physics*, 228(20):7643–7661, Nov 2009.

[18] Karthik Mani. Error estimation and adaptation for functional outputs in time-dependent flow problems. *Journal of Computational Physics*, 229(2):415–440, Jan 2010.

[19] Karthik Mani. *Application of the Discrete Adjoint Method to Coupled Multidisciplinary Unsteady Flow Problems for Error Estimation and Optimization*. PhD thesis, University of Wyoming, May 2009.

[20] Todd A. Oliver. *A high-order, adaptive, discontinuous Galerkin finite element method for the Reynolds averaged Navier-Stokes equations*. PhD thesis, Massachusetts Institute of Technology, sept 2008.

[21] H. Atkins and C. W. Shu. Quadrature-free implementation of discontinuous Galerkin method for hyperbolic equations. *AIAA Journal*, 36(5):775–782, 1998.

[22] J. S. Hesthaven and T. Warburton. Nodal high-order methods on unstructured grids I. time-domain solution of Maxwell's equations. *J. Comput. Phys.*, 181(1):186–221, Sep 2002. ICASE Report No.01-6.

[23] Brendan S. Mascarenhas, Brian T. Helenbrook, and Harold L. Atkins. Application of p-multigrid to discontinuous galerkin formulations of the euler equations. *AIAA Journal*, 47(5):1200–1208, May 2009.

[24] Ioannis Toulopoulos and John A. Ekaterinaris. High-order discontinuous galerkin discretizations for computational aeroacoustics in complex domains. *AIAA Journal*, 44(3):502–511, Mar 2006.

[25] R.D. Nair, H-W. Choi, and H.M. Tufo. Computational aspects of scalable high-order discontinuous galerkin atmospheric dynamical core. *Computer and Fluids*, 38(2):309–319, Feb 2009.

[26] Yidong Xia, Hong Luo, Robert Norgaliev, and Chunpei Cai. A class of reconstructed discontinuous galerkin methods for the compressible flows on arbitrary grids. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting, Orlando FL*, Jan 2011. AIAA Paper 2011-199.

[27] Li Wang, Dimitri J. Mavriplis, and W. Kyle Anderson. Adjoint sensitivity formulation for discontinuous galerkin discretizations in unsteady inviscid flow problems. *AIAA Journal*, 48(12):2867–2883, Dec 2010.

[28] Francesco Bassi Andrea Crivellini and Stefano Rebay. High-order discontinuous galerkin discretization of transonic turbulent flows. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting, Orlando FL*, Jan 2009. AIAA Paper 2009-180.

[29] Tobias Leicht and Ralf Hartmann. Goal-oriented error estimation and *hp*-adaptive mesh refinement for aerodynamic flows. In *Proceeding of the 49th Aerospace Sciences Meeting, Orlando, FL*, Jan 2011. AIAA Paper 2011-212.

[30] Krzysztof J. Fidkowski, Todd A. Oliver, James Lu, and David L. Darmofal. p-multigrid solution of high-order discontinuous Galerkin discretizations of the compressible navier-stokes equations. *J. Comput. Phys.*, 207(1):92–113, Feb 2005.

[31] Cristian R. Nastase and Dimitri J. Mavriplis. High-order discontinuous Galerkin methods using an hp-multigrid approach. *J. Comput. Phys.*, 213(1):330–357, Mar 2006.

[32] Dimitri J. Mavriplis, Cristian R. Nastase, Li Wang, K. Shahbazi, and Nicholas Burgess. Progress in high-order discontinuous galerkin methods for aerospace applications. In *Proceedings of 47th Aerospace Sciences Meeting and Exhibit, Orlando FL*, 2009. AIAA Paper 2009–0601.

[33] Cristian R. Nastase and Dimitri J. Mavriplis. A parallel hp-multigrid solver for three-dimensional discontinuous galerkin discretizations of the euler equations. In *Proceedings of 45th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2007. AIAA Paper 2007–0512.

[34] Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous galerkin methods. In *Proceedings of 44th Aerospace Sciences Meeting and Exhibit, Reno NV*, Jan 2006. AIAA Paper 2006-112.

[35] M. Yang and Z. J. Wang. A parameter-free generalized moment limiter for high-order methods on unstructured grids. *Advances in Applied Mathematics and Mechanics*, 4(1):451–480, 2009.

[36] Sachin Premasuthan, Chunlei Liang, and Antony Jameson. A spectral difference method for viscous compressible flows with shocks. In *Proceeding of the 19th AIAA Computational Fluid Dynamics Conference, San Antonio, Texas*, Jun 2009. AIAA Paper 2009-3785.

[37] Garrett E. Barter and David L. Darmofal. Shock capturing with pde-based artificial viscosity for dgfem: Part i. formulation. *Journal of Computational Physics*, 229(5):1810–1827, Mar 2010.

[38] Sachin Premasuthan, Chunlei Liang, and Antony Jameson. Computation of flows with shocks using spectral difference scheme with artificial viscosity. In *Proceeding of the 48th Aerospace Sciences Meeting, Orlando, FL*, Jun 2010. AIAA Paper 2010-1449.

[39] P.E. Vincent and A. Jameson. Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists. *Mathematical Modeling of Natural Phenomena*, 8(3):97–140, May 2011.

[40] J. VonNeumann and R.D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232–238, Mar 1950.

[41] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows. *Le Recherche Aerospátiale*, 1:5–21, 1994.

[42] Francesco Bassi, Andrea Crivellini, Stefano Rebay, and Marco Savini. Discontinuous galerkin solution of the reynolds-averaged navier-stokes and k-ω turbulence model equations. *Computers and Fluids*, 34(4):507–540, May 2005.

[43] Ralf Hartmann, Joachim Held, and Tobias Leicht. Adjoint-based error estimation and adaptive mesh refinement for the rans and k-ω turbulence model equations. *Journal of Computational Physics*, 230(11):4268–4284, May 2011.

[44] Donald J. Estep. A posteriori error bounds and global error control for approximation of ordinary differential equations. *SIAM Journal of Numerical Analysis*, 32(1):1–48, 1995.

[45] David A. Venditti and David L. Darmofal. Anisotropic grid adaption for functional outputs:application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22–46, May 2003.

[46] Todd A. Oliver and David L. Darmofal. An unsteady adaptation algorithm for discontinuous galerkin discretizations of the rans equations. In *Proceeding of the 18th AIAA CFD Conference, Miami, FL*, Jun 2007. AIAA Paper 2007-3940.

[47] Nicholas K. Burgess and Dimitri J. Mavriplis. An *hp*-adaptive discontinuous galerkin solver for aerodynamic flows on mixed-element meshes. In *Proceeding of the 49th Aerospace Sciences Meeting, Orlando, FL*, Jan 2011. AIAA Paper 2011-490.

[48] Niles A. Pierce and Michael B. Giles. Adjoint and defect error bounding and correction for functional estimates. *Journal of Computational Physics*, 200(2):769–794, Nov 2004.

[49] Ralf Hartmann. Dual consistency analysis of discontinuous galerkin discretizations. *SIAM Journal of Numerical Analysis*, 45(6):2671–2696, 2007.

[50] F. Eleuterio Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Applied Mechanics. Springer-Verlag, New York, NY, 1999.

[51] S. F. Davis. Simplified second-order Godunov-type methods. *SIAM J. Sci. Statist. Comput.*, 9(3):445–473, 1988.

[52] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[53] D. Hänel and R. Schwane. An implicit flux-vector splitting scheme for the computation of viscous hypersonic flow. In *Proceeding of the 27th Aerospace Sciences Meeting, Reno, NV*, Jan 1989. AIAA Paper 1989-274.

[54] Khosro Shahbazi. *A Parallel High-Order Discontinuous Galerkin Solver For The Unsteady Incompressible Navier-Stokes Equations in Complex Geometries*. PhD thesis, University of Toronto, May 2007.

[55] Ralf Hartmann and Paul Houston. Symmetric interior penalty dg methods for the compressible navier-stokes equations i: Method formulation. *Internal Journal of Numerical Analysis and Modeling*, 3(1):1–20, 2006.

[56] Ralf Hartmann and Paul Houston. An optimal order interior penalty discontinuous galerkin discretization of the compressible navier-stokes equations. *Journal of Computational Physics*, 227(22):9670–9685, Nov 2008.

[57] Khosro Shahbazi, Dimitri J. Mavriplis, and Nicholas K. Burgess. Multigrid algorithms for high-order discontinuous galerkin discretizations of the compressible navier-stokes equations. *J. Comput. Phys.*, 228(21):7917–7940, Nov 2009.

[58] Pavel Solin, P. Segeth, and I.D. Zel. *High-Order Finite Element Methods*. Studies in Advanced Mathematics. Chapman and Hall, 2003.

[59] Li Wang. *Techniques For High-Order Adaptive Discontinuous Galerkin Discretizations in Fluid Dynamics*. PhD thesis, University of Wyoming, Jun 2009.

[60] George EM Karniadakis and Spencer Sherwin. *Spectral/hp Finite Element Methods for Computational Fluid Dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2005.

[61] D. A. Dunavant. Higher degree efficient symmetrical gaussian quadrature rules for the triangle. *Int. J. Numer. Meth. Eng.*, 21:1129–1148, 1985.

[62] D. A. Dunavant. Economical symmetrical quadrature rules for complete polynomials over a square domain. *Int. J. Numer. Meth. Eng.*, 21:1777–1784, 1985.

[63] B. Cockburn, S. Hou, and C.-W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Math. Comput.*, 54(545), 1990.

[64] Dimitri J. Mavriplis. Third drag prediction workshop using the nsu3d unstructured mesh solver. *AIAA Journal of Aircraft*, 45(3):750–761, Mar 2008.

[65] Eric J. Nielsen. Fun3d user's manual.

[66] Sherrie L. Krist, Robert T. Biedron, and Christopher L. Rumsey. Cfl3d user's manual (version 5.0). NASA Technical Report 1988-208444, NASA, Jun 1998.

[67] L. Krivodonova, J. Xin, J.F. Remacle, N. Chevaugeon, and J. Flaherty. Shock detection and limiting with discontinuous galerkin methods for hyperbolic conservation laws. *Appl. Numer. Math*, 48(3-4):323–338, Mar 2004.

[68] Nicholas K. Burgess and Dimitri J. Mavriplis. An *hp*-adaptive discontinuous galerkin method for the navier-stokes equations. In *Proceedings of the SIAM Conference on Computational Science and Engineering*, Reno, NV, Mar 2011. Society of Industrial and Applied Mathematics.

[69] Garret E. Barter and David L. Darmofal. Shock capturing with higher-order, pde-based artificial viscosity. In *Proceeding of the 18th Computational Fluid Dynamics Conference, Miami FL*, Jun 2007. AIAA Paper 2007-3823.

[70] Per-Olof Persson Ngoc Cuong Nguyen and Jaime Peraire. Rans solutions using high order discontinuous galerkin methods. In *Proceeding of the 45th Aerospace Sciences Meeting, Reno, NV*, Jan 2007. AIAA Paper 2007-914.

[71] Garrett E. Barter. *Shock Capturing with PDE-based artificial viscosity for an adaptive, higher-order discontinuous Galerkin finite element method*. PhD thesis, Massachusetts Institute of Technology, Jun 2008.

[72] Todd A. Oliver and David L. Darmofal. Analysis of dual consistency for discontinuous galerkin discretizations of source terms. *SIAM J. Numer. Anal.*, 47(1):3507, 2009.

[73] F. Bassi and S. Rebay. Numerical evaluation of two discontinuous galerkin methods for the compressible navier-stokes equations. *Int. J. Numer. Meth. in Fluids*, 40(1):197–207, Sept 2002.

[74] Todd A. Oliver and David L. Darmofal. Impact of turbulence model irregularity on high-order discretizations. In *Proceeding of the 47th Aerospace Sciences Meeting and Exhibit, Orlando FL*, Jan 2009. AIAA Paper 2009-953.

[75] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*. Springer, New York, NY, 2008.

[76] Cristian R. Nastase and Dimitri J. Mavriplis. High-order discontinuous Galerkin methods using a spectral multigrid approach. In *Proceedings of the 43rd Aerospace Sciences Meeting and Exhibit, Reno NV*, 2005. AIAA Paper 2005–1268.

[77] Li Wang and Dimitri J. Mavriplis. Implicit solution of the unsteady euler equations for high-order accurate discontinuous galerkin discretizations. *J. Comput. Phys.*, 225(2):1994–2015, Aug 2007.

[78] T. Haga and Z.J. Wang. Efficient solution techniques for high-order methods on 3-d anisotropic hybrid meshes. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting, Orlando FL*, Jan 2011. AIAA Paper 2011-45.

[79] Laslo T. Diosady and David L. Darmofal. Preconditioning methods for discontinuous galerkin solutions of the navier-stokes equations. *J. Comput. Phys.*, 228(1):3917–3935, Mar 2009.

[80] D.J. Mavriplis. An advancing front delaunay triangulation algorithm designed for robustness. *Journal of Computational Physics*, 117(1):90–101, Mar 1995.

[81] Dimitri J. Mavriplis. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. In *Proceeding of the 16th Computational Fluid Dynamics Conference, Orlando, FL*, Jun 2003. AIAA Paper 2003-3986.

[82] Per-Olof Persson and Jaime Peraire. Curved mesh generation and mesh refinement using lagrangian solid mechanics. In *Proceeding of the 47th Aerospace Sciences Meeting, Orlando, FL*, Jan 2009. AIAA Paper 2009-949.

[83] D.J. Mavriplis and V. Venkatakrishnan. A unified multigrid solver for the navier-stokes equations on mixed element meshes. ICASE Report 95-53, NASA ICASE, 1995.

[84] Ulrich Trottenberg, Cornelis Oosterlee, and Anton Schuller. *Multigrid*. Elsevier Academic Press, San Diego, CA, 2001.

[85] D. J. Mavriplis. Directional agglomeration multigrid techniques for high-Reynolds number viscous flow solvers. *AIAA Journal*, 37(10):1222–1230, Oct 1999.

[86] U. Trottenberg, A. Schuller, and C. Oosterlee. *Multigrid*. Academic Press, London, UK, 2000.

[87] Dimitri J. Mavriplis. An assessment of linear versus non-linear multigrid methods for unstructured mesh solvers. *J. Comput. Phys.*, 175:302–325, 2002.

[88] Krysztof J. Fidkowski and David L. Darmofal. Development of a higher-order solver for aerodynamic applications. In *Proceedings of the 42nd Aerospace Sciences Meeting and Exhibit, Reno NV*, 2004. AIAA Paper 2004–0436.

[89] Dimitri J. Mavriplis. Multigrid techniques for unstructured meshes. In *VKI Lecture Series VKI-LS 1995-02*, Mar 1995.

[90] Lloyn N. Trefethen and David III Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[91] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 1997.

[92] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2003.

[93] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 1996.

[94] B. Cockburn and C.-W. Shu. The runge-kutta discontinuous galerkin method for conservation laws v: multidimensional systems. *SIAM Journal of Numerical Analysis*, 141(1):199–224, 1998.

[95] Konstantinos Kontzialis and John A. Ekaterinaris. Limiters for discontinuous galerkin discretizations for mixed type meshes with p-type adaptivity. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting, Orlando FL*, Jan 2011. AIAA Paper 2011-297.

[96] Haiyang Gao and Z.J. Wang. A residual-based procedure for hp-adaptation on 2d hybrid meshes. In *Proceeding of the 49th Aerospace Sciences Meeting, Orlando, FL*, Jan 2011. AIAA Paper 2011-492.

[97] Eric J. Nielsen, James Lu, Mike A. Park, and David L. Darmofal. An implicit, exact dual adjoint solution method for turbulent flows on unstructured grid. *Computers and Fluids*, 33(9):1131–1155, Nov 2004.

[98] D.E. Coles and E.A. Hirst. Computation of turbulent boundary layers. In *AFOSR-IFP-Stanford Conference*, 1969. Vol. II Stanford University.

[99] B. Cockburn and C.-W. Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *SIAM J. Sci. Comput.*, 16(3):173–261, 2001.

[100] A. Klöckner, T. Warburton, and J.S. Hesthaven. Viscous shock capturing in a time-explicit discontinuous galerkin method. *Mathematical Modeling of Natural Phenomena*, 6(3):57–83, Jan 2011.

[101] John D. Anderson. *Modern Compressible Flow with Historical Perspective*. McGraw-Hill Series in Aeronautical and Aerospace Engineering. McGraw-Hill, 2003.

[102] H. W. Liepmann and A. Roshko. *Elements of Gas Dynamics*. Dover, 2001.

[103] Brian A. Lockwood and Dimitri J. Mavriplis. Parameter sensitivity for hypersonic viscous flow using a discrete adjoint approach. In *Proceeding of the 48th Aerospace Sciences Meeting, Reno, NV*, Jan 2010. AIAA Paper 2010-447.

[104] Willima J. Rider, Dana A. Knoll, and Gordon L. Olson. A multigrid newton-krylov method for multi-material equilibrium radiation diffusion. *Journal of Computational Physics*, 152(1):164–191, Jun 1999.

[105] Wataru Yamazaki and Dimitri J. Mavriplis. Derivative-enhanced variable fidelity surrogate modeling for aerodynamic functions. In *Proceeding of the 49th Aerospace Sciences Meeting, Reno, NV*, Jan 2011. AIAA Paper 2011-1172.

[106] Markus Rumpfkeil, Wataru Yamazaki, and Dimitri J. Mavriplis. A dynamic sampling method for kriging and cokriging surrogate models. In *Proceeding of the 49th Aerospace Sciences Meeting, Reno, NV*, Jan 2011. AIAA Paper 2011-883.

[107] Brian Lockwood, Markus Rumpfkeil, Wataru Yamazaki, and Dimitri J. Mavriplis. Uncertainty quantification in viscous hypersonic flows using gradient information and surrogate modeling. In *Proceeding of the 49th Aerospace Sciences Meeting, Reno, NV*, Jan 2011. AIAA Paper 2011-885.

[108] Zhong-Hua Han, Stefan Görtz, and Ralf Zimmermann. On improving efficiency and accuracy of variable-fidelity surrogate modeling in aero-data for loads context. In *Proceeding of the CEAS 2009 European Air and Space Conference, Manchester, UK*, Oct 2009.

[109] Serhat Hosder and Robert W. Walters. Non-intrusive polynomial chaos methods for uncertainty quantification in fluid dynamics. In *Proceeding of the 48th Aerospace Sciences Meeting, Reno, NV*, Jan 2010. AIAA Paper 2010-129.

[110] Markus Rumpfkeil. Personal Communication, 2011.

[111] Francesca Iacono, Georg May, and Z.J. Wang. Relaxation techniques for high-order discretizations of steady compressible inviscid flows. In *Proceedings of the 40th AIAA Fluid Dynamics Conference,Chicago IL*, Jun 2010. AIAA Paper 2010-4991.

[112] Venkateswaran Sankaran, Andrew Wissink, Anubhav Datta, Jayanarayanan Sitaraman, Buvana Jayaraman, Mark Potsdam, Aaron Katz, Sean Kamkar, Beatrice Roget, Dimitri Mavriplis, Wei-Bin Chen Hossein Saberi, Wayne Johnson, and Roger Strawn. Overview of the helios version 2.0 computational platform for rotorcraft simulations. In *Proceeding of the 49th Aerospace Sciences Meeting, Reno, NV*, Jan 2011. AIAA Paper 2011-1105.

[113] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier-stokes equations. *J. Comput. Phys.*, 131(2):267–279, Mar 1997.

[114] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463, 1998.