A Combined Entropy and Output-Based Approach for Mesh Refinement and Error Estimation Using Finite-Element Formulations

by

Kevin T. Doetsch

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2022

"Education never ends, Watson.
It is a series of lessons, with the greatest for the last."

–Sir Arthur Conan Doyle
His Last Bow

Kevin T. Doetsch

kevintd@umich.edu

ORCID iD: 0000-0002-6705-1705

For my grandfather, William Stiens

ACKNOWLEDGMENTS

First and foremost, I would like to extend my sincere gratitude to my advisor, Professor Krzysztof Fidkowski, for all of his support and guidance during this research. When I was first introduced to Chris, I was just an entry-level engineer who barely knew anything about graduate research and what it takes to earn a Ph.D. All I knew was I wanted to learn how CFD works and write a CFD code. Despite not being a typical student fresh out of school who could focus exclusively on the research, he took a chance on me and I am eternally grateful for that chance. What I admire most about Chris is neither his expansive knowledge of CFD nor his impressive publication list. What I admire most is his dedication to teaching. In my opinion, the best professors are those who put their teaching responsibilities ahead of their research. Chris is one of those professors. I also admire Chris for his incredible patience. It has been a challenge balancing my graduate responsibilities from afar while working as a full-time engineer in St. Louis and raising a family. I greatly appreciate having an advisor who is patient and understands that my responsibilities outside of the research sometimes have to come first. I also appreciate his willingness to work with me from a distance and give up one hour of his night every Monday to discuss my progress and provide excellent guidance and direction. Having Chris as a mentor and advisor has been one of the greatest privileges of my life and I will forever be grateful.

I would also like to thank my committee members, Dr. Behzad R. Ahrabi of The Boeing Company, Professor Jesse Capecelatro, and Professor Venkat Raman, for their time and effort spent supporting my research. I especially appreciate their willingness to be on my committee despite not being able to work or meet with them face-to-face, since I am rarely on campus due to my job in St. Louis. The feedback and support they have provided are much appreciated.

While I was only on campus for a short amount of time, I met fellow students who made that time truly memorable. I especially want to thank my research group members: Steve Kast, Johann Dahm, Kyle Ding, Yukiko Shimizu, Devina Sanjaya, Guodong Chen, Gustavo Halila, Gary Collins, Vivek Ojha, and Matteo Franciolini. They were all so welcoming and made me feel at home, even though I had not been a student for many years. I am thankful for the times we shared and all of the knowledge I learned from them. I also want to thank Alexander Vazsonyi, Candice Kaplan, Kaelan Hansson, and Mike Holloway for spending all those hours in the library studying for preliminary exams with me. I would not have passed without them. I look back with incredible fondness on the short time I was able to spend on campus, thanks to all of the aforementioned people I met there. I cannot thank them enough for that.

I have been fortunate to come across many talented and impressive engineers during the almost ten years I have worked in the aerospace industry. During that time I have met so many engineers who challenge me to do better and set my expectations higher. I want to especially thank Mori Mani for introducing me to Professor Fidkowski and encouraging me to pursue my dream of obtaining a Ph.D. I also want to thank Andrew Cary and Andy Dorgan. After we met for lunch during my summer internship in 2012, when I barely knew anything about CFD, I remember thinking that I wanted to be like them. Andrew and Andy personify technical excellence and I strive every day to meet the high standards they exhibit. Finally, I want to thank my industry mentor, Dave Stookesberry. When I first started working with him he was always patient with me and went the extra mile to make sure the tasks assigned to me were completed professionally and efficiently. Even today, I can always count on him to answer my calls and provide incredibly helpful guidance and advice. I would not be where I am right now without his support.

I would like to thank my parents, Tom and Sandy, and my sisters, Nicole and Kristine, for all of the support and encouragement they have given me over the years. I also want to thank my 18-month-old son, Lincoln. While your presence has not always been the most conducive to quality research, you give me more happiness and joy than anything in my life ever has. I am so lucky to be your dad.

Finally, I want to thank my wife, Kate. One of Kate's passions is showing young girls that being an engineer is not only an amazing career, it is also really cool. She is an inspiration to so many girls through her outreach efforts in the Society of Women Engineers, and she is also an inspiration to me. She inspires me to be a better engineer, a better father, a better husband, and a better man. I am so fortunate that she is an integral part of my life, as I could not have done this without her.

This work was supported by The Boeing Company Learning Together Program and the Rackham Merit Fellowship Program.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

ABSTRACT

As aerospace design programs rely more on computational fluid dynamics (CFD) in all phases of the engineering design process, there is a growing interest in obtaining high-fidelity solutions using high-order finite-element methods. Such aspirations require considerable research on error estimation and mesh adaptation techniques to obtain high-quality results with lower computation costs. Adaptive mesh refinement approaches generally rely on an error indicator to drive the adaptation. Feature-based error indicators are based on directly computable solution characteristics and provide a relatively inexpensive way to target areas of the mesh for refinement. However, these indicators are not robust as they assume local error depends solely on the local resolution of the mesh. Particularly for hyperbolic problems, such as convection-dominated flows, these methods may inadequately refine areas essential for the accurate prediction of integrated forces and moments.

Alternatively, goal-oriented error indicators specifically target areas of the computational domain that are critical to the prediction of a specific output. These methods are particularly effective at accounting for propagation effects through the use of adjoint solutions that provide the sensitivity of the output to local residuals. This dissertation presents a strategy for mesh adaptation driven by a new error indicator that combines two previously-investigated adjoint-based indicators: one based on a user-specified engineering output such as drag or lift coefficient, and the other based on entropy variables. The novelty of this approach lies not only in its construction but also in its implementation with multiple types of high-order, finite-element discretizations.

Using the entropy-variable indicator to adapt a mesh is computationally advantageous since it does not require the solution of an auxiliary adjoint equation, which is costly for unsteady problems. However, the entropy-variable indicator targets any region of the domain where spurious entropy is generated, regardless of whether or not this region affects an engineering output of interest. Conversely, an indicator computed from an engineering output generally targets only those regions important for the chosen output, though it is more computationally taxing because of the required adjoint solution. Approximations in the adjoint calculation reduce this cost, at the expense of indicator accuracy. In combining these indicators, the objective is to maintain the low cost of approximate adjoint solutions while achieving improved indicator accuracy from the entropy variables. To further reduce the computational cost, various modifications to the combined approach are also analyzed. In this dissertation, we demonstrate the benefits of the combined approach for not only steady-state simulations, where only spatial error is considered, but also unsteady simulations where both spatial and temporal errors must be considered. Various applications of fluid governing equations with multiple types of mesh adaptation strategies are considered to show how robust this novel combined approach currently is.

Much of the work presented in this dissertation is in the context of the Discontinuous Galkerin (DG) finite-element discretization. To not only further demonstrate the robustness of the novel combined approach, but to also investigate a different finite-element discretization, a code was written as part of this dissertation that incorporates the Streamline-Upwind Petrov-Galerkin (SUPG) method. Specifically for moderate orders of accuracy, SUPG requires fewer degrees of freedom than DG for comparable accuracy. Comparisons of output error estimates using the combined approach for both SUPG and DG methods are included in this work to further demonstrate the potential of this new error estimation approach.

CHAPTER 1

Introduction

1.1 Motivation

Over the last several decades, the rapid advance of computational fluid dynamics (CFD) technology has completely changed the aerospace design process. Thanks to the advancement in computing power and the growing availability of high-performance computing (HPC) clusters and fast-paced graphical units (GPUs), CFD has become an integral part of the engineering design process in a variety of fields and disciplines. In the context of aerospace engineering, CFD has become a critical piece of the design process and has significantly reduced the need for wind tunnel tests [1–5]. Due to the lack of wind tunnel availability and model complexity, relying primarily on wind tunnel tests can be incredibly expensive for novel aircraft programs. CFD, on the other hand, is comparatively inexpensive and turnaround time is much shorter. CFD is also advantageous since it allows for extensive flow visualization. The ability to visualize the entire computational domain, from surface contours to flowfield cuts, is particularly useful in all aspects of the aerospace design process. As a result, CFD is now an invaluable tool to lower programmatic costs and enable novel designs in the aerospace industry and beyond. However, as computational resources become more vast and available, the reliance on efficient and accurate CFD in the design process becomes ever greater. This section will focus on the current state of CFD and where potential bottlenecks in the CFD workflow exist.

1.1.1 Use of CFD in Aerospace Development

The origins of CFD can be traced back to World War II, when scientists at Los Alamos National Laboratory first used numerical tools to model the violent flow created from the atomic bomb [6]. Generally, fluids can be treated as a continuum whose dynamics are described by conservation laws that can be discretized in the form of partial differential equations (PDEs). When these equations are formulated to a small, finite volume, they can be discretized as integral equations. In CFD, these equations are discretized on a computation domain/grid, which leads to finite solution approximations. These integrated formulations can be categorized as finite-volume and finite-element. With the ability to solve complex flow equations and analyze physical phenomena that may be too costly or difficult in real-life experiments, CFD methods are particularly useful in aerospace development and design.

During the 1960s, the underlying principles of fluid dynamics and the formulation of the governing equations were established. Major aerospace companies such as Boeing, NASA, McDonnell, Lockheed, and Douglas developed panel method codes to determine optimal wing shapes [6]. However, it was not until the 1970s-1990s that significant advancements in CFD methods and computing resources allowed for significant growth in the use of CFD for aerospace development programs [7].

One significant example was the use CFD played in the unlocking of nacelle/wing interference drag for the 737-300 program during the 1970s-1980s [8–10]. From 1955 until 1975, wind tunnel experiments indicated that nacelles could not be placed too close to the wing without causing excessive drag. These tests limited where the nacelle could be positioned relative to the wing. Thanks to studies using CFD, Boeing was able to unlock the secret of nacelle/wing interference drag. For the 737-300 program in the early 1980s, Boeing modified the installation of the CFM56-3 engine by removing the pylon and moving the accessory gearbox from the bottom to the side of the engine. This gave the nacelle its distinct, flat-bottom shape. All of these changes were thanks to advancements in CFD. Without these advancements, there would not have been a successful 737-300 program.

As CFD capabilities increased and computational resources became more prevalent in the 1990s and 2000s, CFD's role in aerospace development further increased, especially in the field of multi-disciplinary analysis and optimization (MDAO). MDAO leverages CFD technology to optimally design solutions to complex flow problems quickly and accurately. MDAO using CFD was used extensively in the Boeing 787 Dreamliner development program [11]. Almost all aspects of the aerodynamic design of the 787, especially in the design of the nacelles, wing, and forebody, relied on an unprecedented level of CFD. As the timeline of aerospace programs development programs shrinks and the applications get more complex, the need to have reliable and cost-efficient CFD will only increase.

1.1.2 Current Research Areas of CFD Methods

The NASA CFD Vision 2030 study [12] outlined several research areas of CFD that are critical for the tool's future use in the aerospace design process. These areas include HPC computing, the development of more accurate numerical algorithms and physical models, mesh adaptation, and multi-disciplinary analysis and optimization (MDAO). Along with these findings, the study identified several roadblocks that limit the use of CFD in the design process.

One of the most significant roadblocks is adaptive grid techniques and error estimation for complex flows and geometries. The study admits that past research efforts have shown that current error estimation techniques, in conjunction with mesh adaptation, offer tremendous potential in terms of reducing the computational cost and providing high-fidelity CFD solutions. However, these techniques are not widely used in industry due to software and computational complexity, complex geometries, and insufficient error estimation capabilities. These conclusions and observations were further expanded upon in the context of unstructured grid adaptation [13]. This work includes an exhaustive list of the various error estimation techniques and ideas that have been investigated using unstructured mesh adaptation over the last few decades. The study categorizes these error indicators into three categories: feature-based, goal-oriented, and norm-oriented. Each family of error indicators has its strengths and weaknesses, which will be further elaborated on in Subsection 1.3.2. No one type of error indicator has been universally identified as possessing the advantages of all three types of error indicator families. A major focus of this work is finding an indicator that leverages the advantages present in multiple error indicator types.

Another roadblock identified in NASA CFD Vision 2030 study is the lack of algorithmic improvements to enable future advancements in computing resources, mesh adaptation, and MDAO. According to the study, CFD algorithm development has slowed down in recent years due to a lack of investment in these areas. Future algorithmic advancements are necessary for enabling the growing HPC and GPU landscape, as well as for enabling breakthroughs in error estimation and mesh adaptation. In particular, the study notes that discretizations must be robust, scalable, and able to solve highly complex simulations. Based on current industry-standard methods obtained from the most recent AIAA drag prediction workshop (DPW) and high-lift prediction workshop (HLPW), current CFD methods still produce inconsistent convergence behavior and scalar output predictions for complex problems [14, 15].

1.2 Dissertation Objectives

The primary goal of this dissertation is to present a novel approach for error estimation and mesh adaptation that can produce reliable predictions of outputs such as lift and drag coefficient for various applications. This new approach combines two previously studied error estimation approaches, one goal-oriented and one norm-oriented, so that the advantages of each approach can be fully exploited while limiting their respective disadvantages. Since this new approach involves combining two existing approaches, the approach, and its subsequent variations explored in this work, are often referred to as the *combined approach*. In the context of this new combined approach, the objectives of this dissertation are listed in the following subsections.

1.2.1 Demonstrate Effectiveness of the Combined Approach

As mentioned in Section 1.1, the need for robust and cost-effective error estimation techniques is essential for the continued use of CFD for aerospace development programs. Existing error estimation techniques have inherent characteristics that limit their general use in the aerospace industry. The construction of a suitable error indicator represents one of the weakest points of most mesh adaptation strategies [13, 16]. By demonstrating the performance of this error indicator for various applications in this work, we can show the potential benefits in accuracy and computational expense it offers.

We accomplish this by testing the combined approach with various mesh adaptation strategies for numerous governing equations in both steady-state and unsteady contexts. Unsteady simulations are of particular interest, as they increase the complexity and computation expense of any error estimation and mesh adaptation approach. Outputs such as lift and drag coefficient are extracted from these adaptive simulations and directly compared to identical outputs obtained using the existing error estimation and mesh adaptation strategies. Sample meshes are also highlighted to show where the combined approach error indicators focus the mesh refinement compared to the existing approaches. A major goal of this dissertation will be to show that the novel combined approach developed in this work has strong potential as an error estimation technique for mesh adaptation in CFD moving forward.

1.2.2 Compare Combined Approach Across Multiple High-Order Methods

Another major focus of this dissertation is the evaluation of the combined approach in the context of high-order discretizations. Currently, second-order accurate methods are the industry standard. However, with the already acknowledged advancements in computing resources, MDAO, and mesh adaptation, aerospace design programs will rely even heavily on CFD to reduce costs associated with physical testing. This will require higher-fidelity CFD simulations. One way of achieving this is through the use of high-order methods. Though high-order methods have been studied extensively in recent decades [17], these methods are still a work in progress and no universally preferred high-order discretization has been identified.

The novel combined approach developed for this work was initially limited to one type of high-order discretization. To demonstrate the strength of this approach, an entirely new code was written that incorporates a different high-order discretization. Besides the obvious benefit of being able to compare the performance of the combined approach between these two high-order discretizations, this also allowed for the added benefit of comparisons among existing error estimation and mesh adaptation approaches. Multiple applications are analyzed by both high-order discretizations in this work to make valuable comparisons regarding output-error convergence. By demonstrating the effectiveness of this novel combined approach with a new code written especially for this work, we demonstrate that the conclusions drawn based on the first objective outlined in the previous subsection are not limited to one type of algorithm.

1.3 Background

In Section 1.2, we introduced the concept of the new combined approach for error estimation and mesh adaptation that combines two existing error estimation approaches that have previously been developed. We also mentioned how this combined approach was tested in multiple high-order discretizations, including one that formed the foundation for a code written specifically for this research. In this section, we review three key ideas pertinent to the combined approach in the context of this work. We begin with a review of high-order CFD methods, specifically focusing on those used in this research. Special attention is also paid to why the additional high-order method was chosen to be included in this dissertation. Next is a thorough review of the two distinct error estimation approaches that make up the combined approach. The section ends with a brief review of the topic of mesh adaptation and which methods were chosen to be used in conjunction with the combined approach for this work. This section is only meant to provide a brief review of the aforementioned concepts, as subsequent chapters in this dissertation will provide more extensive details.

1.3.1 High-Order CFD Methods

As has been the case over the last few decades, high-order discretizations of aerodynamics continue to generate significant interest due to the potential to achieve higher accuracy at a reduced cost [18–23]. The order of accuracy of a discretization is measured based on the solution error convergence rate, r, at which the error of the solution, $\mathcal{E}$, decreases as the characteristic mesh size, h, is refined. This means that for a particular mesh size, the order of the discretization is $\mathcal{O}\left(h^r\right)$. High-order CFD methods possess an order of accuracy greater than two, $r > 2$ [17]. Any consistent CFD method will converge towards the exact solution as $h \to 0$, but high-order methods do so at higher rates. Low-order methods must use a highly refined mesh to achieve the same numerical accuracy as high-order methods, whereas high-order methods can achieve the same accuracy with a much coarser mesh.

It is important to note that there is a trade-off between the order of accuracy and mesh resolution. For a given mesh, low-order methods will be comparatively less expensive. A difficult question to answer is whether those low-order methods produce acceptable results for the particular application. Generally, high-order methods are more efficient for CFD applications with high-fidelity calculations. Assuming a convergence rate, $r = p + 1$, so that the error $\mathcal{E} \approx \mathcal{O}\left(h^{p+1}\right)$, where p is the solution approximation order, the time to solution T_f can be expressed as [24]

$$\log T_f = d\left(-\frac{1}{p+1}\log\mathcal{E} + a\log\left(p+1\right)\right) - \log F + \text{constant}. \tag{1.1}$$

In Equation 1.1, d is the spatial dimension, F is the computational speed, and a is the algorithm complexity. For high-fidelity calculations $\mathcal{E} << 1$, which means the computational time T depends exponentially on the ratio $d/(p+1)$. For high-order methods where $p > 1$, this conclusion means the computational time will be much less compared to low-order methods.

Currently, second-order finite-volume methods are the dominant methodologies in CFD applications. This is primarily due to their robustness, high flexibility, and ease of implementation, thanks to considerable research done during the 1970s to the 1990s [7, 17]. However, the emerging need for high-fidelity requirements in some applications has emphasized the importance of high-order methods. For example, high-order methods resolve small features for much greater distances without considerable dissipation than second-order methods [25]. Another example is the tracking of vortices emanating from wing tips or helicopter blades. In these cases, second-order finite-volume methods require very refined meshes to resolve these features over a large distance [21]. Extending finite-volume schemes to high-order is difficult due to the required extended stencils where degrees of freedom now must be coupled to non-neighboring elements. This leads to a wide range of challenges, from increased memory requirements for implicit methods and parallelization to the lack of stable iterative algorithms [24].

Finite-element discretizations, on the other hand, use compact discretization stencils that only require immediately adjacent elements. This means that the introduction of high-order degrees of freedom can occur locally, which allows for high-order discretizations in both space and time to have an established path [21]. This also makes them very adaptable for use in parallelization. Many of the distinct advantages finite-element methods have over finite-volume methods are summarized in [17]. For smooth problems, high-order finite-element methods produce lower errors at a given computational cost than second-order finite-volume schemes. However, for non-smooth problems, such as flows involving shocks, high-order finite element schemes do not always optimally converge. However, when coupled with solution-based mesh adaptation, high-order methods can outperform low-order methods by focusing mesh refinement on these non-smooth regions.

In finite-element discretizations, a set of basis functions, usually polynomials, is defined locally for each element. This leads to a functional representation of the solution that has local support, resulting in the aforementioned compact stencil. The basis functions can be set up so that element interface continuity is maintained throughout the computational domain. This is called the continuous Galerkin (CG) method. The CG method works very well for diffusion-dominated problems. For convection-dominated problems, however, such as compressible fluid flow problems, it is unstable.

Alternatively, the basis functions can be arranged so that the continuity is not maintained at the element interfaces. This is called the discontinuous Galerkin (DG) method. First introduced in 1973 in the context of the transport of neutrons [26], the DG method has been the subject of extensive research for use in CFD [18–20, 27–34]. Since the solution in a DG discretization is discontinuous at the element interfaces, coupling is done through fluxes on the element boundaries. The discontinuous approximation leads to extra degrees of freedom on the element interfaces, which leads to an increase in computational cost. However, the discontinuous nature of DG simplifies certain aspects of error estimation, mesh adaptation, and preconditioning. Unlike CG, DG is stable for solving convection-dominated problems [35, 36] because the numerical fluxes across element interfaces are computed via a traditional finite-volume (approximate) Riemann solver. Note that for convection-diffusion problems, a stabilization term must be added to diffusive operators when using DG [37–39].

Due to the extensive amount of research and familiarity we have with DG, a significant focus of this work will utilize the DG discretization, the complete details of which are found in Section 2.2. However, because of the additional computation cost associated with DG due to the large memory footprint, recent focus in the high-order CFD community has been placed on reducing the computational cost of high-order finite-element discretizations such as DG. One such approach is using the hybridized discontinuous Galerkin (HDG) method [40–45], where the total number of globally-coupled degrees of freedom is reduced by decoupling element solution approximations and stitching them back together using weak flux continuity enforcement. This is accomplished by adding face unknowns that become the only globally-coupled degrees of freedom in the entire system. In DG, the number of globally-coupled degrees of freedom scales as p^d, where p is the solution approximation order and d is the number of spatial dimensions. In HDG, the exponent reduces by one to p^{d-1} since the unknowns are defined on a space of one lower dimension (faces instead of elements). This means that HDG methods can be cheaper and require less memory compared to DG. Another similar approach to HDG is the embedded discontinuous Galerkin (EDG) method [45, 46], where the approximation space of the face unknowns is continuous. This approach further reduces the number of globally-coupled degrees of freedom.

Another set of approaches that reduce the number of globally-coupled degrees of freedom compared to DG are stabilized continuous Galerkin (CG) finite-element methods. In continuous finite element schemes, nodes on edges, vertices, and faces are shared between adjacent elements, which leads to a continuous solution over the domain. This means that for moderate formal orders of accuracy, stabilized finite element methods require far fewer degrees of freedom than DG methods on the same mesh [47, 48][1]. However, as previously mentioned, CG methods are unstable for convection-dominated problems. To overcome the inherent stability issues, stabilization terms can be formulated for convection operators. One such approach that been studied over the last four decades is the Streamline-Upwind Petrov-Galerkin (SUPG) method [21, 49–59]. In particular, recent focus on the SUPG method has produced promising results. For example, SUPG results using linear ($p = 1$) basis functions (which have the same nominal order of accuracy as second-order finite volume methods) have produced far more accurate velocity profiles for turbulent flow over the NACA 0012 airfoil using the same computational mesh compared to both finite volume and DG methods [25, 48]. The ability to produce high-fidelity solutions using far less computational expense is why additional focus on the SUPG method was included in this work, the details of which are located in Section 2.4 of Chapter 2.

1.3.2 Error Estimation

Error estimation increases the robustness of the CFD by quantifying the error in the solution. Accurate error estimates are critical for all high-fidelity CFD applications. Without an accompanying error estimate, it is difficult to quantify the accuracy of a CFD simulation. This is an essential piece of information that is necessary to assess the reliability of the prediction. In the context of mesh adaptation, error estimates help identify where adaptation can be productively applied. Since there are many sources of error in CFD simulations, it is important to quantify which types of error estimates are helpful, so we can learn from those estimates to improve the accuracy of the CFD.

Discretization error is the difference between the exact solution of the governing equations and the discrete solution that comes from the discretized form of the equations. Since the exact solutions of not known for flow governing equations, the discretization error must be estimated. Error estimation approaches for finite element discretizations generally fall into two categories: *a priori*, where the error is measured using the exact solution, and *a posteriori*, where the error is measured using the computed solution. Development of numerical methods and information such as a method's convergence rate and stability often rely on *a priori* error estimates, since they can be used to assess the asymptotic behavior of the method [60]. Unfortunately, these methods rely on parameters not always known *a priori*. This means *a priori* error estimates cannot be used to quantify actual errors for a given computation domain. In contrast, *a posteriori* error estimates rely on the difference between the discrete numerical solution and a finer representation of the solution, usually through a reconstruction process. We can define the residual error as how well the approximate solution satisfies the exact equations, which can be approximated by evaluating

[1]This does not mean that continuous methods are universally more accurate for a given number of degrees of freedom, as the error magnitudes could be different at the same order.

the residual on the finer space [61]. Since *a posteriori* error estimates are based on the discrete solution, which is known, they are very attractive in the context of mesh adaptation.

Solution-based error estimates have long been used for mesh adaptation in Computational Fluid Dynamics (CFD) to obtain accurate solutions for problems that exhibit a wide range of spatial length scales [19, 62–67]. These methods generally rely on an error indicator to drive the adaptation. Numerous examples in the literature detail extensive analysis of various indicators and their relative performance, robustness, and accuracy.

An error indicator can be classified as heuristic if there is no link between the indicator and the numerical error. One class of heuristic error indicators relies on solution features to drive the mesh adaptation [46, 68–70]. Using these indicators for error estimation is commonly referred to as feature-based error estimation. These features can be gradients, curvatures, or any other characteristics of the solution that be computed directly. While these indicators are often cheap to evaluate, they are not always robust since they assume local error only depends on the local resolution. For elliptic or nearly elliptic problems, such as low-speed flows, feature-based error indicators work well. However, for hyperbolic problems, such as high-speed, convection-dominated flows, errors can propagate by the convection-dominated nature of the system. In problems with desired scalar outputs of interest (such as lift and drag), these methods can overrefine areas of the domain that do not affect the output and not refine areas that are important [64, 71, 72].

To remedy the inherent issues with feature-based error estimation, error estimation methods have been developed that use an error indicator based on a user-specified engineering scalar output [64, 73, 74]. These types of error indicators are referred to as goal-oriented indicators [13]. These methods are particularly effective since the indicator specifically targets areas of the computational domain that are critical to the prediction of the desired output [19, 64, 75, 76]. They also are effective in accounting for propagation effects intrinsic to hyperbolic problems [77], through the use of output-specific adjoint solutions, which provide the sensitivity of the output to local residuals. The downside is that they can be computationally expensive since they require the implementation of a transpose linear solver and they involve quantities that must be computed in a refined-space computational domain. Another downside is that they are not always effective due to a suboptimal approximation of the adjoint, especially in the presence of adjoint singularities such as on a leading-edge stagnation streamline and near an airfoil trailing edge [78]. These singularities contribute to noise in the error estimate, which leads to areas of unnecessary over-refinement of the computational mesh.

Unsteady simulations pose significantly higher computational costs for output-based adjoint approaches, due to the unsteady adjoint equation. Current research has increasingly focused on extending output-based adaption to unsteady problems, using a variety of adaptation mechanics [79–85]. The issue with output-based approaches is that the solution of the unsteady adjoint equation is expensive, as it must be marched backward in time. This leads to significantly higher computational costs when compared to steady-state simulations. It also requires saving the primal state at each time step or checkpoint, further adding to the computational cost.

A different error estimation approach leverages entropy variables instead of a separate adjoint computation [86, 87]. This type of error indicator is commonly referred to as a norm-oriented error indicator [13]. Using entropy variables is less computationally expensive, since these variables can be computed from a direct variable transformation of the conservative state. Entropy-variable indicators are equivalent to an adjoint solution corresponding to an output that measures the net entropy production in the computational domain. Consequently, they can be used in the output-based adjoint error estimation framework. The drawback of this approach is similar to that of feature-based error indicators, in that entropy-variable indicators target all regions of the domain where *spurious entropy* is produced, e.g. shocks or 3D vortices. It is important to note that the entropy-based indicator does not target all regions of the domain with large entropy production. It differs from feature-based methods in that it only targets regions of the domain where there is *spurious entropy production*. Regardless of that subtlety, this approach is not as globally discriminating as the output-based approach.

Both the output-based adjoint error estimation approach and the entropy-variable approach have unique advantages and disadvantages. However, the entropy-variable approach shows great promise due to the relative computational savings compared to the output-based adjoint approach. Unfortunately, it is not as robust, as it targets all regions of the domain where spurious entropy is generated. As stated in Section 1.2, a major objective of this research is to more robustly extend the entropy-variable adaptive approach for these cases by combining it with indicators obtained from output-based adjoints. The goal is the calculation of a combined approach that encompasses the strengths of both the output-based adjoint and the entropy-variable approach, while limiting the disadvantages of each.

1.3.3 Mesh Adaptation

The previous subsection outlined multiple ways of obtaining error estimates. Regardless of which approach is used, the error estimates can be used in conjunction with mesh adaptation to modify the solution space to equidistribute, and ultimately decrease, the error. As shown in Figure 1.1, the mesh adaptation framework [88] is a feedback loop where first, the discretized governing equations are solved in a space defined by a current mesh and solution approximation order. After estimating the numerical error, the computational space is either modified or enriched based on that estimate. A new solution is obtained in this new space. This process repeats until an error or cost tolerance is met.

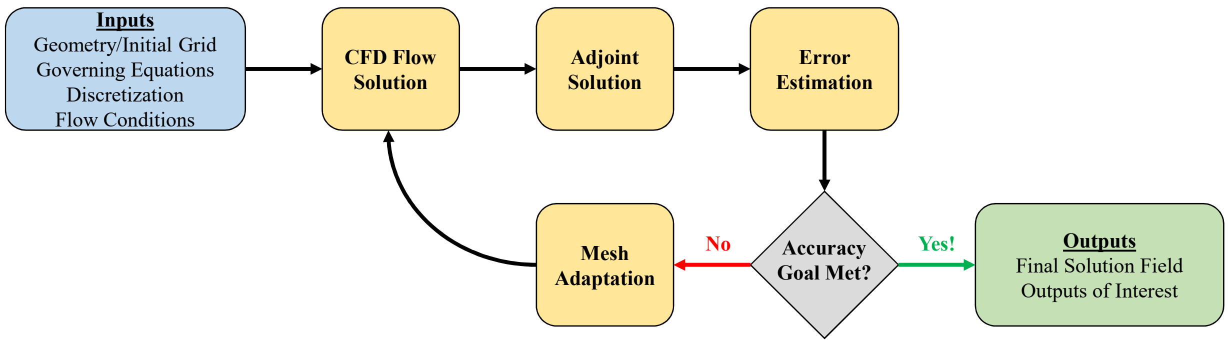


Figure 1.1: Diagram of the mesh adaptation process.

For high-order finite-element methods, the solution space depends on the local mesh resolution, h, and the solution approximation order, p. In this context, we can define three commonly used approaches for mesh adaptation: p-adaptation, h-adaptation, and hp-adaptation. In h-adaptation, the local refinement is accomplished by subdividing the elements into smaller elements, while keeping the solution approximation order p constant. In p-adaptation, the local solution approximation order of the elements' shape functions is modified while the mesh is held constant. The combination of both p-adaptation and h-adaptation is referred to as hp-adaptation.

Comparing h-adaptation and p-adaptation directly, p-adaptation is more effective for error reduction for equal degrees of freedom in smooth problems. Since the solution approximation can easily be increased, p-adaptation is also much easier to implement. However, this approach requires a reasonable starting mesh and has difficulties in regions with sharp gradients in either the solution or geometry, particularly areas with high anisotropy. Resolving these areas with h-adaptation is comparatively more effective, at the expense of needing to implement a potentially complex mesh adaptation algorithm. The most effective approach is hp-refinement where p-refinement is used in areas where the solution is smooth, while h-refinement is used near singularities and anisotropic areas [21, 89]. If used properly, hp-refinement can yield exponential convergence with respect to degrees of freedom [24]. The challenge in hp-refinement is developing an algorithm that determines where to use the appropriate type of refinement.

The focus of this work is h-adaptation only, where the solution approximation order, p, is always held constant. This does not take advantage of the cost savings available using hp-adaptation, but it significantly reduces the overall complexity of the mesh adaptation process. As mesh adaptation by itself is not the primary focus of this dissertation, this decision is sound. What is a primary focus is how the various error estimates outlined in Subsection 1.3.2 perform using h-adaptation. One h-adaptation approach utilized in this work is a fixed-fraction hanging-node strategy where a fixed fraction of elements that have the largest error indicators is marked for uniform refinement using a series of hanging nodes. Assuming no mesh coarsening is involved, this approach is very simple to implement and useful for analyzing different error estimates. This isotropic mesh adaptation strategy has demonstrated success in the past [90], but is not very effective for flow problems with strong directional features like wakes, shocks, and boundary layers. In cases such as these, the optimal mesh resolution can vary by multiple orders of magnitude in the orthogonal directions. Isotropic mesh adaptation approaches can only resolve these areas by adding more elements, which is not very efficient.

To resolve the inherent issues in the aforementioned hanging node adaptation strategy where refinement occurs locally, mesh strategies that incorporate global re-meshing are a popular alternative. In these methods, a new mesh, typically unstructured, is generated for the entire computational domain. Two different global re-meshing strategies that allow for mesh refinement and coarsening, as well as redistribution of the mesh degrees of freedom to allow for better productivity, are used in this work. The two methods differ regarding how they detect anisotropy. Unfortunately, no anisotropic information can be directly obtained from the local error indicators. That information must come from another source. In the first approach, anisotropic information is determined by sizing the elements using *a priori* rate estimates and using the Hessian of a scalar computed from the state to determine the anisotropic information of each element. In the second approach, the mesh is optimized to minimize the error using a prescribed cost. This is a variation of mesh optimization via error sampling and synthesis (MOESS) [91, 92] and is built upon the continuous mesh framework. Further details regarding all of the mesh adaptation strategies used in this dissertation are found in Chapter 5.

1.4 Overview of Dissertation

The main objective of this dissertation is to demonstrate the effectiveness of a novel combined approach for error estimation and mesh adaptation in a variety of fluid governing equations. This approach has the potential to improve upon existing error estimation techniques by reducing the computational cost to obtain even more accurate, high-fidelity solutions for many fluid flow applications. In addition, the approach is robust as it has been implemented and tested using multiple high-order finite element discretizations.

1.4.1 Major Contributions

The main contributions of this dissertation are as follows:

- Developed a novel approach for error estimation and mesh adaptation that combines error indicators obtained using output-based adjoints with indicators obtained using entropy variables.

- Formulated variations of the combined approach that reduce computational expense and improve output error estimates.
- Implemented the combined approach, and its various derivatives, into an existing high-order, discontinuous Galerkin (DG) finite element code to quantify the performance of the approach for both steady-state and unsteady simulations with multiple sets of governing fluid flow equations.
- Wrote a high-order, continuous finite-element code in ANSI C to implement the Streamline-Upwind Petrov-Galerkin (SUPG) discretization.
- Implemented the combined approach, as well the output-based adjoint and entropy-variable error estimation approaches, into the SUPG code to directly compare output error predictions between SUPG and DG for all error estimation techniques.
- Implemented multiple global re-meshing strategies that incorporate anisotropic elements and high-order elements into the SUPG code to test the effectiveness of the combined approach and compared directly to results generated with similar capabilities using the existing DG framework.

1.4.2 Dissertation Outline

The remaining chapters in this dissertation provide more extensive details and findings on the aforementioned topics. The outline of this work is as follows:

- Chapter 2 begins with a review of the governing equations of interest. Next is a thorough review of both the DG and SUPG methods. For each method we review the solution approximation methodology, the weak form discretization, and solution techniques implemented for this dissertation. This chapter also includes a section on the extension of DG for unsteady problems.
- Chapter 3 introduces the concept of an adjoint and how it can be used for output error estimation. This leads to a thorough review of the output-based adjoint error estimation approach for steady-state problems, as well as an overview of how entropy variables can be used instead of an output-based adjoint for error estimation. The chapter closes with a review of the various combined output-based adjoint and entropy-variable approaches created for this dissertation.

- Chapter 4 details how the error estimation approaches outlined in Chapter 3 change when implemented for unsteady problems. The chapter begins with an overview of the unsteady output-based adjoint, followed by a discussion of how error estimation changes for unsteady problems when both spatial and temporal errors are considered, in the context of the output-based adjoint approach. Subsequent discussions regarding the use of the entropy-variable error estimation approach and the various combined approaches for unsteady problems conclude the chapter.

- Chapter 5 is an overview of the three mesh adaptation approaches used in this work. In addition to providing a detailed overview of how each method is implemented, special attention is given to the benefits of each approach and which applications each approach is best suited for.

- Chapter 6 presents the various findings from this research, which are separated into three distinct sections. The first section explores the benefits of the combined approaches for a large variety of steady-state problems, using the DG discretization. The second section compares results from the SUPG code written for this work to similar results obtained using DG, in the context of inviscid, steady-state simulations. Particular attention is given to how the novel combined approach performs in SUPG relative to DG. The last section explores the performance of the combined approach for an unsteady problem when using DG.

- Chapter 7 provides an overview of the findings from the present work, as well as a discussion on future work directions.

CHAPTER 2

Discretization and Equations

The primary focus of this chapter is a thorough explanation of two finite element discretizations adopted for solving the fluid flow governing equations in this work. We begin with a review of the governing equations used for both the Discontinuous Galerkin (DG) finite-element discretization and the Streamline-Upwind Petrov-Galerkin (SUPG) continuous finite element discretization. The next section is a comprehensive overview of the DG method and how it is applied to a general conservation law. Special attention is also paid to the solution techniques used to solve problems using DG. This chapter also includes a brief review of how the DG discretization is applied to unsteady problems. The chapter closes with a thorough review of the SUPG method applied to the Euler equations and how the implementation and solution techniques differ from those of the DG discretization.

2.1 Governing Equations

In this section, three different sets of governing equations are discussed in detail. These equations represent the target applications for the discretizations and mesh adaptation strategies discussed in this work. First, we present an overview of the compressible Navier-Stokes (NS) equations. Next, there is a brief review of the Euler equations for the flow of a perfectly inviscid gas. While the Euler equations are a simplified form of the NS equations (in which viscous terms are ignored) a more detailed review of the Euler equations is included in this work since the SUPG discretization in Section 2.4 is based only on the Euler equations. Finally, there is a description of the Reynolds-averaged Navier-Stokes (RANS) equations, the time-averaged form of the NS equations.

2.1.1 Compressible Navier-Stokes Equations

When using the Discontinuous Galerkin discretization, this work considers solutions of the compressible Navier-Stokes (NS) equations of gas dynamics that govern the flow of a viscous, Newtonian fluid. The NS equations come directly from Newton's second law of motion, conservation of momentum, and include both convective and diffusive effects. Combined with the conservation of mass and energy, the NS equations represent conservation laws with $d + 2$ conserved states. Written in terms of the conservation of mass, momentum, and energy, the NS equations are

$$\begin{aligned} \partial_t \rho + \partial_i(\rho u_i) &= 0, \\ \partial_t(\rho u_i) + \partial_i(\rho u_i u_j + p\delta_{ij}) - \partial_i \tau_{ij} &= 0, \\ \partial_t(\rho E) + \partial_i(\rho u_i H) - \partial_i(\tau_{ij} u_j - q_i) &= 0, \end{aligned} \tag{2.1}$$

where the variables are defined as

ρ: density

p: pressure

u_i: i^{th}component of the velocity $\vec{v}$

E: specific (per unit mass) total energy

H: specific (per unit mass) total enthalpy

q_i: i^{th}component of the heat flux $\vec{q}$

τ_{ij}: viscous stress tensor

δ_{ij}: Kronecker delta function : $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$

The specific total energy E can be defined as the internal energy of the fluid, e, combined with the fluid's kinetic energy:

$$E = e + \frac{1}{2}|\vec{v}|^2. \tag{2.2}$$

Similarly, the specific total enthalpy is the internal enthalpy of the fluid, h, combined with the kinetic energy of the fluid:

$$H = h + \frac{1}{2}|\vec{v}|^2. \tag{2.3}$$

The internal energy and enthalpy equations come from the definition of a calorically perfect gas, which states that

$$\begin{aligned} e &= c_v T, \\ h &= c_p T = e + \frac{p}{\rho}, \\ \gamma &= \frac{c_p}{c_v}, \end{aligned} \tag{2.4}$$

where T is the temperature of the fluid and c_p and c_v are the specific heat capacity at a constant pressure and constant volume, respectively. They are related by the constant specific heat ratio γ. The heat flux q_i is defined using Fourier's law:

$$q_i = -\kappa_T \partial_i T, \tag{2.5}$$

where κ_T is the thermal conductivity. The viscous stress tensor is defined as

$$\tau_{ij} = 2\mu\epsilon_{ij} + \lambda\delta_{ij}\delta_k u_k, \tag{2.6}$$

where μ is the dynamic viscosity, λ is the bulk viscosity, and ϵ_{ij} is the strain rate tensor. The strain rate tensor describes the deformation of the fluid in all dimensions, which can be written as

$$\epsilon_{ij} = \frac{1}{2}\left(\partial_i u_j + \partial_j u_i\right). \tag{2.7}$$

The following expressions are derived from the ideal gas law:

$$\begin{aligned} p &= \rho R T, \\ p &= (\gamma - 1)(\rho e), \\ \rho e &= \rho E - \frac{1}{2}\rho|\vec{v}|^2, \\ H &= \frac{p}{\rho} + E. \end{aligned} \tag{2.8}$$

where R is the specific gas constant.

Equation 2.1 can be written in a compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} + \frac{\partial \mathbf{G}_i}{\partial x_i} = \mathbf{0}, \tag{2.9}$$

where i indexes the spatial dimension, and the state and flux vectors are

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix}, \quad \vec{\mathbf{F}} = \begin{bmatrix} \rho u_i \\ \rho u_i u_j + p\delta_{ij} \\ \rho u_i H \end{bmatrix}, \quad \vec{\mathbf{G}} = \begin{bmatrix} 0 \\ \tau_{ij} \\ \tau_{ij} u_j - q_i \end{bmatrix}. \tag{2.10}$$

Relevant properties for air with temperature-dependent dynamics viscosity are defined as follows:

$$\begin{aligned} \text{specific heat ratio}&: \gamma = 1.4, \\ \text{specific heat}&: c_v = \frac{R}{\gamma - 1}, \;\; c_p = \gamma c_v = \frac{\gamma R}{\gamma - 1}, \\ \text{specific gas constant}&: R = 287 \text{ J/(kg} \cdot \text{K)}, \\ \text{Prandtl number}&: Pr = \frac{\mu c_p}{\kappa_T} = 0.71, \\ \text{thermal conductivity}&: \kappa_T = \frac{\gamma R}{(\gamma - 1)} \cdot \frac{\mu}{Pr}, \\ \text{bulk viscosity}&: \lambda = -\frac{2}{3}\mu, \\ \text{kinematic viscosity}&: \nu = \frac{\mu}{\rho}. \end{aligned}$$

The compressible Navier-Stokes equations are primarily used to simulate laminar flows of viscous fluids, but can also be used to simulate low Reynolds number turbulent viscous flows. The Reynolds number is a non-dimensional quantity used to measure the effect of viscosity and is useful in gauging if a flow will be laminar or turbulent. The Reynolds number can be expressed mathematically as

$$Re = \frac{\rho |\vec{v}| L}{\mu}, \tag{2.11}$$

where L is the reference length scale (e.g. airfoil chord). Another useful non-dimensional parameter is the Mach number, M. The Mach number is the ratio between the fluid speed and the speed of sound, which mathematically can be expressed as

$$M = \frac{|\vec{v}|}{a}, \tag{2.12}$$

where the speed of sound is related to the static temperature via $a = \sqrt{\gamma R T}$.

2.1.2 Euler Equations

When using the Streamline-Upwind Petrov-Galerkin discretization in this work (see Section 2.4), we only consider solutions of the compressible Euler equations of gas dynamics that govern the flow of a perfect inviscid gas. These equations are obtained from the negligible-viscosity assumption at the high Reynolds number limit of the compressible Navier-Stokes equations. Any viscous-related terms in the Navier-Stokes equations from Subsection 2.1.1 are ignored in this approach. Written in terms of the conservation of mass, momentum, and energy as in Equation 2.1, the Euler equations are

$$\begin{aligned} \partial_t \rho + \partial_j(\rho u_i) &= 0, \\ \partial_t(\rho u_i) + \partial_i(\rho u_i u_j + p\delta_{ij}) &= 0, \\ \partial_t(\rho E) + \partial_i(\rho u_i H) &= 0, \end{aligned} \tag{2.13}$$

where all of the variables match those found in Subsection 2.1.1. The Euler equations can be written in a compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} = \mathbf{0}, \tag{2.14}$$

where i indexes the spatial dimension. In vector form, the state and flux vectors become

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix}, \quad \vec{\mathbf{F}} = \begin{bmatrix} \rho u_i \\ \rho u_i u_j + p\delta_{ij} \\ \rho u_i H \end{bmatrix}. \tag{2.15}$$

The divergence of the inviscid flux vector, $\frac{\partial \mathbf{F}_i}{\partial x_i}$, can be rewritten in terms of the unknown state values as

$$\frac{\partial \mathbf{F}_i}{\partial x_i} = \frac{\partial \mathbf{F}_i}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial x_i} = \mathbf{A}_i \frac{\partial \mathbf{u}}{\partial x_i}, \tag{2.16}$$

where $\mathbf{A}_i$ is the inviscid flux Jacobian. Rewriting the inviscid flux vector divergence in terms of the inviscid flux Jacobian is helpful, as it is directly related to the stabilization term used in SUPG. This will be further elaborated on in Subsection 2.4.3.

2.1.3 Reynolds-Averaged Compressible Navier-Stokes Equations

For flows at a high Reynolds number, turbulence is often present. As the Reynolds number increases, the viscosity often decreases, which means that viscous effects on the fluid are comparatively smaller compared to inertial effects. When this happens, small perturbations cannot be damped out easily and may be amplified. This directly contributes to unstable flow and the transition from laminar to turbulent fluid flow. In turbulent flow, unsteady vortices of varying length scales are present in the domain. To obtain an accurate computational solution, these length scales must be sufficiently captured. Since the length scales have such extreme variation throughout the domain, it is often not practical to resolve them using a mesh. Instead, turbulence modeling is used to account for the effect of the unresolved scales on the resolved variables.

Reynolds-averaged Navier-Stokes (RANS) equations model turbulence by breaking up primitive flow variables into mean values and their fluctuating parts. All flow quantities in the Navier-Stokes equations shown in Equation 2.1 can be broken up into time-averaged values and time-fluctuating parts. For example, a decomposition of a flow variable $U(\vec{x}, t)$ into the time-averaged value and fluctuating components is expressed mathematically as

$$\begin{aligned} U(\vec{x}, t) &= \bar{U}(\vec{x}) + U'(\vec{x}, t), \\ \bar{U}(\vec{x}) &= \lim_{\Delta t \to \infty} \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} U(\vec{x}, t) dt, \end{aligned} \tag{2.17}$$

where t_0 is the initial time and $t_0 + \Delta t$ is the final time. For compressible flows, we use a density-weighted time average of the flow quantities, which for $U(\vec{x}, t)$ is defined as

$$\widetilde{U}(\vec{x}) = \frac{1}{\bar{\rho}} \lim_{\Delta t \to \infty} \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} \rho\left(\vec{x}, t\right) U(\vec{x}, t) dt = \frac{\overline{\rho Q}}{\bar{\rho}}. \tag{2.18}$$

The RANS equations are derived by substituting this decomposition into the Navier-Stokes equations from Equation 2.1. After applying various approximations based on turbulence properties, the only term that remains that cannot be closed is the Reynolds stress term, $\overline{\rho u_i u_j}$. This term comes directly from the momentum conservation equations, but it is unique in that it resembles a stress term. The only way to completely solve the RANS equations is to model this term with a turbulence model. There are many turbulence models to choose from, but this work focuses on the Spalart-Allmaras (SA) turbulence model with a negative viscosity modification [93]. Under these assumptions, the governing equations in Equation 2.1 can be written in terms of time-averaged states as follows[1]:

$$\begin{aligned} \partial_t \rho + \partial_i (\rho u_i) &= 0, \\ \partial_t (\rho u_i) + \partial_i (\rho u_i u_j + p \delta_{ij}) - \partial_i \tau_{ij}^{\text{RANS}} &= 0, \\ \partial_t (\rho E) + \partial_i (\rho u_i H) - \partial_i (\tau_{ij}^{\text{RANS}} u_j - q_i^{\text{RANS}}) &= 0, \end{aligned} \tag{2.19}$$

[1]The conserved states are all time-averaged. For the sake of simplicity, the notation $\bar{\cdot}$ is removed.

where τ_{ij}^{RANS} is the effective viscous stress tensor that accounts for both viscous and Reynolds stresses and q_i^{RANS} is the effective heat flux that includes the turbulent transport effects. These two terms can be written as

$$\tau_{ij}^{\text{RANS}} = 2(\mu + \mu_t)\epsilon_{ij}, \qquad \epsilon_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i) - \frac{1}{3}\partial_k u_k \delta_{ij}. \tag{2.20}$$

$$q_i^{\text{RANS}} = -\left(\kappa_T^{\text{RANS}}\right)\partial_j T = -c_p\left(\frac{\mu}{\text{Pr}}\frac{\mu_t}{\text{Pr}_t}\right)\partial_j T, \tag{2.21}$$

where Pr_t is the turbulent Prandtl number and μ_t is the eddy viscosity by analogy to molecular diffusion. Using the Boussinesq approximation, the Reynolds stress tensor can be rewritten as [94]

$$\rho u_i u_j = -2\mu_t\left(\frac{1}{2}(\partial_i u_j + \partial_j u_i) - \frac{2}{3}\partial_k u_k \delta_{ij}\right) + \frac{2}{3}\rho k \delta_{ij}, \tag{2.22}$$

where $\rho k = (1/2)\rho u_i u_j$ is the turbulent kinetic energy per unit volume. In the negative SA model, the eddy viscosity is defined as

$$\mu_t = \begin{cases} \rho\tilde{\nu} f_{v1}(\chi) & \tilde{\nu} \geq 0 \\ 0 & \tilde{\nu} < 0, \end{cases}, \qquad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad \chi \equiv \frac{\tilde{\nu}}{\nu}, \tag{2.23}$$

where $\nu = \mu/\rho$ is the kinematic viscosity. The above expression ensures non-negative viscosity, which can occur in the original SA model if the mesh is under-resolved. The above equation introduces the turbulent kinetic viscosity, $\tilde{\nu}$. This new parameter requires an additional equation added to the governing equations to close the system:

$$\partial_t(\rho\tilde{\nu}) + \partial_i(\rho u_i \tilde{\nu}) = \frac{\rho}{\sigma}\partial_i((\nu + \nu')\partial_i\tilde{\nu}) + \frac{c_{b2}\rho}{\sigma}\partial_i\tilde{\nu}\partial_i\tilde{\nu} + P - D, \tag{2.24}$$

where P and D are the production and destruction terms for $\tilde{\nu}$. The production term controls how much additional turbulent kinetic viscosity is added to the simulation:

$$P = \begin{cases} c_{b1}\tilde{S}\rho\tilde{v} & \chi \geq 0 \\ c_{b1}S\rho\tilde{v} & \chi < 0, \end{cases}, \qquad \tilde{S} = \begin{cases} S + \bar{S} & \bar{S} \geq -c_{v2}S \\ S + \frac{S\left(c_{v2}^2 S + c_{v3}\bar{S}\right)}{(c_{v3} - 2c_{v2})S - \bar{S}} & \bar{S} < -c_{v2}S, \end{cases}, \tag{2.25}$$

where $S = \sqrt{2\Omega_{ij}\Omega_{ij}}$ is the vorticity tensor magnitude. The parameter $\bar{S}$ is the modified vorticity magnitude and is defined as

$$\bar{S} = \frac{\tilde{v} f_{v2}}{\kappa^2 d_w^2}, \qquad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}, \tag{2.26}$$

where d_w is the distance to the closest wall. The destruction term, D, counteracts the production term by limiting the increase of turbulent kinetic viscosity and is defined as

$$D = \begin{cases} c_{w1} f_w \frac{\rho \tilde{v}^2}{d_w^2} & \chi \geq 0 \\ -c_{w1} \frac{\rho \tilde{v}^2}{d_w^2} & \chi < 0, \end{cases} \tag{2.27}$$

where

$$f_w = g \left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right)^{1/6}, \qquad g = r + c_{w2}(r^6 - r), \qquad r = \frac{\tilde{v}}{\tilde{S} \kappa^2 d_w^2}. \tag{2.28}$$

The additional viscosity term ν', unique to the negative SA model, is defined as

$$\nu' = \begin{cases} \nu \chi & \chi \geq 0 \\ \nu f_n(\chi) & \chi < 0, \end{cases}, \qquad f_n(\chi) = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}. \tag{2.29}$$

The SA model constants are defined as

$$\begin{aligned} c_{b1} &= 0.1355 & c_{w2} &= 0.3 \\ \sigma &= \frac{2}{3} & c_{w3} &= 2.0 \\ c_{b2} &= 0.622 & c_{v1} &= 7.1 \\ \kappa &= 0.41 & c_{v2} &= 0.7 \\ c_{w1} &= \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma} & c_{v3} &= 0.9 \\ Pr_t &= 0.9 & c_{n1} &= 16.0. \end{aligned}$$

The SA working variable, $\tilde{\nu}$, is often many orders of magnitude smaller than most other components of the conservation equations. To prevent convergence issues for nonlinear solvers, this term is often scaled to resemble the order of other states in the following fashion,

$$\tilde{\nu}_s = \frac{\tilde{\nu}}{c_s}, \tag{2.30}$$

where c_s is the scaling factor on the order of $\sqrt{Re}$. The additional conservation equation is also divided by c_s,

$$\partial_t(\rho\tilde{\nu}_s) + \partial_i(\rho u_i\tilde{\nu}_s) = \frac{\rho}{\sigma}\partial_i((\nu+\nu')\partial_i\tilde{\nu}_s) + c_s\frac{c_{b2}\rho}{\sigma}\partial_i\tilde{\nu}_s\partial_i\tilde{\nu}_s + \frac{P-D}{c_s}. \tag{2.31}$$

By scaling this equation, the residual of the SA equation becomes closer to that of other equations.

2.2 Discontinuous Galerkin Discretization

The primary focus of this section is a thorough description of the Discontinuous Galerkin discretization of the governing equations. We begin with a review of general conservation laws that are used for the derivation of the general DG discretization. What follows is a detailed review of the DG discretization as applied to the general set of conservation laws. A review of the solution technique used to solve the discrete system for steady-state simulations is also included. All of the material presented in this section is based on previous and ongoing work from the computational fluid dynamics research group at the University of Michigan [24, 95–101].

2.2.1 General Conservation Equations

A general unsteady conservation law, made up of convective, diffusive, and source terms, can be written in conservation form as a partial differential equation (PDE),

$$\partial_t\mathbf{u} + \nabla\cdot\vec{\mathbf{H}}\left(\mathbf{u},\nabla\mathbf{u}\right) + \mathbf{S}\left(\mathbf{u},\nabla\mathbf{u}\right) = \mathbf{0}, \tag{2.32}$$

where $\mathbf{u}(\vec{x},t) \in \mathbb{R}^s$ is the state vector, $\vec{\mathbf{H}} \in \mathbb{R}^{d\times s}$ is the total flux, and $\mathbf{S} \in \mathbb{R}^s$ is the source term. The variable s refers to the rank of the state vector, while d refers to the total number of spatial dimensions. For steady problems, $\partial_t\mathbf{u} = \mathbf{0}$. The total flux can be broken up between convective and diffusive terms,

$$\vec{\mathbf{H}}\left(\mathbf{u},\nabla\mathbf{u}\right) = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u},\nabla\mathbf{u}), \tag{2.33}$$

where $\vec{\mathbf{F}} \in \mathbb{R}^{d\times s}$ is the convective flux and $\vec{\mathbf{G}} \in \mathbb{R}^{d\times s}$ is the diffusive flux. The diffusive, or viscous, flux depends linearly on the gradient of the state vector,

$$\vec{\mathbf{G}}(\mathbf{u},\nabla\mathbf{u}) = -\underline{\mathbf{K}}(\mathbf{u})\nabla\mathbf{u}, \tag{2.34}$$

where $\underline{\mathbf{K}} \in \mathbb{R}^{d^2\times s}$ is the viscous diffusivity tensor.

2.2.2 Solution Approximation

One of the hallmarks of the DG method is approximating the state **u** in a functional form using linear combinations of basis functions on each element. In this work, the basis functions are all polynomial expressions, where continuity among elements is not mandated. Mathematically, this can be expressed by denoting T_h as the set of N_e elements in a non-overlapping tessellation of the domain Ω, as illustrated in Figure 2.1. The state on element Ω_e is approximated as

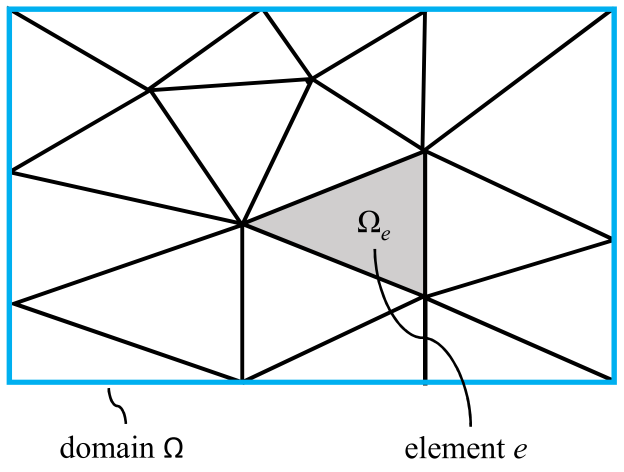


Figure 2.1: Partition of domain into various triangular elements.

$$\mathbf{u}_h(\vec{x})\big|_{\Omega_e} = \sum_{n=1}^{N_p} \mathbf{U}_{en}\phi_{en}(\vec{x}), \tag{2.35}$$

where

N_p = number of basis functions required for an order p approximation of the element

p = spatial order of basis functions for each element

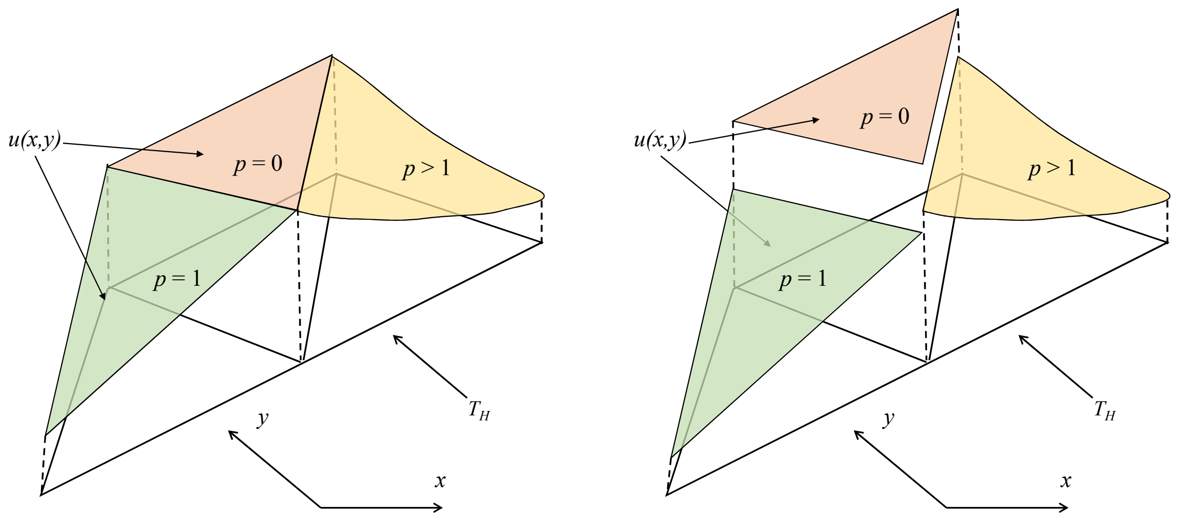
$\phi_{en}(\vec{x})$ = n^{th}order p basis function on element e (zero on every other element)

$\mathbf{U}_{en}$ = vector of s coefficients on n^{th} basis function on element e

Formally, the solution $\mathbf{u}_h$ exists in the approximation space $\boldsymbol{\mathcal{V}}_h$, $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where

$$\mathcal{V}_h = \{u \in L_2(\Omega) : u\big|_{\Omega_e} \in \mathcal{P}^p \;\forall \Omega_e \in T_h\}, \tag{2.36}$$

and $\mathcal{P}^p$ denotes polynomials of order p. Discontinuous finite element methods differ from continuous finite-element methods in that the solution is not continuous across the entire domain. The approximation space $\boldsymbol{\mathcal{V}}_h$ is only continuous inside each element, which means the solution is generally discontinuous across all elements. This leads to more degrees of freedom for a given mesh compared to continuous finite-element methods that ultimately need to be accounted for, adding to the overall computational expense. However, DG provides convective stability due to the Riemann solvers used to compute convective fluxes across element interfaces and significantly simplifies hanging node refinement and local order enrichment. A direct comparison between the approximation space in DG versus its continuous counterpart (CG) for a state component u is shown in Figure 2.2. While the order p is not consistent among the elements in the aforementioned figure, the order is uniform across all elements in this work.



(a) Continuous Galerkin (CG)

(b) Discontinuous Galerkin (DG)

Figure 2.2: Solution approximations using continuous and discontinuous basis functions. Despite the discontinuous solution in DG, the inter-element flux is single valued as in the finite-volume method.

2.2.3 Weak Form

Now that the approximation space has been properly defined via Equation 2.35, the DG method searches to find a solution, $\mathbf{u}_h$, that satisfies the *weak form* of the governing equations. The weak form ensures that the strong form residual of the PDEs is orthogonal with respect to the test functions, $\mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h$. This weak form of Equation 2.32 is found by multiplying the PDE by the tests functions, integrating by parts, and coupling elements with numerical fluxes. The semilinear weak form can be expressed mathematically as

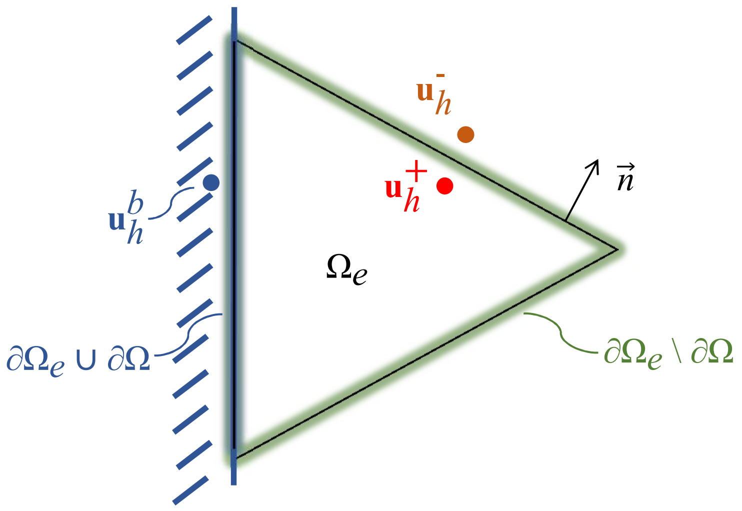
$$\sum_{e=1}^{N_e} \int_{\Omega_e} \mathbf{v}_h^T \partial_t \mathbf{u}_h d\Omega + \mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = 0, \qquad \forall\, \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h, \tag{2.37}$$

where $\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h)$ is the semilinear form of the spatial component of the residual. Contributions from the inviscid flux, the viscous flux, and the source terms all factor into this term, so that we can write

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = \sum_{e=1}^{N_e} \left[\mathcal{R}_{h,I}(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) + \mathcal{R}_{h,V}(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) + \mathcal{R}_{h,S}(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) \right], \tag{2.38}$$

where the subscripts I, V, and S denote the inviscid flux, viscous flux, and source term, respectively.

To further delve into the details of the discretization of each of the three aforementioned residual terms, elemental notation for the DG discretization of one element must be established. Figure 2.3 shows a schematic of the DG elemental notation that is used in the rest of this section. For the element boundary, $\partial\Omega_e$, the state from the element interior is denoted by $(\cdot)^+$, while the state on the neighboring element is denoted by $(\cdot)^-$. The normal vector points away from the element, from $+$ to $-$. If the element borders a boundary, then the boundary state is noted as $(\cdot)^b$. The boundary state $\mathbf{u}^b$ is a projection of the interior state and the boundary condition data, $\mathbf{u}_h^b = \mathbf{u}_h^b(\mathbf{u}_h^+, \mathrm{BC})$.



$\mathbf{u}_h^-$
$\mathbf{u}_h^+$
$\vec{n}$
$\mathbf{u}_h^b$
Ω_e
$\partial\Omega_e \cup \partial\Omega$
$\partial\Omega_e \setminus \partial\Omega$

Figure 2.3: Elemental notation for a DG discretization on a triangle.

Focusing exclusively on the inviscid flux term, applying integration by parts produces an inviscid residual which can be written as,

$$\mathcal{R}_{h,I}(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) = -\int_{\Omega_e} \partial_i \mathbf{v}_h^T \mathbf{F}_i d\Omega + \int_{\partial\Omega_e \backslash \partial\Omega} \mathbf{v}_h^{+T} \widehat{\mathbf{F}} dl + \int_{\partial\Omega_e \cup \partial\Omega} \mathbf{v}_h^{+T} \widehat{\mathbf{F}}^b dl, \tag{2.39}$$

where $\widehat{\mathbf{F}}$ is the numerical inviscid flux on the interior face σ^f and $\widehat{\mathbf{F}}^b$ is the boundary flux on the boundary face σ^b. Mathematically, these two fluxes can be expressed as

$$\begin{aligned}\widehat{\mathbf{F}} &= \widehat{\mathbf{F}}\left(\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}\right)\\ \widehat{\mathbf{F}}^b &= \widehat{\mathbf{F}}^b\left(\mathbf{u}_h^b, \mathrm{BC}, \vec{n}\right).\end{aligned}$$

The computation of the numerical inviscid flux $\widehat{\mathbf{F}}$ requires solving the local Riemann problem on the element interface. It is worth emphasizing that the solution can be approximate. In this work, we use the Roe approximate Riemann solver [102] with an entropy fix, which takes the form

$$\widehat{\mathbf{F}}\left(\mathbf{u}_h^+, \mathbf{u}_h^-\right) \cdot \vec{n} = \frac{1}{2}\left[\mathbf{F}(\mathbf{u}_h^+) + \mathbf{F}(\mathbf{u}_h^-)\right] - \frac{1}{2}\left|\frac{\partial \mathbf{F}}{\partial \mathbf{u}}(\mathbf{u}^*)\right|\left(\mathbf{u}_h^+ - \mathbf{u}_h^-\right), \tag{2.40}$$

where $\mathbf{F} = \vec{\mathbf{F}} \cdot \vec{n}$ and $\left|\frac{\partial \mathbf{F}}{\partial \mathbf{u}}(\mathbf{u}^*)\right|$ is a term that represents performing an eigenvalue decomposition on the normal flux Jacobian matrix $\partial\mathbf{F}/\partial\mathbf{u}$. Note that the inclusion of the absolute value in that expression means taking absolute values of eigenvalues. The boundary flux $\widehat{\mathbf{F}}^b$ is computed analytically based on the boundary state. One noteworthy exception is the convective flux for a boundary condition where an entire exterior state is specified. In this case, the Roe approximate Riemann solver is used to compute the boundary convective flux. More details on the boundary flux for the various boundary conditions used in this work are found in Appendix A.

In DG, the discretization of the viscous terms is far more complicated compared to CG and requires stabilization. This is because the diffusion terms have no preference whether a solution is differentiable everywhere, which is not realistic for DG solutions. This means, in this work, the viscous terms are discretized using the second form of Bassi and Rebay (BR2) [39]. Integrating by parts and applying the BR2 diffusion treatment, the elemental viscous discretization for DG can be written as

$$\begin{aligned}\mathcal{R}_{h,V}(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) = &-\int_{\Omega_e} \partial_i \mathbf{v}_h^T \mathbf{G}_i d\Omega + \int_{\partial\Omega_e \backslash \partial\Omega} \mathbf{v}_h^T \widehat{\mathbf{G}} dl + \int_{\partial\Omega_e \cup \partial\Omega} \mathbf{v}_h^{+T} \widehat{\mathbf{G}}^b dl\\ &- \int_{\partial\Omega_e \backslash \partial\Omega} \partial_i \mathbf{v}_h^{+T} \mathbf{K}_{ij}^+ \left(\mathbf{u}_h^+ - \widehat{\mathbf{u}}_h\right) n_j dl\\ &- \int_{\partial\Omega_e \cup \partial\Omega} \partial_i \mathbf{v}_h^{+T} \mathbf{K}_{ij}^b \left(\mathbf{u}_h^+ - \mathbf{u}_h^b\right) n_j dl,\end{aligned} \tag{2.41}$$

where $\widehat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$ is the unique state on the interior face, $\widehat{\mathbf{G}}$ is the stabilized viscous flux on the interior face σ^f, and $\widehat{\mathbf{G}}^b$ is boundary flux on the boundary face σ^b. Expressions for these two fluxes are

$$\widehat{\mathbf{G}} = \widehat{\mathbf{G}}\left(\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla\mathbf{u}_h^+, \nabla\mathbf{u}_h^-, \vec{n}\right)$$
$$\widehat{\mathbf{G}}^b = \widehat{\mathbf{G}}^b\left(\mathbf{u}_h^b, \nabla\mathbf{u}_h^+, \mathrm{BC}, \vec{n}\right).$$

The last two terms in Equation 2.41 symmetrize the weak form to maintain adjoint consistency. The BR2-stabilized viscous flux is given by

$$\widehat{\mathbf{G}} = \frac{1}{2}\left(\mathbf{G}_i^+ + \mathbf{G}_i^-\right) n_i^+ + \eta\frac{1}{2}\left(\boldsymbol{\delta}_i^+ + \boldsymbol{\delta}_i^-\right) n_i^+, \tag{2.42}$$

where η is the stabilization parameter that should be at least the number of faces for each element. In this work, η is set to double the maximum number of faces on adjacent elements. The auxiliary variables $\boldsymbol{\delta}_i^+, \boldsymbol{\delta}_i^- \in [\boldsymbol{\mathcal{V}}_h]^d$ have support on elements that are directly adjacent to the interior face σ^f. The auxiliary variables can be obtained by solving $\forall \boldsymbol{\tau}_{hi} \in \boldsymbol{\mathcal{V}}_h$

$$\int_{\Omega_e^+} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i^+ d\Omega + \int_{\Omega_e^-} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i^- d\Omega = \int_{\sigma^b} \frac{1}{2}\left(\boldsymbol{\tau}_{hi}^{+T}\mathbf{K}_{ij}^+ + \boldsymbol{\tau}_{hi}^{-T}\mathbf{K}_{ij}^-\right)\left(\mathbf{u}_h^+ - \mathbf{u}_h^-\right) n_j^+ dl. \tag{2.43}$$

The BR2-stabilized boundary viscous flux is given by

$$\widehat{\mathbf{G}}^b = \Pi_G^{\mathrm{BC}}\left[\mathbf{G}_i\left(\mathbf{u}_h^b, \nabla\mathbf{u}_h^+\right) n_i + \eta\boldsymbol{\delta}_i n_i\right]. \tag{2.44}$$

The auxiliary variable $\boldsymbol{\delta}_i \in [\boldsymbol{\mathcal{V}}_h]^d$ includes support on the element that is adjacent to the boundary face σ^b and is found by solving $\forall \boldsymbol{\tau}_{hi} \in \boldsymbol{\mathcal{V}}_h$

$$\int_{\Omega_e} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i d\Omega = \int_{\sigma^b} \boldsymbol{\tau}_{hi}^{+T}\mathbf{K}_{ij}^b\left(\mathbf{u}_h^+ - \mathbf{u}_h^b\right) n_j^+ dl. \tag{2.45}$$

The projection Π_G^{BC} in Equation 2.44 incorporates viscous flux boundary conditions. It is important to highlight that the choice of viscous fluxes is not unique. However, only certain options will lead to consistent, dual-consistent, and compact discretizations [103]. The set of fluxes incorporated into this work is shown in Table 2.1.

The source term is discretized through multiplication with the test functions, and then integration over the entire computational domain. For each element, the source is given by

$$\mathcal{R}_{h,S}\left(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}\right) = \int_{\Omega_e} \mathbf{v}_h^T \mathbf{S}\left(\mathbf{u}_h, \nabla\mathbf{u}_h\right). \tag{2.46}$$

Boundary Condition	$\widehat{\mathbf{G}}^b$	$\mathbf{K}_{ij}^b\mathbf{u}_h^b$
Dirichlet Boundary	$-\mathbf{K}_{ij}^b\partial_j\mathbf{u}_h^+ n_i + \eta\boldsymbol{\delta}_i n_i$	$\mathbf{K}_{ij}^b\mathbf{u}_h^b$
Neumann Boundary	$-\mathbf{K}_{ij}^b\partial_j\mathbf{u}_h^b n_i$	$\mathbf{K}_{ij}^+\mathbf{u}_h^+$

Table 2.1: Sample boundary viscous fluxes for different boundary conditions.

It is important to note that no boundary information is required to compute the source term discretization in Equation 2.46.

2.2.4 Discrete Form

By choosing the trial basis functions ϕ_{en} from Equation 2.35 as the general test functions in Equation 2.37, we obtain a discrete set of residuals over all elements in the computational domain. Given that span$\{\phi_{en}\} = \boldsymbol{\mathcal{V}}_h$, the solution over all elements in the domain is expressed mathematically as

$$\mathbf{u}_h(\vec{x}) = \sum_{e=1}^{N_e}\sum_{n=1}^{N_p}\mathbf{U}_{e,j}\phi_{e,j}(\vec{x}), \tag{2.47}$$

where $\mathbf{U}_{e,j}$ contains s expansion coefficients for basis function j on element e. The basis functions are actually defined in a reference space as order p polynomials, $\phi(\vec{\xi})$. The advantage of doing this is that they are identical for each element, no matter where they are located in the approximation space. Geometric mapping is used for each element to properly define the approximation space, given that the basis functions are always in reference space. This means that basis functions take the mathematical form of $\phi(\vec{x}(\vec{\xi}))$. More information on the geometric mapping is found in Section B.1 of Appendix B.

Even though the local approximations of the state have unique expansion coefficients, the number of basis functions per element is identical. This means that for each element, the number of unknowns is equal to N_p. These unknowns are referred to as the number of degrees of freedom per element. For two dimensional triangles, a full-order basis typically has $N_p = (p+1)(p+2)/2$, while a tensor product basis used on quadrilaterals has $N_p = (p+1)^2$. In practice, the state (and residual) vectors are stored as unrolled coefficients in a vector for all elements,

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{\text{elem1}} \\ \mathbf{U}_{\text{elem2}} \\ \mathbf{U}_{\text{elem3}} \\ \vdots \\ \mathbf{U}_{\text{elem}_{N_e}} \end{bmatrix}, \qquad \mathbf{U}_{\text{elem}_{N_e}} = \begin{bmatrix} \mathbf{U}_0 \\ \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_{N_p} \end{bmatrix}, \qquad \mathbf{U}_{N_p} = \begin{bmatrix} \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_s \end{bmatrix} \tag{2.48}$$

For each element, there is a smaller vector that contains the number of states at each basis node. It is important to emphasize that the number of basis functions per element depends on the order p. For each basis node, there is another vector equal to the state rank s. The total rank of the state vector is $N = N_e \times N_p \times s$, which is also equal to the number of equations that must be solved for and thus, the total number of residuals.

The steady component of the residuals can be assembled in a discrete spatial residual vector $\mathbf{R} \in \mathbb{R}^N$, that can be expressed as

$$\mathbf{R}(\mathbf{U}) = \mathcal{R}_h\left(\mathbf{u}_h, \phi_i\right). \tag{2.49}$$

The unsteady term from Equation 2.37 can be written in terms of a discrete mass matrix, $\mathbf{M} \in \mathbb{R}^{N\times N}$, where

$$\mathbf{M}_{ij} = \mathbf{I}_s \int_\Omega \phi_i \phi_j d\Omega. \tag{2.50}$$

In the mass matrix equation, $\mathbf{I}_s \in \mathbb{R}^{s\times s}$ is the identity matrix and $1 \leq i, j \leq N$ cover all degrees of freedom. Since the basis functions ϕ_i have support over only one element at a time, $\mathbf{M}$ is element-wise block diagonal. Using these definitions, the semi-discrete form of Equation 2.32 can be written as

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}\left(\mathbf{U}\right) = \mathbf{0}. \tag{2.51}$$

where $\mathbf{U} \in \mathbb{R}^N$ is the unrolled state vector that consists of the total number of degrees of freedom. For steady-state problems, the unsteady term drops out, so that Equation 2.51 simplifies to

$$\mathbf{R}\left(\mathbf{U}\right) = \mathbf{0}. \tag{2.52}$$

For steady problems, we use an implicit *pseudo* time-stepping method to solve Equation 2.51. To maintain stability regardless of the time step size, an implicit time discretization, backwards Euler, is used to temporally evolve the system. Given $\mathbf{U}^n \in \mathbb{R}^{N_{\text{DOF}}}$, where the time step is n and the number of degrees of freedom is N_{DOF}, backwards Euler finds $\mathbf{U}^{n+1} \in \mathbb{R}^{N_{\text{DOF}}}$ such that

$$\frac{\mathbf{M}}{\Delta t}\left(\mathbf{U}^{n+1} - \mathbf{U}^{n}\right) + \mathbf{R}\left(\mathbf{U}^{n+1}\right) = \mathbf{0}, \tag{2.53}$$

where time step, Δt, is computed using the Courant-Friedrichs-Lew (CFL) number. For each element, the CFL number is related to the artificial time step via

$$\text{CFL} = \frac{\Delta t_e c_e}{h_e}, \qquad \Delta t = \min_{\Omega_e \in T_h} \Delta t_e, \tag{2.54}$$

where h_e represents the element length and c_e is the maximum convective wave speed over the element. In two dimensions, the element length is the hydraulic radius,

$$h_e = \frac{2A_e}{P_e}, \tag{2.55}$$

where A_e is the element area and P_e is the element perimeter. The edge-weighted average of wave speeds is used to compute c_e. For a triangle, this is defined as

$$c_e = \sum_{e=1}^{3} c_{e,i} \frac{\Delta l_{e,i}}{P_e}, \tag{2.56}$$

where $c_{e,i}$ is the wave speed on edge i of element e and $\Delta l_{e,i}$ is the length of edge i.

2.2.5 Non-Linear Solver

To solve the discrete, non-linear system in Equation 2.53, the Newton-Raphson method is used. This method aims to iteratively find a solution using a succession of linearizations of the governing PDE. For a given Newton-Raphson iteration k, the unsteady residual can be written as

$$\widetilde{\mathbf{R}}\left(\mathbf{U}_k\right) = \frac{\mathbf{M}}{\Delta t}\left(\mathbf{U}_k - \mathbf{U}^{n}\right) + \mathbf{R}\left(\mathbf{U}_k\right), \tag{2.57}$$

where $\mathbf{U}_k$ is an estimate for $\mathbf{U}^{n+1}$. The goal of this method is to find the true $\mathbf{U}^{n+1}$. If $\widetilde{\mathbf{R}}\left(\mathbf{U}_k\right) = \mathbf{0}$, then $\mathbf{U}_k = \mathbf{U}^{n+1}$ and the algorithm is complete. To accomplish this, we define $\mathbf{U}_{k+1} = \mathbf{U}_k + \Delta\mathbf{U}_k$ as the solution that will yield the actual solution $\mathbf{U}^{n+1}$. We seek to update $\Delta\mathbf{U}_k$ so that

$$\widetilde{\mathbf{R}}\left(\mathbf{U}_{k+1}\right) \approx \mathbf{0}. \tag{2.58}$$

Linearizing Equation 2.58 produces

$$\begin{aligned}
&\frac{\mathbf{M}}{\Delta t}\left(\mathbf{U}_k - \mathbf{U}^n\right) + \frac{\mathbf{M}}{\Delta t}\Delta\mathbf{U}_k + \mathbf{R}\left(\mathbf{U}_k\right) + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_{\mathbf{U}_k}\Delta\mathbf{U}_k = \mathbf{0},\\
\rightarrow &\left(\frac{\mathbf{M}}{\Delta t} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_{\mathbf{U}_k}\right)\Delta\mathbf{U}_k + \frac{\mathbf{M}}{\Delta t}\left(\mathbf{U}_k - \mathbf{U}^n\right) + \mathbf{R}\left(\mathbf{U}_k\right) = \mathbf{0}.
\end{aligned} \tag{2.59}$$

For a steady-state solution, the second term in Equation 2.59 is ignored, since the evolution details are unnecessary. Consequently, $\Delta\mathbf{U}_k$ is found to be

$$\Delta\mathbf{U}_k = -\left(\frac{\mathbf{M}}{\Delta t} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_{\mathbf{U}_k}\right)^{-1}\mathbf{R}\left(\mathbf{U}_k\right). \tag{2.60}$$

The solution is marched forward in time by the given time step Δt until a desired error tolerance on the residual norm is satisfied. The linearized system in Equation 2.59 can be simplified to the general form of

$$\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}, \tag{2.61}$$

where

$$\begin{aligned}
\mathbf{A} &= \frac{\mathbf{M}}{\Delta t} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_{\mathbf{U}_k}\\
\mathbf{x} &= \Delta\mathbf{U}_k\\
\mathbf{b} &= \mathbf{R}\left(\mathbf{U}_k\right).
\end{aligned}$$

Solving Equation 2.61 requires a method that can handle solving a large linear system iteratively. In this work, we use an element-line preconditioned generalized minimal residual method (GMRES) [104, 105]. A preconditioner, **P**, is a pseudo-inverse to the matrix **A**. It improves the performance of GMRES since the resulting search space of vectors hones in on the true solution faster. The preconditioned form of Equation 2.61 is

$$\mathbf{P}^{-1}\left(\mathbf{A}\mathbf{x} + \mathbf{b}\right) = \mathbf{0}. \tag{2.62}$$

Obtaining the preconditioner can add additional computational time. However, this additional computational time is offset by the improvement in GMRES convergence.

2.3 Extension of DG to Usteady Problems

This section presents a brief analysis of the unsteady discontinuous Galerkin method. Specifically, we describe the solution approximation and discretization process. The work presented in this section is primarily based on reference [106].

2.3.1 Solution Approximation

For unsteady simulations, when the unsteady term in Equation 2.51 is not zero, we can define a strong-form unsteady residual, $\bar{\mathbf{R}}$, as

$$\bar{\mathbf{R}} \equiv \mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}\left(\mathbf{U}\right) = \mathbf{0}. \tag{2.63}$$

General time marching schemes are used for advancing the system of ODEs in time from the initial time at $t = 0$ to the final time at $t = T_f$. This requires computing the primal state at each time step Δt from time node n to $n+1$. Both multi-step and multi-stage methods can be used to advance the solution in time. Appendix C presents details of various backwards-difference multi-step methods (BDF) and implicit Runge-Kutta multi-stage methods.

2.3.2 Temporal Reconstruction

Temporal reconstruction of the primal solution at various time nodes is necessary for nonlinear problems because the unsteady residual in Equation 2.63 depends on the primal state $\mathbf{U}$. In multi-stage time integration, as well as error estimation, the primal state is not known for many of the intermediate time nodes. This makes it difficult, without significantly more storage or computation, to obtain an accurate representation of the unsteady residual. To alleviate this issue, during error estimation we solve for the primal at the intermediate time nodes using a temporal reconstruction with a prescribed order of accuracy.

By rearranging the semi-discrete form in Equation 2.63, we can evaluate the slope of the state,

$$\frac{d\mathbf{U}}{dt} = -\mathbf{M}^{-1}\mathbf{R}\left(\mathbf{U}\right). \tag{2.64}$$

Consequently, the temporal slope depends on a spatial residual calculation at a known state. If we know the states at the endpoints of a time interval from $t = t^n$ to t^{n+1}, a linear reconstruction between them only produces second-order accuracy. The accuracy can be improved by solving for the temporal slope and each endpoint using Equation 2.64. Figure 2.4 shows how using two slopes at the endpoints can lead to a cubic solution in time.

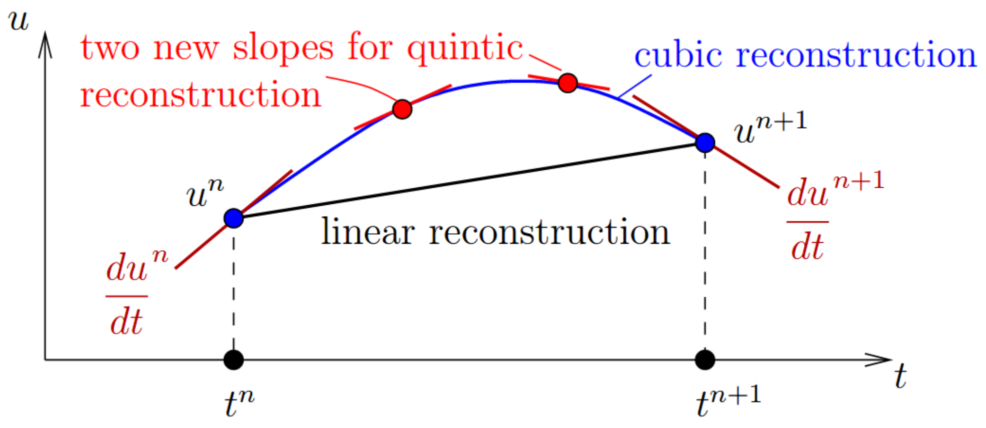


Figure 2.4: Graphical representation of solution reconstruction over the time interval of t^n to t^{n+1} (Figure reproduced from [106]).

Greater accuracy can be gained by solving for additional slopes between the two endpoints. Quadrature points are used in this work as the location of these additional temporal slope computations as they allow for recycling of residual evaluations when integrating. Using these two additional slopes produces a quintic solution in time. This process can be repeated indefinitely by using the current approximation to re-evaluate the slopes at the interior points. Each time this process is done, the temporal order of accuracy increases by one order. Further details of this process, including a numerical test of the reconstruction for a scalar governed by a simple ODE, can be found in [106].

2.4 The Streamline-Upwind Petrov-Galerkin Discretization

The primary focus of this section is to review the SUPG discretization of the Euler equations. We begin with a review of how the representation of the solution differs from DG when using SUPG since it is a continuous finite-element discretization. A full review of the SUPG weak form for the Euler equations follows. Extra attention is devoted to the stabilization term that is necessary for convection-dominated flows. Additional information regarding the linear solvers and other aspects of the numerical implementation is also discussed. The analysis presented in this section is an expansion of what was presented in [107].

2.4.1 Solution Approximation

Recall from the previous section that in DG the state **u** is approximated using a linear combination of basis functions on each element, as described in Subsection 2.2.2. This leads to a discontinuous solution approximation space instead of the continuous approximation space typical of CG discretizations. Since SUPG is a continuous finite-element discretization, the state will be approximated using a continuous space of the form

$$\mathbf{u}(\vec{x}) = \sum_{j=1}^{N} \mathbf{U}_j \phi_j(\vec{x}), \tag{2.65}$$

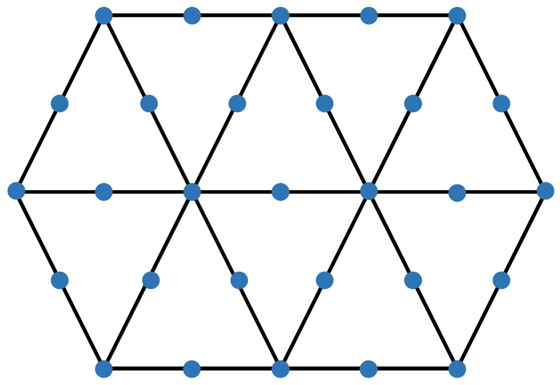
where N is the total number of degrees of freedom in the mesh, ϕ_j are the continuous trial basis functions over the entire domain, and $\mathbf{U}_j$ are the expansion coefficients. This approximation of the solution differs from the form in Equation 2.47 in that the solution is now continuous over the entire domain. Instead of approximating each elemental solution individually (as was done in DG), the solution is represented over the entire domain. Note that the same types of basis functions outlined in Subsection 2.2.2 are used in SUPG for this work. The main difference is how they are applied to the solution. However, just as in DG, the basis functions still have elemental compact support.

A disadvantage of using a continuous approximation space and a Galerkin formulation is that there is far less convective stability compared to a discontinuous space. However, while order enrichment is also more difficult in CG, there are far fewer total degrees of freedom. Figure 2.5 presents an example of the degree of freedom placement for $p = 2$ solution approximation on a triangular mesh with ten elements. In the SUPG mesh, there are a total of 29 degrees of freedom, while in the DG mesh there are 60 degrees of freedom. That is a significant difference in the number of degrees of freedom, which means there is a far greater computational cost associated with DG for a given mesh.

Table 2.2 shows the average degree of freedom counts per vertex of DG and SUPG for simple triangular meshes [23]. These results normalize the total degrees of freedom by the number of nodes in the mesh, assuming mesh regularity and ignoring boundaries. The cost savings of SUPG compared to DG is greatest at the lowest order. As the order increases, the relative difference in computational cost between the two methods decreases.

For a mesh made up of triangular elements, the total number of degrees of freedom is, again ignoring boundaries,

$$N = N_{\text{vertices}} + (p-1)N_{\text{faces}} + \left[\frac{(p+1)(p+2)}{2} - 3p\right] N_{\text{elem}}, \tag{2.66}$$


(a) SUPG

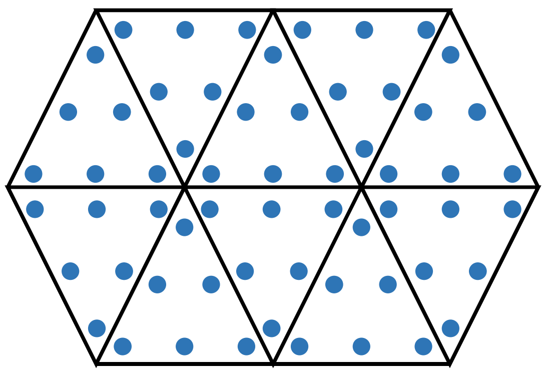

(b) DG

Figure 2.5: Degrees of freedom in a ten-element triangular mesh for $p = 2$ using various discretizations.

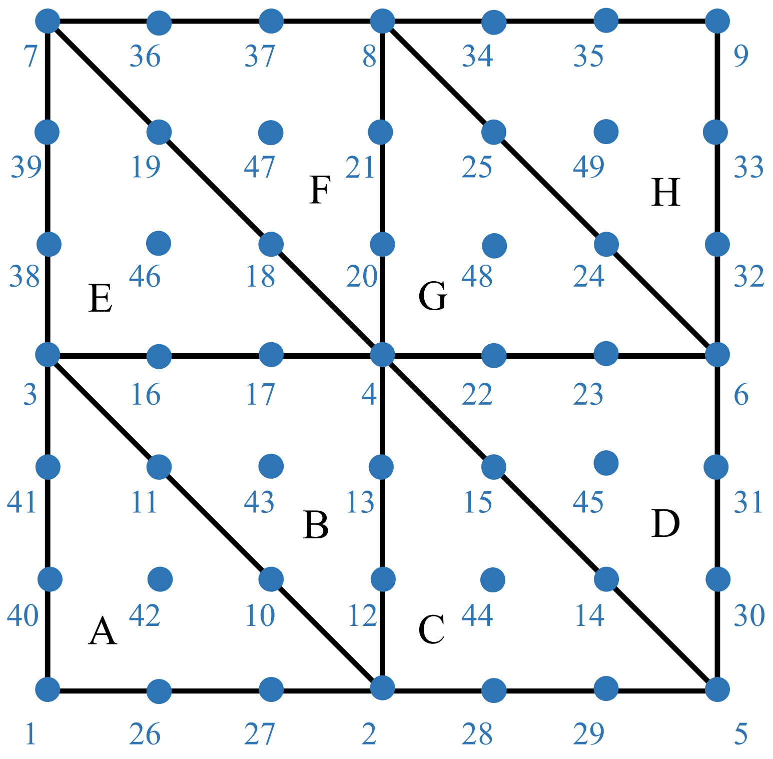
Method	$p = 1$	$p = 2$	$p = 3$
SUPG	1	4	9
DG	6	12	20

Table 2.2: Degree of freedom counts per vertex for a regular triangular mesh.

where N_{vertices} is the total number of vertices in the mesh, N_{faces} is the total number of faces, and N_{elem} is the total number of elements. Since the degrees of freedom are not unique to a particular element, the state and residual vectors cannot be stored as they were in Equation 2.48. Instead, we assign a global number to each degree of freedom, based on where it is located in the mesh. The order of degrees of freedom is based on the following progression:

1. Vertex nodes
2. Interior face nodes
3. Boundary face nodes
4. Interior element nodes

An example degrees of freedom map for an eight-element, triangular mesh with a solution approximation order of $p = 3$ is shown in Figure 2.6. In this mesh, we assume all exterior edges are boundary edges that will have a boundary condition associated with them. Using Equation 2.66, there should be 49 degrees of freedom associated with the solution for this mesh when using $p = 3$. In this mesh, there are a total of nine vertex nodes numbered 1 to 9 in Figure 2.6. Since the solution approximation order is $p = 3$, there are two nodes associated with each edge. This means nodes 10 to 25 are located on the eight interior edges and nodes 26 to 41 are located on the boundary edges. Finally, for $p = 3$ there is one node associated with each element interior. Consequently, nodes 42 to 49 are associated with each of the eight element interiors. This ordering accounts for all 49 degrees of freedom associated with this mesh.



$$\textbf{Elements}\left\{\begin{array}{c}\text{A:}\\\text{B:}\\\text{C:}\\\text{D:}\\\text{E:}\\\text{F:}\\\text{G:}\\\text{H:}\end{array}\right.\begin{bmatrix}1 & 26 & 27 & 2 & 40 & 42 & 10 & 41 & 11 & 3\\ 4 & 17 & 16 & 3 & 13 & 43 & 11 & 12 & 10 & 2\\ 2 & 28 & 29 & 5 & 12 & 44 & 14 & 13 & 15 & 4\\ 6 & 23 & 22 & 4 & 31 & 45 & 15 & 30 & 14 & 5\\ 3 & 16 & 17 & 4 & 38 & 46 & 18 & 39 & 19 & 7\\ 8 & 37 & 36 & 7 & 21 & 47 & 19 & 20 & 18 & 4\\ 4 & 22 & 23 & 6 & 20 & 48 & 24 & 21 & 25 & 8\\ 9 & 35 & 34 & 8 & 33 & 49 & 25 & 32 & 24 & 6\end{bmatrix}$$

Figure 2.6: Elemental degrees of freedom for at ten-element triangular mesh for $p = 3$ using SUPG.

Figure 2.6 also shows how the degrees of freedom are ordered and stored for each element. This ordering is based on the unit right-triangle ordering used for the polynomial basis functions. A sample order of degrees of freedom for a triangle at $p = 3$ is shown in Figure 2.7.

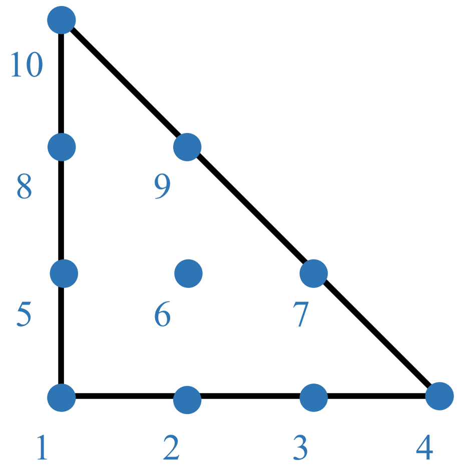


Figure 2.7: Ordering of degrees of freedom for a triangular element at $p = 3$.

2.4.2 Weak Form

The starting part of the SUPG discretization of the Euler equations is multiplying Equation 2.14 by a general test vector over the states, $\boldsymbol{\phi}$, to create the following weak form:

$$\int_{\Omega} \boldsymbol{\phi}^T \left[\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} \right] d\Omega = \mathbf{0}. \tag{2.67}$$

After integrating Equation 2.67 by parts, the weak form can be written as

$$\int_{\Omega} \left[\boldsymbol{\phi}^T \frac{\partial \mathbf{u}}{\partial t} - \frac{\partial \boldsymbol{\phi}^T}{\partial x_i} \mathbf{F}_i \right] d\Omega + \int_{\Gamma} \boldsymbol{\phi}^T \left(\hat{\mathbf{F}}_i^b n_i \right) d\Gamma = \mathbf{0}, \tag{2.68}$$

where $\hat{\mathbf{F}}_i^b n_i$ is the normal component of the flux on the domain boundary, Γ. For the far-field boundaries, the boundary flux is constructed using the Roe scheme [102], where the interior and freestream state values are used in the Riemann solver. For the inviscid wall boundaries, the boundary flux at the wall is computed using the boundary pressure and velocity. More details on constructing the boundary flux are given in Appendix A.

The Galerkin continuous finite element discretization (as shown in Equation 2.68) will be unstable for advection-dominated flows. The standard Galerkin formulation produces odd-even decoupling between adjacent nodes in the solution which leads to oscillatory behavior. These spurious oscillations will eventually lead to instability in the time integration. Several stabilization methods in the past have been studied to correct this instability. One such approach, the Streamline-Upwind Petrov-Galerkin (SUPG) method [50, 51, 57, 108], was chosen for this discretization. In this approach, a stabilization term is added to the weak form to compensate for the lack of dissipation in the streamwise direction [49].

To better understand the stability notation, it is useful to rewrite Equation 2.14 using a differential operator $\mathcal{L}$ [54],

$$\mathcal{L}(\mathbf{u}) = \mathbf{0}, \qquad \mathcal{L} = \frac{\partial}{\partial t} + \mathbf{A}_i \frac{\partial}{\partial x_i}, \tag{2.69}$$

where $\mathbf{A}_i$ is the inviscid flux Jacobian. The added stability necessary to obtain converged solutions can be achieved by adding an upwind bias to the test functions $\boldsymbol{\phi}$. In this case, the convective portion of the aforementioned operator, $\mathcal{L}_c$, acting on the basis functions is added to the existing test functions to produce [56, 109]

$$\hat{\boldsymbol{\phi}} = \boldsymbol{\phi} + \boldsymbol{\tau}\mathcal{L}_c(\boldsymbol{\phi}) = \boldsymbol{\phi} + \boldsymbol{\tau}\mathbf{A}_i \frac{\partial \boldsymbol{\phi}}{\partial x_i}, \tag{2.70}$$

where $\boldsymbol{\tau}$ is the stabilization matrix. The purpose of this matrix is to limit the amount of numerical dissipation added to the scheme to as low of an amount as possible while preserving stability.

Using this augmented test function, the weak form in Equation 2.68 can be rewritten as [21, 59]

$$\int_\Omega \boldsymbol{\phi}^T \frac{\partial \mathbf{u}}{\partial t} - \frac{\partial \boldsymbol{\phi}}{\partial x_i}\mathbf{F}_i d\Omega + \int_\Gamma \boldsymbol{\phi}^T \left(\hat{\mathbf{F}}_i^b n_i\right) d\Gamma + \underbrace{\sum_{e=1}^{N_e} \int_\Omega \mathbf{P}_e \left(\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i}\right) d\Omega}_{\text{Stabilization term}} = \mathbf{0}. \tag{2.71}$$

In the above equation, it is important to note that the stabilization term is calculated over all of the elements in the domain Ω. The stabilization term only affects the element interior and does not have an impact on the boundary terms. The elemental perturbation term, $\mathbf{P}_e$, can be expanded as [57, 59, 110] (summation implied on i)

$$\mathbf{P}_e = \mathbf{A}_i \frac{\partial \boldsymbol{\phi}}{\partial x_i} \boldsymbol{\tau}_e. \tag{2.72}$$

A detailed discussion of the elemental stabilization matrix, $\boldsymbol{\tau}_e$, is included in the next subsection.

2.4.3 Stabilization Matrix

The elemental stabilization matrix is obtained using the eigensystem decomposition of the projection of the flux Jacobian matrices onto the spatial gradients of the basis functions [21, 52]. For element e,

$$\boldsymbol{\tau}_e^{-1} = \sum_{j=1}^{N_p} \left| \frac{\partial \phi_{e,j}}{\partial x_i} \mathbf{A}_i \right| \tag{2.73}$$

where $\phi_{e,j}$ are the basis functions and N_p is the number of nodes within element e. In Equation 2.73,

$$\left| \frac{\partial \phi_{e,j}}{\partial x_i} \mathbf{A}_i \right| = \mathbf{T} \left| \boldsymbol{\Lambda} \right| \mathbf{T}^{-1}, \tag{2.74}$$

where $\mathbf{T}$ denotes the matrix of right eigenvectors and $|\boldsymbol{\Lambda}|$ denotes the diagonal matrix of absolute values of the eigenvalues. The quantity i indexes the spatial dimension, whereas j indexes the basis functions for the specific element e. It is important to note that the inverse of the stability matrix is evaluated at each Gauss quadrature point for volume integration, which means that there is a unique elemental stability matrix for each quadrature point. Note that even for a high-order discretization, $p \geq 2$, only the linear shape functions are used for the computation of the stabilization matrix. It has been previously shown that including the higher-order basis functions will reduce the amount of dissipation added to the scheme [111]. Since the stabilization term was initially developed for linear basis functions [108] and there is no formal derivation of the stability matrix for high-order discretization, the choice was made to use linear basis functions in Equations 2.72 and 2.73.

An eigensystem decomposition of the flux Jacobian matrix is necessary for the implementation of the Roe scheme [102]. To make the calculation easier, for two dimensions we assume a coordinate system so that $u = \vec{v} \cdot \vec{n}$. This means that the second component of the inviscid flux Jacobian is zero. For the compressible Euler equations, the inviscid flux Jacobian matrix can be written as

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -u^2 + \frac{\gamma-1}{2}V^2 & (3-\gamma)u & -(\gamma-1)v & \gamma-1 \\ -uv & v & u & 0 \\ \frac{\gamma-2}{2}uV^2 - \frac{uc^2}{\gamma-1} & H-(\gamma-1)u^2 & -(\gamma-1)uv & \gamma u \end{bmatrix}, \tag{2.75}$$

where $V^2 = u^2 + v^2$ is the square of the velocity magnitude.

Using the flux Jacobian matrix expression from Equation 2.75, the right ($\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4]$)

and left $\left(\mathbf{L} = \mathbf{R}^{-1} = [\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3, \mathbf{l}_4]\right)$ eigenvectors can be expanded as

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ u+a & u-a & 0 & u \\ v & v & v & v \\ H+ua & H-ua & v^2 & \frac{1}{2}V^2 \end{bmatrix},$$
$$\mathbf{L} = \frac{\gamma-1}{2a^2}\begin{bmatrix} \frac{V^2}{2} - \frac{ua}{\gamma-1} & -u + \frac{a}{\gamma-1} & -v & 1 \\ \frac{V^2}{2} + \frac{ua}{\gamma-1} & -u - \frac{a}{\gamma-1} & -v & 1 \\ -\frac{2a^2}{\gamma-1} & 0 & \frac{2a^2}{(\gamma-1)v} & 0 \\ -V^2 + \frac{2a^2}{\gamma-1} & 2u & 2v & -2 \end{bmatrix} \tag{2.76}$$

and the corresponding eigenvalues are

$$\mathbf{\Lambda} = \text{diag}\left([\lambda_1, \lambda_2, \lambda_3, \lambda_4]\right) = \text{diag}\left([u+a, u-a, u, u]\right). \tag{2.77}$$

The above equations not only assume the use of the Euler equations, but also a particular coordinate system. Since the elemental stabilization matrix, $\boldsymbol{\tau}_e$, depends on the projection of the flux Jacobian matrices onto the spatial gradients of the basis functions, a coordinate system rotation must be performed on the subset of the state vector in Equation 2.65 that corresponds to the element e at Gaussian quadrature point q before using Equations 2.76 and 2.77. The state vector is represented at Gaussian quadrature points since we use Gaussian quadrature rules to solve numerical integration. We will refer to the state vector term as $\mathbf{u}_{e,q}$. In two dimensions, we can define the rotation matrix as

$$\mathbf{Q}_j = \frac{1}{\left|\frac{\partial \phi_{e,j}}{\partial x_i}\right|}\begin{bmatrix} \left|\frac{\partial \phi_{e,j}}{\partial x_i}\right| & 0 & 0 & 0 \\ 0 & \frac{\partial \phi_{e,j}}{\partial x} & \frac{\partial \phi_{e,j}}{\partial y} & 0 \\ 0 & -\frac{\partial \phi_{e,j}}{\partial y} & \frac{\partial \phi_{e,j}}{\partial x} & 0 \\ 0 & 0 & 0 & \left|\frac{\partial \phi_{e,j}}{\partial x_i}\right| \end{bmatrix} \tag{2.78}$$

where $\frac{\partial \phi_{e,j}}{\partial x}$ is the gradient of the basis functions in the x direction and $\frac{\partial \phi_{e,j}}{\partial y}$ is the gradient of the basis functions in the y direction, and $\left|\frac{\partial \phi_{e,j}}{\partial x_i}\right|$ is the magnitude of the gradient of the basis function. It is worth noting that the rotation matrix is unique for each node j in element e.

Once the rotation matrix is applied to the elemental state vector, the eigenvectors and eigenvalues can be calculated using Equations 2.76 and 2.77, respectively. However, before computing the expression in Equation 2.74, we must use the rotation matrix to rotate the result back to the original coordinate system. This can be done using the following expression,

$$\left|\frac{\partial \phi_{e,j}}{\partial x_i}\mathbf{A}_i\right| = \sum_{j=1}^{N_p} \mathbf{Q}_j^T \left(\mathbf{T}\left|\boldsymbol{\Lambda}\right|\mathbf{T}^{-1}\right)\mathbf{Q}_j. \tag{2.79}$$

where $\mathbf{T} = \mathbf{R}(\mathbf{u}_{e,q})$, $\mathbf{T}^{-1} = \mathbf{L}(\mathbf{u}_{e,q})$, and $|\boldsymbol{\Lambda}| = |\boldsymbol{\Lambda}(\mathbf{u}_{e,q})|$. We can then compute the inverse of stability matrix using Equation 2.73.

2.4.4 General Numerics and Solver

The SUPG code written for this work also uses an open-source suite of various data structures and routines, PETSc [112–114], for solving large linear systems of equations that were already expanded upon for DG in Subsection 2.2.5. In this work, we use PETSc routines that solve systems using preconditioned Krylov subspace methods. Incomplete LU factorization (ILU) with two levels of fill was chosen as the preconditioner [115–117]. It is worth noting that when using DG, no levels of fill in the preconditioner were necessary. However, the two levels of fill were required to achieve convergence for most applications of the SUPG code in this work. The generalized minimal residual method (GMRES) [118] with Gram-Schmidt orthogonalization was used for the Krylov subspace method.

CHAPTER 3

Adjoints and Error Estimation Approaches

This chapter reviews the three main types of error estimation approaches used in this work in the context of steady-state problems. We first review the basic concepts of the discrete adjoint and how adjoints can be used for error estimation, which leads to a detailed description of the adjoint-weighted residual method and the implementation of error estimation using output-based adjoints. Next, we introduce the concept of the entropy variables and how they can act as adjoints, which leads to a different error estimation technique. With both the output-based adjoint and entropy variables approaches defined, we then detail the various novel combined approaches implemented for steady-state problems. The chapter closes with a summary of the general adaptation strategy implemented in this work.

3.1 Discrete Adjoint

Accurate prediction of an output (e.g. drag or lift) relies on an appropriate amount of resolution in areas of the computational domain that may not always be obvious upon visual inspection. In hyperbolic problems, it is common for disturbances to propagate and affect areas of the solution far downstream of their origin. Adjoints provide a way to quantify the effects of input parameters on a particular output of interest. These inputs could be anything from flow field conditions to geometry specifications. By computing an adjoint to measure these sensitivities, there is no need to repeatedly solve the complete system of equations for various sets of inputs.

Consider a discretized PDE,

$$\mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{3.1}$$

where $\mathbf{U} \in \mathbb{R}^N$ is a discrete state vector, $\mathbf{R}$ is the residual vector, and N is the number of discrete equations. The discrete adjoint, $\mathbf{\Psi} \in \mathbb{R}^N$, is a Green's function that relates residual source perturbations to a scalar output of interest, $J(\mathbf{U})$. Mathematically, this can be expressed as

$$\boldsymbol{\Psi}^T = \frac{\partial J(\mathbf{U})}{\partial \mathbf{R}(\mathbf{U})}. \tag{3.2}$$

In other words, $\boldsymbol{\Psi}$ is a vector of sensitivities of a scalar output $J(\mathbf{U})$ to the residual vector $\mathbf{R}$. A common source of residual perturbations is a change in inputs parameters, $\boldsymbol{\mu}$. Using a local sensitivity analysis, a chain of dependence linking these inputs to the scalar output can be written as

$$\underbrace{\boldsymbol{\mu}}_{\text{inputs}} \rightarrow \underbrace{\mathbf{R}(\mathbf{U},\boldsymbol{\mu})}_{\text{residuals}} \rightarrow \underbrace{\mathbf{U}}_{\text{state}} \rightarrow \underbrace{J(\mathbf{U})}_{\text{output}}. \tag{3.3}$$

In the context of output-based error estimation, the effects of the inputs $\boldsymbol{\mu}$ on the output $J(\mathbf{U})$ are of particular interest. A straightforward way of computing these sensitivities is a forward linearization, in which the sequence in Equation 3.3 is linearized. However, when the number of inputs is large, resolving the forward problem for each input is computationally expensive and not feasible for typical application purposes.

An alternate approach uses the discrete adjoint to relate the sensitivity of the inputs to the scalar output. Using the definition of the adjoint from Equation 3.2, the sensitivity chain can be expressed as

$$\frac{\partial J}{\partial \boldsymbol{\mu}} = \left(\frac{\partial J}{\partial \mathbf{U}}\right)\left(\frac{\partial \mathbf{U}}{\partial \mathbf{R}}\right)\left(\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\right) = \boldsymbol{\Psi}^T\left(\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\right). \tag{3.4}$$

The adjoint approach significantly reduces the computational cost because the effects of $\mathbf{R}$ on J only have to be computed once, regardless of how many times the inputs change. The effect of the inputs on $\mathbf{R}$ must be computed for each perturbation, but fortunately, this is not as computationally expensive. By separating the effects of $\boldsymbol{\mu}$ on $\mathbf{R}$ and $\mathbf{R}$ on J, the computationally expensive forward problem can be bypassed, as shown in Figure 3.1.

To derive the adjoint equation, the following chain of operations is considered when small perturbations are assumed.

1. A small perturbation is applied to the input parameters:
   $\boldsymbol{\mu} \rightarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}$
2. The residual is linearized about the perturbed inputs:
   $\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}+\delta\boldsymbol{\mu}) = \delta\mathbf{R} \neq \mathbf{0} \;\; \rightarrow \mathbf{R}(\mathbf{U},\boldsymbol{\mu}) + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\big|_{\mathbf{U},\boldsymbol{\mu}}\delta\boldsymbol{\mu} = \delta\mathbf{R}$
3. The residual is linearized about the perturbed state and inputs:
   $\mathbf{R}(\mathbf{U}+\delta\mathbf{U}, \boldsymbol{\mu}+\delta\boldsymbol{\mu}) = \mathbf{0} \;\; \rightarrow \mathbf{R}(\mathbf{U},\boldsymbol{\mu}) + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\big|_{\mathbf{U},\boldsymbol{\mu}}\delta\boldsymbol{\mu} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\big|_{\mathbf{U},\boldsymbol{\mu}}\delta\mathbf{U} = \mathbf{0}$

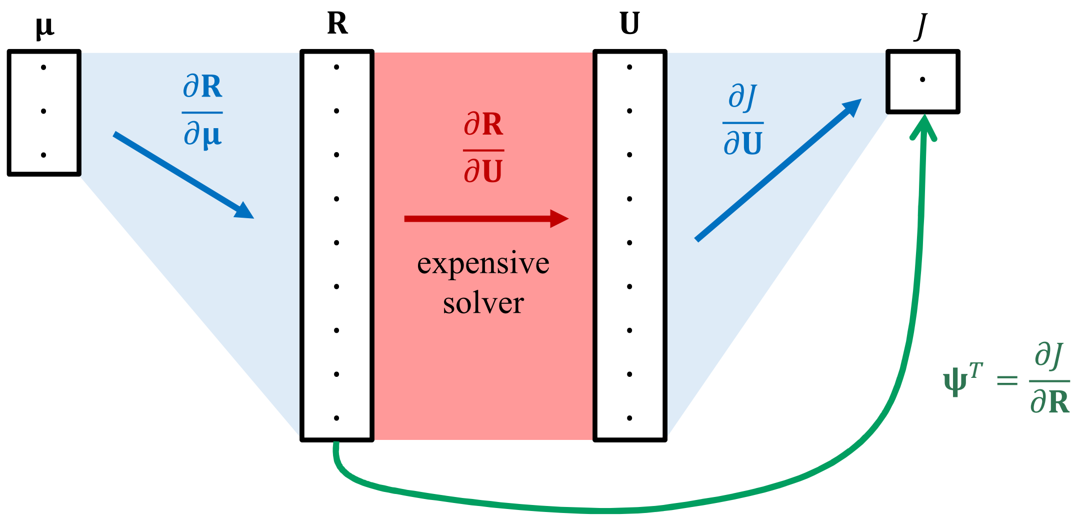


Figure 3.1: Adjoint method for sensitivity analysis.

4. The output is linearized about the perturbed states:
   $J(\mathbf{U} + \delta\mathbf{U}) = J(\mathbf{U}) + \delta J \rightarrow \delta J = \frac{\partial J}{\partial \mathbf{U}}\delta\mathbf{U}$
5. Subtract step 2 from step 3 to obtain the perturbated state $\delta\mathbf{U}$:
   $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U},\boldsymbol{\mu}}\delta\mathbf{U} = -\delta\mathbf{R} \Rightarrow \delta\mathbf{U} = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1}\delta\mathbf{R}$
6. Combine results from step 5 with the output linearization in step 4 to produce the output perturbation in terms of the residual perturbation:
   $\delta J = \frac{\partial J}{\partial \mathbf{U}}\delta\mathbf{U} = \underbrace{-\frac{\partial J}{\partial \mathbf{U}}\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1}}_{\boldsymbol{\Psi}^T \in \mathbb{R}^N}\delta\mathbf{R}$
7. Taking the transpose of the equation for $\boldsymbol{\Psi}^T$ and rearranging the equation in step 6 produces the *adjoint equation*:

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \boldsymbol{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = \mathbf{0}. \tag{3.5}$$

The n^{th} component of $\boldsymbol{\Psi}$ directly relates how changes in the n^{th} residual affect the output J. Given than $\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = \mathbf{0}$, the expression from Step 2 can be simplified to $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\Big|_{\mathbf{U},\boldsymbol{\mu}}\delta\boldsymbol{\mu} = \delta\mathbf{R}$. With this new expression, the result from Step 6 can be written as

$$\delta J = \boldsymbol{\Psi}^T\delta\mathbf{R} \rightarrow \delta J = \boldsymbol{\Psi}^T\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\bigg|_{\mathbf{U},\boldsymbol{\mu}}\delta\boldsymbol{\mu} \rightarrow \frac{\partial J}{\partial \boldsymbol{\mu}} = \boldsymbol{\Psi}^T\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\bigg|_{\mathbf{U},\boldsymbol{\mu}}. \tag{3.6}$$

The key conclusion from the above expression is that $\boldsymbol{\Psi}$ only has to be computed once for the same output, regardless of how many input sensitivities exist. The term $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}$ must be computed for each input. However, it is far cheaper to solve for this quantity than that obtained from the expensive forward problem, $\frac{\partial \mathbf{R}}{\partial \mathbf{U}}$.

3.1.1 Adjoint Consistency

The discrete adjoint from Equation 3.4, $\mathbf{\Psi}$, has a continuous counterpart called $\boldsymbol{\psi}(\vec{x})$. We can think of the N values in $\mathbf{\Psi}$ as expansion coefficients in an approximation of $\boldsymbol{\psi}$ using identical basis functions used for the primal problem. The accuracy of this approximation is important for error estimation and mesh adaptation.

Suppose the primal problem has an exact solution, $\mathbf{u}_h \in \mathcal{V}_h$, such that

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = 0, \qquad \forall\, \mathbf{v}_h \in \mathcal{V}_h, \tag{3.7}$$

where $\mathcal{V}_h$ is the trial and test function space and $\mathcal{R}_h(\cdot,\cdot) : \mathcal{V}_h \times \mathcal{V}_h \to \mathbb{R}$ is the semilinear operator that represents the weak form of the differential equation. The subscript h indicates a discretization of the computational domain, such as a finite-element method. Consider a residual perturbation, $\delta\mathcal{R}_h(\mathbf{v}_h)$, added to the governing PDE in Equation 3.7. The adjoint $\boldsymbol{\psi}_h \in \mathcal{V}_h$ is the sensitivity of a scalar output $\mathcal{J}$ to the residual perturbation in the following relationship:

$$\delta\mathcal{J}_h \equiv \mathcal{J}_h(\mathbf{u}_h + \delta\mathbf{u}_h) - \mathcal{J}_h(\mathbf{u}_h) = \delta\mathcal{R}_h(\boldsymbol{\psi}_h). \tag{3.8}$$

The state and residual perturbations can be related via the following statement [73]:

$$\mathcal{R}_h'[\mathbf{u}_h](\delta\mathbf{u}_h, \mathbf{v}_h) + \delta\mathcal{R}_h(\mathbf{v}_h) = 0, \qquad \forall\, \mathbf{v}_h \in \mathcal{V}_h, \tag{3.9}$$

where the prime denotes Fréchét linearization about $\mathbf{u}$. Linearizing the output in Equation 3.8 and combing the result with Equation 3.9 produces the following statement that the adjoint must satisfy: determine $\boldsymbol{\psi}_h \in \mathcal{V}_h$ such that

$$\mathcal{R}_h'[\mathbf{u}_h](\mathbf{v}_h, \boldsymbol{\psi}_h) + \mathcal{J}'[\mathbf{u}_h](\mathbf{v}_h) = 0, \qquad \forall\, \mathbf{v}_h \in \mathcal{V}_h, \tag{3.10}$$

The solution of Equation 3.10 is an adjoint, $\boldsymbol{\psi}_h$, which can be represented by N numbers in a finite-dimensional space. Given the exact primal solution, $\mathbf{u} \in \mathcal{V}$, satisfies

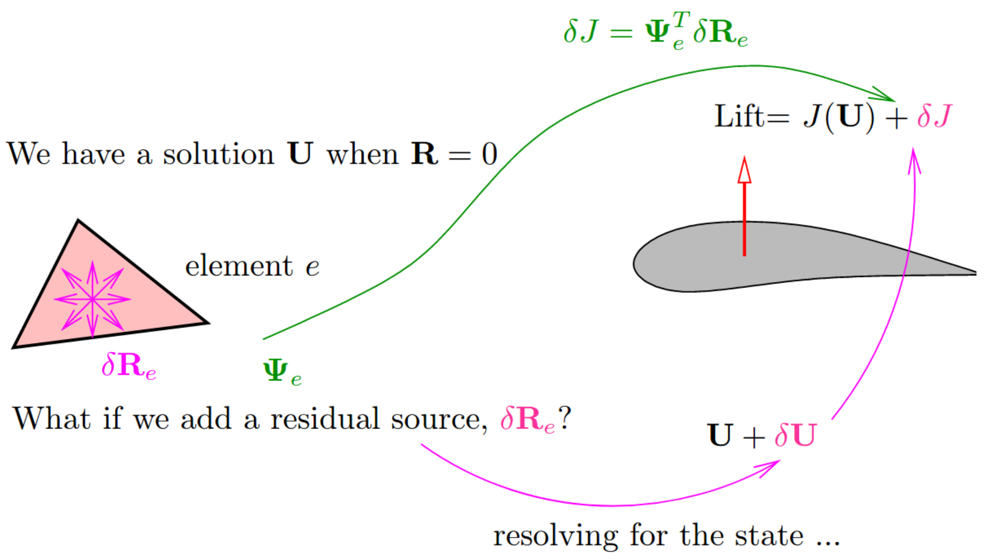
$$\mathcal{R}(\mathbf{u}, \mathbf{v}) = 0, \qquad \forall\, \mathbf{v} \in \mathcal{V} \tag{3.11}$$

for an appropriately defined space $\mathcal{V}$, the exact adjoint $\psi \in \mathcal{V}$ satisfies

$$\mathcal{R}'[\mathbf{u}](\mathbf{v},\psi) + \mathcal{J}'[\mathbf{u}](\mathbf{v}) = 0, \qquad \forall\, \mathbf{v} \in \mathcal{V}. \tag{3.12}$$

It is important to note that $\mathcal{R}$ and $\mathcal{J}$ are continuous versions of semilinear form and output, respectively.

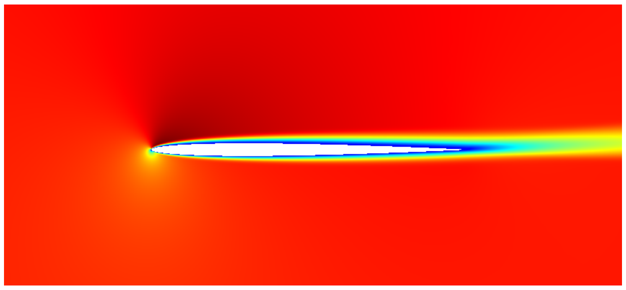
The exact adjoint can be regarded as a Green's function that relates residual source perturbations in the original PDE to perturbations in the output, as shown in Figure 3.2. A sample adjoint solution

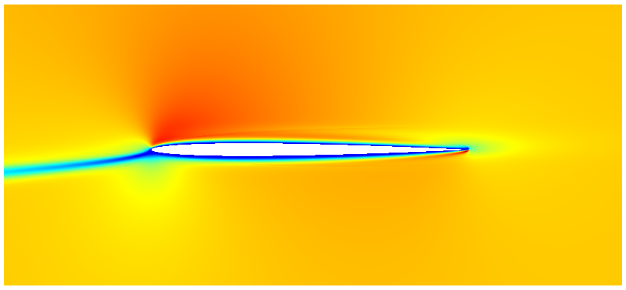


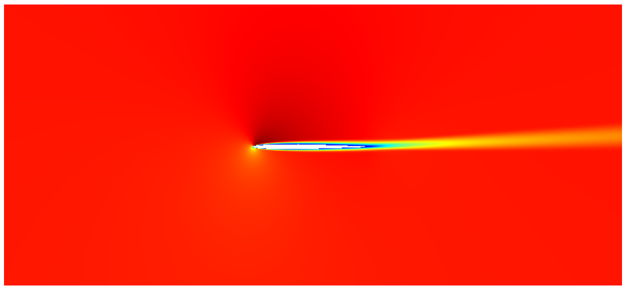
$\delta J = \boldsymbol{\Psi}_e^T \delta \mathbf{R}_e$
Lift= $J(\mathbf{U}) + \delta J$
We have a solution $\mathbf{U}$ when $\mathbf{R} = 0$
element e
$\delta \mathbf{R}_e$
$\boldsymbol{\Psi}_e$
What if we add a residual source, $\delta \mathbf{R}_e$?
$\mathbf{U} + \delta \mathbf{U}$
resolving for the state ...

Figure 3.2: Diagram of the adjoint acting as the sensitivity of a scalar output to residual source perturbations (Figure reproduced from [95]).

is shown in Figure 3.3 for subsonic flow over a NACA 0004 airfoil using the RANS equations. The freestream Mach number is 0.5, the Reynolds number is 10000 and the airfoil angle of attack is $3°$. The output of interest, J, is the total drag on the airfoil. The plots on the left show the x-momentum component of the primal state, while the plots on the right show the conservation of x-momentum component for the drag adjoint. The primal solution shows the presence of the boundary layer near the surface of the airfoil and the wake emanating from the trailing edge. The adjoint solution shows similar features as well, though, there are a few major differences. The most pronounced difference is the "wake reversal" emanating from the leading edge of the airfoil and traveling in the opposite direction of the left-to-right flow. This indicates the drag output is more sensitive to residual perturbations upstream of the airfoil and around the stagnation streamline than perturbations in other regions of the domain. The adjoint boundary layer and "reverse wake" are inherent to primal-adjoint symmetry and not just unique to this problem. The propensity of the output-based adjoint to focus refinement on the “reverse wake” region can lead to over-refinement of regions of the computational domain far from the airfoil. This point will be further discussed in future chapters when entropy variables and combined approaches for error estimation are analyzed.


(a) x-momentum (zoom in)


(b) x-momentum adjoint (zoom in)


(c) x-momentum (zoom out)

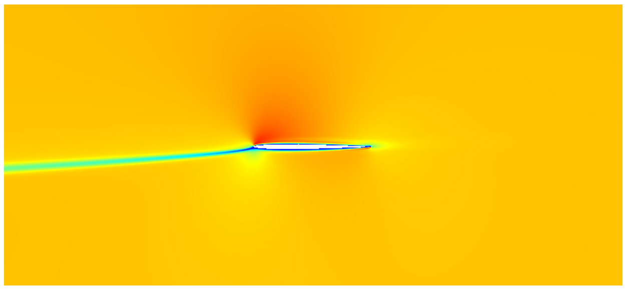

(d) x-momentum adjoint (zoom out)

Figure 3.3: Samples of the primal solution (x-momentum component) and adjoint solution (conservation of x-momentum equation component) for drag output in RANS flow over a NACA 0004 airfoil. The color scales are clipped to show interesting features. In the adjoint plots, red and blue regions highlight regions where the drag output is most sensitive to.

The discrete adjoint solution presented in Figure 3.1 can be regarded as a faithful representation of the continuous (exact) adjoint if the discretization is consistent with the exact adjoint problem. *Primal consistency* in the variational problem mandates that the exact solution of the primal state, **u** satisfies the variational form of the PDE,

$$\mathcal{R}_h(\mathbf{u},\mathbf{v}) = 0, \qquad \forall\, \mathbf{v} \in \mathcal{W}_h, \tag{3.13}$$

where $\mathcal{W}_h = \mathcal{V}_h + \mathcal{V} = \{h = f + g \ : \ f \in \mathcal{V}_h,\ g \in \mathcal{V}\}$. The combination of the discrete semi-linear form $\mathcal{R}_h$ and functional $\mathcal{J}_h$ is adjoint consistent if [31, 103, 119]

$$\mathcal{R}'_h[\mathbf{u}](\mathbf{v},\boldsymbol{\psi}) + \mathcal{J}'_h[\mathbf{u}](\mathbf{v}) = 0, \qquad \forall\, \mathbf{v} \in \mathcal{W}_h. \tag{3.14}$$

Even if a discretization is not adjoint consistent, it may be asymptotically adjoint consistent if Equation 3.14 is valid in the limit $h \to 0$, by which we mean the limit of uniformly increasing resolution over suitably normalized $\mathbf{v} \in \mathcal{W}_h$. Achieving adjoint consistency is not only important for convergence of the adjoint approximation, but also the convergence of the primal approximation. With the continuous adjoint consistency defined, it can now be used as an important tool for output error estimation.

3.2 Error Estimation Using Output-Based Adjoints

Given a desired engineering output, output-based error estimation methods refine areas of the mesh important for an accurate prediction of the output. They account for error propagation effects inherent to convection-dominated flow simulations by targeting residuals to which the output is most sensitive. The resulting error estimates provide confidence levels to the output calculations and can be localized to elements to drive adaptation. These error estimations rely on the solution of the adjoint problem explored in the previous section, which yields (in continuous form) the adjoint $\psi(\vec{x})$. In this section, computing output error estimates using an adjointed-weighted residual method will be thoroughly reviewed. The approach is this section is based on [19, 24, 73, 87, 95, 100, 120].

3.2.1 Output-Based Error Estimation Definition

Since estimating the true numerical output is challenging for general nonlinear problems, error in the output is estimated between two finite-dimensional spaces. The first space is the coarse approximation space $(\mathcal{V}_H)$, where the state and output are calculated. The second space is a finer space $(\mathcal{V}_h)$ where the error is estimated. The output-based adjoint error estimation process relies on two observations.

The first observation is that an approximate solution of $\mathbf{u}_H$ will not satisfy the fine-space residual $\mathcal{R}_h(\mathbf{u}_H, \mathbf{v}_h) \neq 0$. However, if the perturbation $\delta\mathbf{u} \equiv \mathbf{u}_H - \mathbf{u}$ is small, we can write

$$\mathcal{R}_h(\mathbf{u}_H, \mathbf{v}_h) = \mathcal{R}_h\left(\mathbf{u} + \delta\mathbf{u}, \mathbf{v}_h\right) \approx \mathcal{R}'_h\left[\mathbf{u}\right]\left(\delta\mathbf{u}, \mathbf{v}_h\right). \tag{3.15}$$

The second observation is that the fine-space adjoint ψ_h translates the residual perturbation to an output perturbation. Using Equation 3.12, we can express this mathematically as

$$\delta\mathcal{J} = \mathcal{J}'[\mathbf{u}](\delta\mathbf{u}) = -\mathcal{R}'_h[\mathbf{u}](\delta\mathbf{u}, \psi_h) = -\mathcal{R}_h(\mathbf{u}_H, \psi_h). \tag{3.16}$$

The above expression quantifies the numerical error in the output via a weighted residual of an approximation of the solution. We use an approximation sign in Equation 3.16 to indicate that the equation is not exact for noninfinitesimal perturbations and is just an estimate of the numerical error. The fine-space adjoint ψ_h is an approximation of ψ on the enriched finite-element space,

$\mathcal{V}_h$. To minimize the computational cost, the finer space $\mathcal{V}_h$ is constructed from $\mathcal{V}_H$ by increasing the order to $p+1$ and keeping the mesh the same. The fine-space adjoint, $\boldsymbol{\psi}_h$, is solved for either exactly or approximately using an iterative smoother.

We can rewrite Equation 3.16 as

$$\delta \mathcal{J} \approx -\mathcal{R}_h\left(\mathbf{u}_H, \boldsymbol{\psi}_h - \boldsymbol{\psi}_H\right) = -\sum_{e \in T_H} \mathcal{R}_h\left(\mathbf{u}_H, (\boldsymbol{\psi}_h - \boldsymbol{\psi}_H)|_e\right). \tag{3.17}$$

In the above expressions, $|_e$ denotes the restriction of an interpolated function to element e of the triangulation T_H. Note that in Equation 3.17 we use the difference between the fine-space and coarse-space adjoints, $\boldsymbol{\psi}_h - \boldsymbol{\psi}_H$, to minimize the error due to possible p-dependence of the residual. When Galerkin orthogonality holds, the subtraction of $\boldsymbol{\psi}_H$ does not impact the error estimate or the error indicator. However, DG discretizations often exhibit p-dependence, which refers to the presence of terms (e.g. in the BR2 stabilization) that explicitly depend on the solution approximation order, p. In these cases, a coarse-space solution prolonged into the fine-space does not necessarily yield zero residuals when tested with coarse-space functions, and these residuals can pollute the error estimates. By subtracting the coarse-space adjoint from $\boldsymbol{\psi}_h$, the effect of these residuals on the error estimates is minimized [91].

A common approach for obtaining the error indicator is obtained by taking the absolute value of the elemental contributions in Equation 3.17 via

$$\mathcal{E}_e = \left|\mathcal{R}_h\left(\mathbf{u}_H, \delta\boldsymbol{\psi}_h|_e\right)\right|, \tag{3.18}$$

where $\delta\boldsymbol{\psi}_h = \boldsymbol{\psi}_h - \boldsymbol{\psi}_H$. This indicator is computed separately, with absolute values, for each equation in a system of equations and then summed together, possibly yielding a magnitude greater than the actual output error estimate. However, such an indicator is still not an estimate for the actual error due to the previously made approximations.

In DG, localizing the error estimate to each element to compute the elemental error indicator from Equation 3.18 is straightforward, since each degree of freedom is associated with only one element (see Figure 2.5b). However, in SUPG, degrees of freedom on edges and nodes are often associated with multiple elements (see Figure 2.5a). Therefore, the adjoint-weighted residual contributions to the elemental error indicator from edges and nodes need to be restricted based on how many elements they come in contact with. This is done by equally distributing these contributions, as shown in Figure 3.4. This distribution is based on the edge-based averaging approach for error localization outlined in [23]. The weight for the division is equal to the inverse of the number of elements adjacent to the structure.

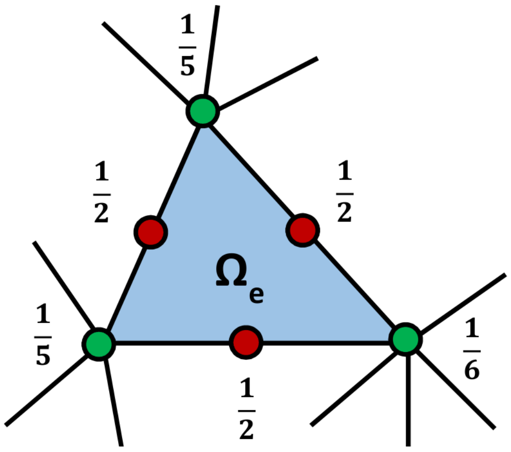


Figure 3.4: Weights for distributing residuals and adjoint-weighted residuals from globally-coupled SUPG degrees of freedom to elements, shown for triangles. The node weights are the inverse of the node cardinality while the face weights are simply $\frac{1}{2}$.

3.2.2 Output-based Error Estimation Implementation

In this section, we will formally define the process for computing the error estimate in Equation 3.18. Recalling from Subsection 3.2.1, we estimate the error between two finite-dimensional spaces. The first space is the coarse approximation space $(\mathcal{V}_H)$ where the state and output are calculated. The second space is a finer space $(\mathcal{V}_h)$ where the error is estimated. The equation and output representations on these spaces are

$$\text{coarse space} : \rightarrow \underbrace{\mathbf{R}_H(\mathbf{U}_H) = \mathbf{0}}_{N_H \text{ equations}} \rightarrow \underbrace{\mathbf{U}_H}_{\text{state} \in \mathbb{R}^{N_H}} \rightarrow \underbrace{J_H(\mathbf{U}_H)}_{\text{scalar output}}$$
$$\text{fine space} : \rightarrow \underbrace{\mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}}_{N_h \text{ equations}} \rightarrow \underbrace{\mathbf{U}_h}_{\text{state} \in \mathbb{R}^{N_h}} \rightarrow \underbrace{J_h(\mathbf{U}_h)}_{\text{scalar output}}$$

The output error estimate of the coarse-space solution relative to the fine-space one is

$$\text{output error: } \delta J \equiv J_h(\mathbf{U}_h) - J_H(\mathbf{U}_H). \tag{3.19}$$

One method of obtaining the fine-space solution is solving for $\mathbf{U}_h$ such that $\mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}$. Depending on the problem, this can be an expensive and therefore impractical option. An alternative approach is prolonging the coarse-space solution into the fine-space by increasing the solution approximation order. This prolonged state, $\mathbf{U}_h^H$, can be defined as

$$\mathbf{U}_h^H \equiv \mathbf{I}_h^H \mathbf{U}_H, \tag{3.20}$$

where $\mathbf{I}_h^H$ is the coarse-to-fine prolongation operator. Note that the state from the coarse-space prolonged to the fine-space does not yield zero fine-space residuals,

$$\mathbf{R}_h\left(\mathbf{U}_h^H\right) \neq \mathbf{0}. \tag{3.21}$$

Therefore, we solve for the prolonged coarse-space solution by computing the following perturbed fine-space problem for $\mathbf{U}_h'$:

$$\mathbf{R}_h\left(\mathbf{U}_h'\right) - \underbrace{\mathbf{R}_h\left(\mathbf{U}_h^H\right)}_{\delta\mathbf{R}_h} = \mathbf{0} \;\;\Rightarrow\;\; \mathbf{U}_h' = \mathbf{U}_h^H. \tag{3.22}$$

Using Equation 3.5, the fine-space adjoint $\mathbf{\Psi}_h$ indicates there will be a perturbation in the output, given the inner product between the adjoint and residual perturbation,

$$\delta J \approx J_h\left(\mathbf{U}_h\right) - J_h\left(\mathbf{U}_h^H\right) = \mathbf{\Psi}_h^T\left(\mathbf{R}_h\left(\mathbf{U}_h\right) - \mathbf{R}_h\left(\mathbf{U}_h^H\right)\right) = -\mathbf{\Psi}_h^T\mathbf{R}_h\left(\mathbf{U}_h^H\right). \tag{3.23}$$

Note that Equation 3.23 is only valid for small perturbations in **U** and **R**.

The fine-space adjoint, $\mathbf{\Psi}_h$, in Equation 3.23 cannot be computed using a prolongation into the fine-space as was done for the solution. Suppose we have a coarse-space adjoint, $\mathbf{\Psi}_H$, prolonged into the fine-space by

$$\mathbf{\Psi}_h^H \equiv \mathbf{I}_h^H\mathbf{\Psi}_H. \tag{3.24}$$

We define the adjoint perturbation as

$$\delta\mathbf{\Psi}_h \equiv \mathbf{\Psi}_h - \mathbf{\Psi}_h^H. \tag{3.25}$$

We then rewrite Equation 3.23 as

$$\delta J \approx -\left(\mathbf{\Psi}_h^H\right)\mathbf{R}_h\left(\mathbf{U}_h^H\right) - \left(\delta\mathbf{\Psi}_h\right)^T\mathbf{R}_h\left(\mathbf{U}_h^H\right). \tag{3.26}$$

The first term in Equation 3.26 is the computable correction and can be used as the sole error estimate. However, it performs poorly, since it uses no new information from the fine-space and is equal to zero for finite-element discretizations with Galerkin orthogonality. We can then rewrite Equation 3.26 as

$$\delta J \approx -\left(\delta\mathbf{\Psi}_h\right)^T\mathbf{R}_h\left(\mathbf{U}_h^H\right), \tag{3.27}$$

which is equivalent to Equation 3.17. Note that now we have to solve for the fine-space adjoint $\mathbf{\Psi}_h$

on the fine-space, which can be computationally expensive. We can solve for $\mathbf{\Psi}_h$ using a cheap iterative smoother at the cost of accuracy, so we try to avoid that if we are only computing the error estimate from $\mathbf{\Psi}_h$.

Localizing the error estimate to each element e, we can compute an elemental error indicator equal to

$$\mathcal{E}_e \approx \left| (\delta \mathbf{\Psi}_h)^T \mathbf{R}_h \left(\mathbf{U}_h^H \right) \right|, \tag{3.28}$$

where the total error estimate $\mathcal{E}$ is given by

$$\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e. \tag{3.29}$$

We use the absolute value elemental error contributions to avoid possible cancellations between elements. Note that Equation 3.28 is equivalent to Equation 3.18.

3.3 Entropy Variables as an Adjoint

The previous sections presented error estimation using output-based adjoints, in which a user prescribes an engineering scalar output to create an adjoint that drives mesh adaptation. This section describes the entropy-based adjoint indicator, which instead uses entropy variables to drive the adaptation. The areas of the mesh targeted by the entropy indicator are those regions that exhibit high net production of *spurious* entropy. The subsequent review of the formulation of the entropy-variable approach is based on the following references [86, 87, 121].

3.3.1 Inviscid Conservation Laws

Consider a steady-state set of inviscid conservation laws in quasi-linear form combined with a scalar entropy conservation law,

$$\mathbf{r}(\mathbf{u}) = \mathbf{A}_i \partial_i \mathbf{u} = \mathbf{0}, \qquad \partial_i F_i = 0, \tag{3.30}$$

where i is the spatial index, $\mathbf{A}_i$ is the inviscid flux Jacobian, $\mathbf{u}$ is the state vector, and $F_i(\mathbf{u})$ is the entropy flux associated with an entropy function $U(\mathbf{u})$. Both the entropy function and flux satisfy the compatibility relation $U_{\mathbf{u}} \mathbf{A}_i = (F_i)_{\mathbf{u}}$, where the subscripts denote differentiation. For the convex entropy function U, the entropy variables corresponding to that function can be defined as $\mathbf{v} = U_{\mathbf{u}}^T$. An important characteristic of the entropy variables is that they symmetrize the conservation laws in the sense that [69, 109]:

- The matrix $\mathbf{A}_i\mathbf{u}_\mathbf{v}$ is symmetric.
- The transformation Jacobian matrix, $\mathbf{u}_\mathbf{v}$, is not only symmetric, but also positive definite.

With these properties established, the conservation law established in Equation 3.30 can be expressed as:

$$\mathbf{0} = \mathbf{A}_i\partial_i\mathbf{u} = \mathbf{A}_i\mathbf{u}_\mathbf{v}\partial_i\mathbf{v} = \mathbf{u}_\mathbf{v}\mathbf{A}_i^T\partial_i\mathbf{v} \quad \Rightarrow \quad \mathbf{A}_i^T\partial_i\mathbf{v} = \mathbf{0}. \tag{3.31}$$

The reason for manipulating the conservation law to the above expression is to create the appearance of the transpose of the inviscid flux Jacobian. This indicates that the expression is a continuous adjoint equation for an output that has no domain boundary integral contribution. In other words, this means that the entropy variables act as adjoint solutions in the above expression. From Subsection 3.1.1, continuous adjoints are Green's functions that indicate how much residual perturbations affect the scalar output J. This can be expressed in a quasi-linear form, similar to how the conservation laws were presented in Equation 3.30, as follows:

$$\delta\mathbf{r} = \mathbf{A}_i\partial_i\delta\mathbf{u}. \tag{3.32}$$

Using an inner product of the adjoint with the residual perturbations, the output perturbation can be found via the following analysis that relies on integration by parts to directly relate the output to the transport of entropy:

$$\begin{aligned}\delta J &= \int_\Omega \mathbf{v}^T\delta\mathbf{r}\,d\Omega \\ &= \int_\Omega \mathbf{v}^T\underbrace{\mathbf{A}_i\partial_i\delta\mathbf{u}}_{\delta\mathbf{r}}\,d\Omega.\end{aligned} \tag{3.33}$$

Using integration by parts on Equation 3.33 produces

$$\begin{aligned}\delta J &= \underbrace{-\int_\Omega \partial_i\mathbf{v}^T\mathbf{A}_i\delta\mathbf{u}\,d\Omega}_{=0 \text{ by Eqn.3.31}} + \int_{\partial\Omega}\underbrace{\mathbf{v}^T\mathbf{A}_i}_{(F_i)_\mathbf{u}}\delta\mathbf{u}n_i\,ds \\ &= 0 + \int_{\partial\Omega}(F_i)_\mathbf{u}\delta\mathbf{u}n_i\,ds \\ &= \delta\left[\underbrace{\int_{\partial\Omega}F_in_i\,ds}_{J}\right].\end{aligned} \tag{3.34}$$

This derivation indicates that the output associated with this adjoint equation is a measure of the net entropy transport through the domain boundary. This indicates that the entropy variables serve as the adjoint solution to an output corresponding to the total entropy flow out of the domain. This adjoint can be used in Equation 3.18 to produce an error indicator driven by areas of the mesh with spurious entropy generation, i.e. those that are important for the prediction of the net entropy outflow.

3.3.2 Viscous Conservation Laws

Now we consider a set of viscous conservation laws in quasi-linear form combined with a scalar entropy conservation law,

$$\mathbf{r}(\mathbf{u}) = \mathbf{A}_i \partial_i \mathbf{u} - \partial_i \left(\mathbf{K}_{ij} \partial_j \mathbf{u}\right) = \mathbf{0}, \tag{3.35}$$

where i is the spatial index, $\mathbf{A}_i \partial_i \mathbf{u}$ is the inviscid flux and $-\mathbf{K}_{i,j} \partial_j \mathbf{u}$ is the viscous flux. The entropy variables are still defined as $\mathbf{v} = U_{\mathbf{u}}^{\mathrm{T}}$ and the two assumptions regarding how they symmetrize the conservation laws defined in Subsection 3.3.1 still hold. In addition, the entropy variables $\mathbf{v}$ also symmetrize $\mathbf{K}_{ij}$, in the sense that $\widetilde{\mathbf{K}}_{ij} = \widetilde{\mathbf{K}}_{ji}^{\mathrm{T}}$, where $\widetilde{\mathbf{K}}_{ij} = \mathbf{K}_{ij} \mathbf{u}_{\mathbf{v}}$ [109]. Substituting $\partial_i \mathbf{u} = \mathbf{u}_{\mathbf{v}} \partial_i \mathbf{v}$ into Equation 3.35, along with taking the transpose, yields the following equation for the entropy variables,

$$\partial_i \mathbf{v}^T \mathbf{A}_i \mathbf{u}_{\mathbf{v}} - \partial_i \left(\partial_j \mathbf{v}^T \widetilde{\mathbf{K}}_{ji}\right) = \mathbf{0}. \tag{3.36}$$

Due to the sign of the second term being negative, this is no longer a strict mathematical adjoint to the primal equation. The entropy variables still represent the sensitivity between the scalar output and residual perturbations, although the output is not solely the net transport of entropy through the domain boundary as it was in Equation 3.34. It now includes terms related to viscous dissipation.

From the viscous conservation law in Equation 3.35, changes in the state and the residual are related via

$$\delta \mathbf{r} = \mathbf{A}_i \partial_i \delta \mathbf{u} - \partial_i \left(\mathbf{K}_{ij} \partial_j \delta \mathbf{u}\right). \tag{3.37}$$

Just as in the previous subsection, the output perturbation can be found by starting with an inner product of the adjoint with the residual perturbations expressed in Equation 3.37 as follows:

$$\begin{aligned}\delta J &= \int_\Omega \mathbf{v}^T \delta \mathbf{r} d\Omega \\ &= -\int_\Omega \partial_i \mathbf{v}^T \mathbf{A}_i \delta \mathbf{u} d\Omega + \int_{\partial\Omega} \mathbf{v}^T \mathbf{A}_i \delta \mathbf{u} n_i ds + \int_\Omega \partial_i \mathbf{v}^T \mathbf{K}_{ij} \partial_j \delta \mathbf{u} d\Omega - \int_{\partial\Omega} \mathbf{v}^T \mathbf{K}_{ij} \partial_j \delta \mathbf{u} n_i ds.\end{aligned} \tag{3.38}$$

The second term in the above equation can be directly related to results from Equation 3.34 to produce

$$\begin{aligned}\delta J = &-\int_\Omega \partial_i \mathbf{v}^T \mathbf{A}_i \delta \mathbf{u} d\Omega + \delta \left[\int_{\partial\Omega} F_i n_i ds \right] \\ &+ \int_\Omega \partial_i \mathbf{v}^T \mathbf{K}_{ij} \partial_j \delta \mathbf{u} d\Omega - \int_{\partial\Omega} \mathbf{v}^T \mathbf{K}_{ij} \partial_j \delta \mathbf{u} n_i ds.\end{aligned} \tag{3.39}$$

Using the relation $\partial_i \mathbf{u} = \mathbf{u}_\mathbf{v} \partial_i \mathbf{v}$ and Equation 3.36, the above expression can be further modified to produce

$$\begin{aligned}\delta J = &-\int_\Omega \partial_i (\partial_j \mathbf{v}^T \widetilde{\mathbf{K}}_{ji}) \delta \mathbf{v} d\Omega + \delta \left[\int_{\partial\Omega} F_i n_i ds \right] \\ &+ \int_\Omega \partial_i \mathbf{v}^T \mathbf{K}_{ij} \partial_j \delta \mathbf{v} d\Omega - \int_{\partial\Omega} \mathbf{v}^T \mathbf{K}_{ij} \partial_j \delta \mathbf{v} n_i ds.\end{aligned} \tag{3.40}$$

Integrating the first term in the above equation by parts and performing further grouping and simplifications yields

$$\begin{aligned}\delta J &= \delta \left[\int_{\partial\Omega} F_i n_i ds \right] + \int_\Omega \left(\partial_i \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \delta \mathbf{v} + \partial_i \delta \mathbf{v} \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} \right) d\Omega \\ &\qquad - \int_{\partial\Omega} \left(\mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \delta \mathbf{v} + \delta \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} \right) n_i ds \\ &= \delta \left[\int_{\partial\Omega} F_i n_i ds + \int_\Omega \partial_i \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} d\Omega - \int_{\partial\Omega} \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} n_i ds \right].\end{aligned} \tag{3.41}$$

Consequently, the entropy variables act as an adjoint for the output

$$J = \int_{\partial\Omega} F_i n_i ds + \int_\Omega \partial_i \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} d\Omega - \int_{\partial\Omega} \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} n_i ds. \tag{3.42}$$

Each of the terms in the above equation has its own physical meaning, as shown in Figure 3.5. The first term is simply the convective outflow of entropy across the domain boundary. The second term is the generation of entropy due to viscous dissipation within shear layers, vortices, or across shocks, while the third term is the entropy diffusion across the boundary.

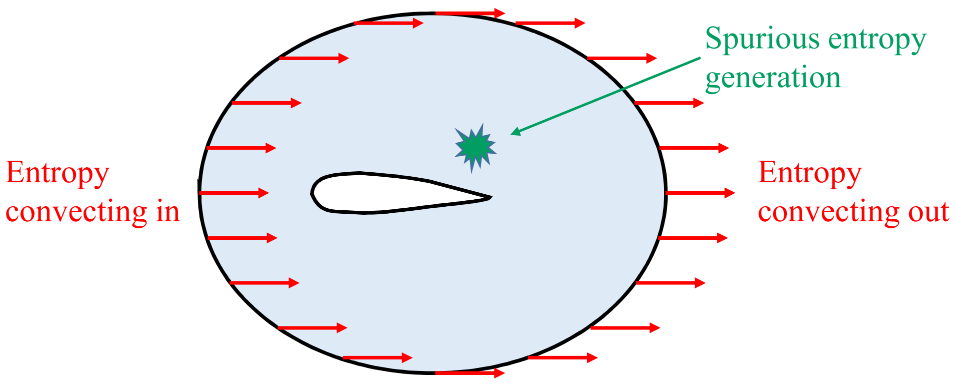


Figure 3.5: Output associated with the adjoint equation when using entropy variables.

3.3.3 Entropy Function

The entropy function that yields entropy variables that symmetrize both the inviscid and viscous terms in the compressible Navier-Stokes equations (with heat-conduction included) is unique up to additive and multiplicative constants [109],

$$U = -\rho S / \left(\gamma - 1\right), \qquad S = \ln p - \gamma \ln \rho, \tag{3.43}$$

where p is the pressure, ρ is the density, γ is the ratio of specific heats, and S is the physical entropy. Differentiating U with respect to the conservative state $\mathbf{u}$ yields the entropy variables,

$$\mathbf{v} = U_{\mathbf{u}}^T = \left[\frac{\gamma - S}{\gamma - 1} - \frac{1}{2}\frac{\rho V^2}{p}, \ \frac{\rho u_i}{p}, \ -\frac{\rho}{p}\right]^T, \tag{3.44}$$

where $V^2 = u_i u_i$ is the square of the velocity magnitude, $p = (\gamma - 1)(\rho E - \rho V^2/2)$ is the pressure, and E is the total energy per unit mass. The entropy variables are obtained via a nonlinear transformation of the conservative variables. The corresponding entropy flux function is $F_i = u_i U$.

Adapting on J using the entropy variables as adjoints constitutes output-based adaptation. Since entropy variables can be computed on the fine-space directly from the state, adapting using entropy variables instead of the output-based adjoint is far less computationally expensive. However, this indicator does not disregard areas of spurious entropy generation that do not affect a particular engineering output. This may lead to over-refinement, particularly for cases with flow discontinuities or unresolved features that propagate downstream (e.g. vortices).

3.3.4 Error Estimation Using Entropy Variables

With the entropy variables $\mathbf{v}$ known, an entropy transport equation can be derived from the Navier-Stokes equations [51, 87, 122]:

$$\underbrace{\partial_i F_i + \partial_i \mathbf{v}^T \mathbf{K}_{ij} \partial_j \mathbf{u} - \partial_i \left(\mathbf{v}^T \mathbf{K}_{ij} \partial_j \mathbf{u} \right)}_{\mathbf{v}^T \mathbf{r}(\mathbf{u})} = \mathbf{0}. \tag{3.45}$$

The residual of this equation indicates how well the entropy transport equation is satisfied. Taking the inner product of the residual in Equation 3.35 with the entropy variables yields the entropy transport equation. This means that the entropy-variable weighted primal residual used for the output error estimate in Section 3.2 is the entropy residual. The adaptive error indicator based on entropy variables can be obtained by substituting the fine-space output-based adjoint, ψ, with the entropy variables computed on the fine-space, $\mathbf{v}_h$ in Equation 3.18 as follows

$$\mathcal{E}_e = \left| \mathcal{R}_h \left(\mathbf{u}_H, \delta \mathbf{v}_h|_e \right) \right|. \tag{3.46}$$

To compute the entropy variables on the fine-space, in this work, the state $\mathbf{u}_H$ is computed on a finer approximation space, $\mathbf{u}_h \in \mathcal{V}_h$ using a least-squares projection. From there, the entropy variables are computed from that state on the fine-space using Equation 3.44.

3.4 Combined Approach in Steady Simulations

In this section, the combined approach is explored for steady-state simulations. There are three different methodologies used to combine the indicators. In the first approach, one error indicator is obtained from the fine-space output-based adjoint using Equation 3.18 and another is obtained with entropy variables on the fine-space using Equation 3.46. Those indicators are obtained for each element and multiplied together to yield a new error indicator. In the second approach, the error indicator based on the output-based adjoint is computed using the adjoint calculated on the coarse-space. No fine-space adjoint is computed. The last approach involves implementing a mask on the indicator obtained using entropy variables. The mask is obtained using the aforementioned error indicator obtained on the coarse-space. The details in this section borrow from the discussion included in the following reference [121].

3.4.1 Combined Approach Using a Fine Output-based Adjoint

In this version of the combined approach, indicators are obtained seperately using the fine-space adjoint, ψ_h, and the fine-space entropy variables, $\mathbf{v}_h$, and then combined through direct, elemental multiplication. For an element e, we can express that mathematically using the elemental indicators obtained from Equation 3.18 and Equation 3.46 via

$$\mathcal{E}_{e,\text{combined}} = \mathcal{E}_{e,\text{output}} \cdot \mathcal{E}_{e,\text{entropy}}, \tag{3.47}$$

where $\mathcal{E}_{e,\text{output}}$ is the elemental indicator obtained using the output-based adjoint and $\mathcal{E}_{e,\text{entropy}}$ is the elemental indicator obtained using the entropy variables.

The decision to combine the indicators through multiplication is not the only choice, as a certain amount of heuristics is involved. There are many possible options beyond simple multiplication. The two indicators could have simply been combined using addition. However, elemental multiplication was chosen instead of addition in order to only target adaptation regions of the domain important to both indicators. Both the indicators based on the output-based adjoint and the indicators based on entropy variables overreact and are susceptible to spurious refinement. However, these regions are not always shared between the two approaches. By multiplying the indicators together, purely spurious regions of one indicator do not dominate. Mathematical addition of the indicators would not do this well: e.g. in the limit of one indicator going to zero, an element could still be refined if the other indicator was high. Another drawback of addition is the mixing of units.

While multiplication was always performed to generate the final combined indicator, various modifications for obtaining the two indicators were tested. These modifications were made in the interest of improving adaptive efficiency and reducing the overall computational cost.

3.4.2 Combined Approach Using a Coarse Output-Based Adjoint

The computation of the error indicator in Equation 3.18 requires computing a fine-space output-based adjoint ψ_h on a finer finite-dimensional space, $\mathcal{V}_h$. The details of why this is necessary to compute an accurate output-based error estimate were discussed in Subsection 3.2.2. Unfortunately, this computation of the fine-space adjoint is expensive and makes the combined approach discussed in Subsection 3.4.1 even more expensive than the typical output-based adjoint approach.

To minimize the cost of this step, a new approach to combining the indicators is implemented in which no fine-space output-based adjoint is computed. Instead, the output-based adjoint at the current approximation order is projected down one order from $\mathcal{V}_H$ to $\mathcal{V}_{\tilde{H}}$ and then prolonged back up one order to $\mathcal{V}_H$ before obtaining the output-based indicator. This can be expressed mathematically as

$$\mathbf{\Psi}_H^{\tilde{H}} \equiv \mathbf{I}_H^{\tilde{H}} \mathbf{I}_{\tilde{H}}^H \mathbf{\Psi}_H, \tag{3.48}$$

where $\mathbf{\Psi}_H^{\tilde{H}}$ is the projected adjoint at the approximation space $\mathcal{V}_{\tilde{H}}$. Therefore, no new fine-space adjoint needs to be obtained since the fine-space is now at the original approximation order, i.e. in $\mathcal{V}_H$. We still subtract the output-based adjoint at the coarser space, $\mathcal{V}_{\tilde{H}}$, from the output-based adjoint at the current approximation space, $\mathcal{V}_H$, to eliminate pollution from p-dependence of the weak form and non-converged residuals. This leads to the following modified output-based adjoint

$$\delta\mathbf{\Psi}_H \equiv \mathbf{\Psi}_H - \mathbf{\Psi}_H^{\tilde{H}}. \tag{3.49}$$

We then use this modified output-based adjoint to solve for the elemental error indicator in Equation 3.28,

$$\mathcal{E}_{e,\text{output}} \approx \left| (\delta\mathbf{\Psi}_H)^T \mathbf{R}_H \left(\mathbf{U}_H^{\tilde{H}}\right) \right|, \tag{3.50}$$

where $\mathbf{U}_H^{\tilde{H}}$ is the projection of the primal from the space H at order p, to space $\tilde{H}$ at order $p-1$, and then back up to space H at order p. This step is necessary since the residual $\mathbf{R}_H$ of the current-space solution $\mathbf{U}_H$ is equal to zero. By projecting the state down to order $p-1$ and then back up to order p, we obtain an approximation for the error in the state that allows us to find the elemental error indicator in Equation 3.50.

The entropy variables on the fine-space, $\mathbf{v}_h$, are still used to obtain the entropy-based indicator. These two indicators can be combined as in Equation 3.47 despite the fact they are not computed using adjoints at the same approximation space. This is possible because Equation 3.18 includes the restriction of the adjoints to the elements in the current approximation space, $\mathcal{V}_H$. Since the output-based indicator should still target similar general regions of the domain where more refinement is necessary, this less-expensive approach should ideally not lead to significant deterioration of the adaptive performance.

3.4.3 Combined Approach Using a Mask

Since the error indicator based on the output-based adjoint in the combined approach from the previous subsection was not obtained on a finer computational space, pollution of the indicator is possible. To mitigate this issue, an additional approach is detailed in this subsection that relies on the addition of a mask on the entropy-based adjoint. The purpose of the mask is to limit the weight of the entropy indicator on the mesh adaptation so that the output-based indicator carries more weight. This is accomplished by creating an element-based refinement indicator, i.e. a mask, based on the output-based adjoint error indicator. This mask, μ_e, is a vector that is the same size as the output-based indicator $\mathcal{E}_{e,\text{output}}$. The mask is created by examining the relative magnitude of the indicator for each element. Using a user-specified percentage, a subset of the output-based indicator is created. For example, if the user wishes to apply a 25% mask on the entropy-based indicator, μ_e is created from the output indicator: it consists of 1 for the 25% elements with the largest $\mathcal{E}_{e,\text{output}}$, and 0 for all other elements. The subset of $\mathcal{E}_{e,\text{output}}$ whose relative magnitude falls in the top 25% will be assigned to the mask set, as $\mathcal{E}_{\text{mask}}$. The mask is then created by looping over all of the elements and setting the magnitude of the indicator to 1 if the elemental indicator falls within the subset $\mathcal{E}_{\text{mask}}$ and 0 if it does not. This can be expressed mathematically as

$$\mu_e = \begin{cases} 1 & \text{if } \mathcal{E}_{e,\text{output}} \subset \mathcal{E}_{\text{mask}} \\ 0 & \text{if } \mathcal{E}_{e,\text{output}} \not\subset \mathcal{E}_{\text{mask}}. \end{cases} \tag{3.51}$$

The mask is then multiplied by the entropy-based indicator in an element-wise fashion to obtain the final indicator that ultimately governs the mesh adaptation,

$$\mathcal{E}_{e,\text{combined}} = \mu_e \cdot \mathcal{E}_{e,\text{entropy}}. \tag{3.52}$$

The reason for using a masking approach is that the mask should eliminate elements targeted by the entropy-based indicator which have little influence on the desired engineering output, regardless of the magnitude of the entropy-based indicator on these elements. This will lead to a different set of elements chosen for refinement compared to the standard approach in Equation 3.47.

3.5 Adaptation Strategy

The previous three sections outline the different processes used in this work to generate localized error estimates on each element of the mesh. A meshing algorithm then uses these error estimates to iteratively modify the mesh to equidistribute (and ultimately decrease) the total error. The following is a sequential breakdown of the various steps implemented for each adaptive iteration for steady-state simulations:

1. Solve the primal problem $\mathbf{R}_H(\mathbf{U}_H) = \mathbf{0}$ on the initial mesh at order p to obtain the coarse-space solution $\mathbf{U}_H$. Use the solution process outlined in Subsections 2.2.4 and 2.2.5.
2. If the adaptation depends on an engineering output, first solve for the output $J_H(\mathbf{U})$. Then, solve the coarse-space adjoint problem:

$$\left(\frac{\partial \mathbf{R}_H}{\partial \mathbf{U}_H}\right)^T \mathbf{\Psi}_H + \left(\frac{\partial J_H}{\partial \mathbf{U}_H}\right)^T = \mathbf{0}.$$

3. Prolong the coarse-space solution to the fine-space:

$$\mathbf{U}_h^H \equiv \mathbf{I}_h^H \mathbf{U}_H.$$

4. Prolong the coarse-space adjoint to the fine-space:

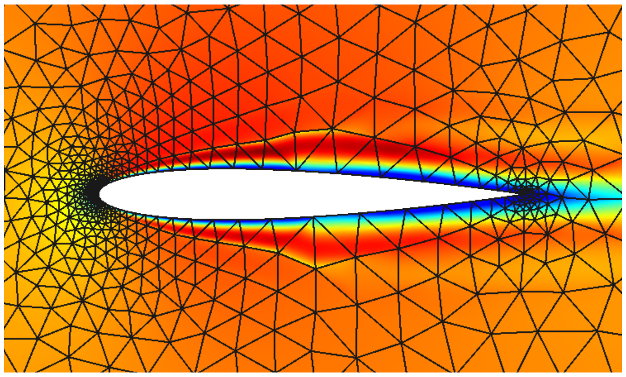
$$\mathbf{\Psi}_h^H \equiv \mathbf{I}_h^H \mathbf{\Psi}_H.$$

5. Solve for the fine-space residuals using the prolonged solution, $\mathbf{R}_h\left(\mathbf{U}_h^H\right)$.
6. If the adaptation depends on an engineering output, solve or iterate the fine-space adjoint problem to obtain $\mathbf{\Psi}_h$:

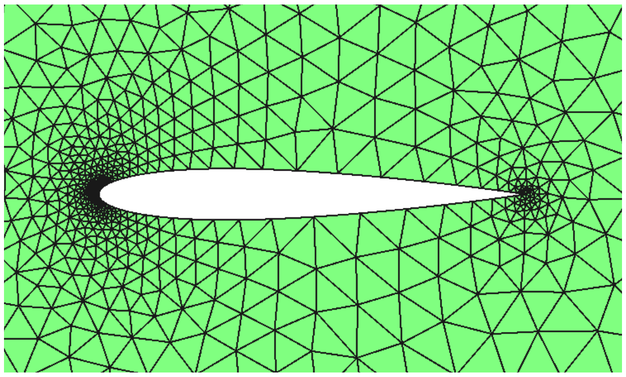
$$\left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right)^T\Bigg|_{\mathbf{U}_h^H} \mathbf{\Psi}_h + \left(\frac{\partial J_h}{\partial \mathbf{U}_h}\right)^T\Bigg|_{\mathbf{U}_h^H} = \mathbf{0}.$$

   If the adaptation depends on entropy variables, compute $\mathbf{v}_h\left(\mathbf{U}_h\right)$ using Equation 3.44 by either solving the fine-space problem exactly or iteratively smoothing ν_{fine} times to obtain $\mathbf{U}_h$.
7. On each element calculate the adaptive indicator, $\mathcal{E}_e$, using Equation 3.28 with $\mathbf{\Psi}_h$ if the adaptation depends on an engineering output, or $\mathbf{v}_h$ if the adaptation depends on entropy variables. For the combined approach, compute indicators based on both the output adjoint and the entropy variables, and then combine them (e.g. via an element-wise multiplication) to obtain a new indicator.
8. Refine the mesh on the elements with the largest indicator using one of the strategies outlined in Chapter 5.
9. Initialize the solution on the adapted mesh with the solution from the previous mesh and return to the first step in this process.

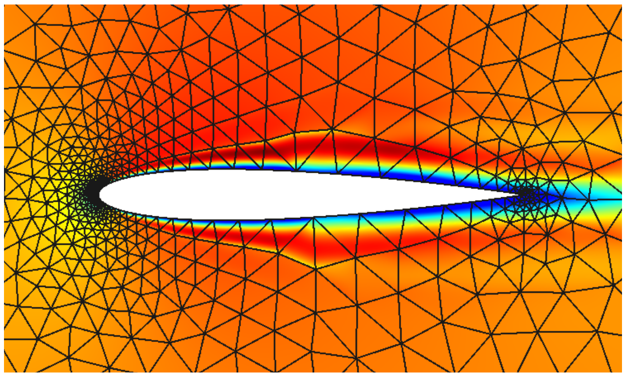
A graphical representation of the aforementioned process for computing the error indicator is outlined in Figure 3.6 for subsonic flow over a NACA 0012 airfoil using the RANS equations. The freestream Mach number is 0.5, the Reynolds number is 5000, and the airfoil angle of attack is $1°$. The output of interest, J, is the total drag on the airfoil. In this example, the output-based adjoint approach is used to compute the error indicator. Since the solution $\mathbf{U}_H$ is converged, the residuals in Figure 3.6b are zero. By prolonging the state into the fine space, we uncover fine-space residuals that can be used to compute the elemental error indicators. In Figure 3.6d, the fine-space residuals are particularly high near the top and bottom surface of the airfoil. Since the adjoint is sensitive there as well (as visible in Figure 3.6e), the elements within these regions are marked for refinement. Consequently, the magnitude of the error indicator for these elements is high, as shown in Figure 3.6f.



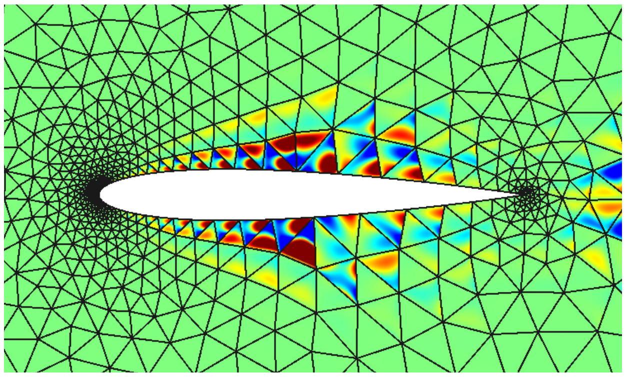
(a) Coarse-space state, $\mathbf{U}_H$



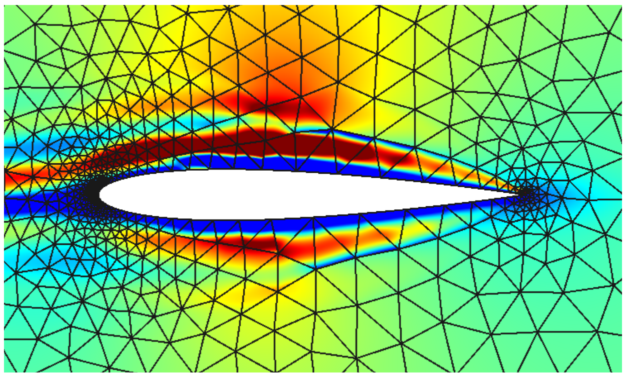
(b) Coarse-space residual, $\mathbf{R}_H\left(\mathbf{U}_H\right)$



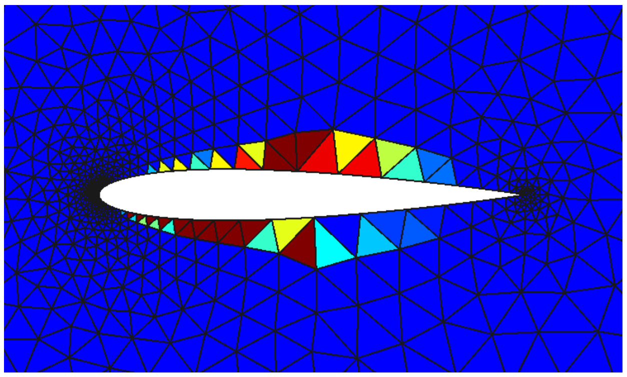
(c) Prolonged state, $\mathbf{U}_h^H$



(d) Fine-space residual, $\mathbf{R}_h\left(\mathbf{U}_h^H\right)$



(e) Fine-space adjoint, $\mathbf{\Psi}_h$



(f) Error Indicator, $\mathcal{E}_e$

Figure 3.6: Summary of the error estimation process using the output-based adjoint approach for $M_\infty = 0.5$, $\alpha = 1^\circ$, $Re = 5000$ flow over a NACA 0012 airfoil.

CHAPTER 4

Unsteady Adjoint and Error Estimation

This chapter is analogous to Chapter 3 in that the chapter's primary focus is to review the three error estimation approaches previously outlined. However, the context of this chapter is unsteady problems. We begin with an overview of the unsteady discrete adjoint and how its computation contributes to significant computation cost. This leads to a discussion of the continuous-in-time output-based adjoint and how it can be used to compute both spatial and temporal errors. A detailed discussion of a cost model that determines how to distribute those errors for the mesh adaptation follows. Next is a detailed description of how entropy variables can act as an adjoint, this time in the context of unsteady conservation laws. The chapter concludes with a review of the various novel combined approaches for unsteady simulations and how they incorporate the spatial and temporal errors indicators into the respective combination logic for each approach.

4.1 Unsteady Output-based Adjoint

In Section 3.1, the steady-state discrete adjoint was derived starting from the general residual equation $\mathbf{R}(\mathbf{U}) = \mathbf{0}$, see Equation 3.1. To derive the unsteady discrete adjoint, we start from a semi-discrete form of the equation where the PDE has already been discretized in space:

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{4.1}$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix,

$$\mathbf{M}_{ij} = \mathbf{I}_s \int_\Omega \phi_i \phi_j d\Omega. \tag{4.2}$$

In the above equation, $\mathbf{I}_s \in \mathbb{R}^{s\times s}$ is the identity matrix and $1 \leq i, j \leq N$ encompass all global degrees of freedom. To discretize the unsteady term in Equation 4.1, we assume (for simplicity) a backward Euler method, which can be written as

$$\frac{d\mathbf{U}}{dt} \approx \frac{\mathbf{U}^m - \mathbf{U}^{m-1}}{\Delta t}, \tag{4.3}$$

where δt is the time step. Using this approximation, the fully-discrete form representing the unsteady residual associated with the m^{th} temporal degree of freedom can be expressed as

$$\underbrace{\mathbf{M}\frac{\mathbf{U}^m - \mathbf{U}^{m-1}}{\Delta t} + \mathbf{R}\left(\mathbf{U}^m\right)}_{\bar{\mathbf{R}}^m(\mathbf{U})} = \mathbf{0}, \tag{4.4}$$

where $\bar{\mathbf{R}}^m$ is the unsteady residual at t^m. Starting from an initial time of $t = t_0$, several temporal iterations are required before the unsteady residual, $\bar{\mathbf{R}}(\mathbf{U})$, drops below a given tolerance, in magnitude. These N_t unsteady residual vectors must be driven to zero sequentially, requiring a computation of the Newton-Raphson method, outlined in Subsection 2.2.5, at each step.

Using a local sensitivity analysis similar to that in Equation 3.3, a chain of dependence between inputs and the scalar output can be written as

$$\underbrace{\boldsymbol{\mu}}_{\text{inputs}} \rightarrow \underbrace{\bar{\mathbf{R}}^m(\mathbf{U}, \boldsymbol{\mu})}_{\text{residuals}} = \mathbf{0} \rightarrow \underbrace{\mathbf{U}}_{\text{state}} \rightarrow \underbrace{J(\mathbf{U})}_{\text{output}}. \tag{4.5}$$

The process used to derive the discrete steady adjoint in Section 3.1 can also be used for the unsteady adjoint derivation. The only difference is that the state vectors are much larger since they include the temporal components ($\mathbf{U} \in \mathbb{R}^{N \times N_t}$). The unsteady adjoint equation can be written as [95]

$$\sum_{m=1}^{N_t} \left(\frac{\partial \bar{\mathbf{R}}^m}{\partial \mathbf{U}}\right)^T \boldsymbol{\Psi}^m + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = \mathbf{0}. \tag{4.6}$$

Due to the transpose on the unsteady Jacobian matrix in Equation 4.6, the ideal approach is solving the adjoint system by marching backwards in time. Given a backwards Euler temporal discretization, the unsteady Jacobian, $\partial\bar{\mathbf{R}}^m/\partial\mathbf{U}$, and its transpose will look like ($* = N \times N$ block),

$$\begin{array}{cc}
\text{Primal Unsteady Jacobian} & \text{Adjoint Unsteady Jacobian} \\
\dfrac{\partial \bar{\mathbf{R}}^m}{\partial \mathbf{U}} = \begin{bmatrix} * & & & & & & \\ * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & * \end{bmatrix} &
\left(\dfrac{\partial \bar{\mathbf{R}}^m}{\partial \mathbf{U}}\right)^T = \begin{bmatrix} * & * & & & & & \\ & * & * & & & & \\ & & * & * & & & \\ & & & * & * & & \\ & & & & * & * & \\ & & & & & * & * \\ & & & & & & * \end{bmatrix}
\end{array}$$

Since the adjoint Jacobian is zero below the main diagonal, backward substitution is the necessary strategy for solving the temporal discretization. Note that the primal state is required for the computation of both primal and adjoint unsteady Jacobians. Therefore, the adjoint cannot be computed until an approximation of the primal state **U** is known. For this reason, the state must either be stored to the disk or recomputed for every backward time step in the approximation of the unsteady adjoint. Once the unsteady adjoint solution is known, the sensitivities can be calculated as

$$\frac{\partial J}{\partial \boldsymbol{\mu}} = \sum_{m=1}^{N_t} (\boldsymbol{\Psi}^m)^T \left(\frac{\partial \bar{\mathbf{R}}^m}{\partial \boldsymbol{\mu}}\right). \tag{4.7}$$

It is obvious from the analysis in this section that unsteady problems create significant implementation challenges and computational costs for output-based methods since the solution of the unsteady adjoint in Equation 4.6 must be marched backward in time. Since the adjoint requires the linearization of the state at each time step, there is a significant increase in the computational cost for both storage and computational time.

4.2 Error Estimation Using Unsteady Output-Based Adjoints

In this section, we describe an unsteady output-based error estimation approach using a continuous-in-time adjoint and mesh optimization technique that relies on the separation of the error between spatial and temporal discretizations. Many previous works [79, 83–85, 96] use a variational discretization, namely the finite element method in space and time, where the discrete adjoint from Equation 4.6 is obtained systematically from the primal system by transposing the operator. However, these methods are expensive and not as common as multi-step and multi-stage integrators. Recall from Section 3.2, that computing the continuous adjoint for steady-state simulations was essential for computing output-error estimates. Without a clear connection between the discrete and continuous adjoint for unsteady problems, it can be difficult to compute the adjoint-weighted residual error estimate.

Instead of using variational time-marching methods, this work relies on multi-step and multi-stage methods (see Appendix C) for solving time integration of unsteady problems. These methods are beneficial due to their relatively lower computational cost and overall ease of implementation. They are particularly advantageous in the context of unsteady error estimation because they accommodate continuous-in-time adjoints. The focus of this section is the derivation of the continuous, unsteady output-based adjoint how it can be used for error estimation. A detailed description of how the error contributions are separated between temporal and spatial components and how they can be used to optimize the spatial mesh is also included. The details in this section are primarily based on [106, 123].

4.2.1 Continuous, Unsteady Outut-Based Adjoint

The backward-in-time propagation of the unsteady adjoint information can also be revealed by analysis of the continuous unsteady adjoint. Recalling from Subsection 3.3.1, a continuous adjoint is a Green's function that relates residual source perturbations in the original equation to an associated output of interest. Consider an unsteady output of the form

$$\bar{J} \equiv \int_0^{T_f} J(\mathbf{U}(t), t)dt + J_{T_f}(\mathbf{U}(T_f)), \tag{4.8}$$

where J and J_T are functionals of the spatial distribution of the state via the discrete coefficients $\mathbf{U}$. It is important to note that J_{T_f} is the function of only the final-time state, $\mathbf{U}(T_f)$. To derive the unsteady adjoint equation, a Lagrangian is defined as

$$\mathcal{L} \equiv \bar{J} + \int_0^{T_f} \mathbf{\Psi}^T \bar{\mathbf{R}} dt = \bar{J} + \int_0^{T_f} \mathbf{\Psi}^T \left(\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) \right) dt. \tag{4.9}$$

Integrating the first term in the integral by parts, substituting the unsteady form of the output from Equation 4.8, and forcing stationarity of the Lagrangian with respect to allowable state variations, $\delta\mathbf{U}$, yields

$$\frac{dJ_{T_f}}{d\mathbf{U}}\delta\mathbf{U}\bigg|_{t=T_f} + \mathbf{\Psi}^T\mathbf{M}\delta\mathbf{U}\bigg|_{t=T_f} - \mathbf{\Psi}^T\mathbf{M}\delta\mathbf{U}\bigg|_{t=T_f} + \int_0^{T_f} \left[\frac{\partial J}{\partial \mathbf{U}} - \frac{d\mathbf{\Psi}^T}{dt}\mathbf{M} + \mathbf{\Psi}^T\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right] \delta\mathbf{U} dt = \mathbf{0}. \tag{4.10}$$

Since the initial condition on the primal state fully constrains the state, meaning $\delta\mathbf{U} = \mathbf{0}$ at $t = 0$, the middle term falls out. After taking the transpose of the time integrand term, we are left with the unsteady adjoint equation,

$$-\mathbf{M}\frac{d\mathbf{\Psi}}{dt} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \mathbf{\Psi} + \frac{\partial J^T}{\partial \mathbf{U}} = \mathbf{0}, \tag{4.11}$$

subject to the terminal condition

$$\mathbf{\Psi}(T_f) = -\mathbf{M}^{-1}\frac{d\left(J_{T_f}\right)^T}{d\mathbf{U}}. \tag{4.12}$$

The terminal condition makes it necessary to solve for the adjoint backward in time using any time-integration scheme. We can define a reverse time variable, $\tau = T_f - t$, in order to rewrite Equation 4.11 as

$$\mathbf{M}\frac{d\mathbf{\Psi}}{d\tau} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\mathbf{\Psi} + \frac{\partial J^T}{\partial \mathbf{U}} = \mathbf{M}\frac{d\mathbf{\Psi}}{d\tau} + \mathbf{R}^{\mathbf{\Psi}}(\mathbf{U}, \mathbf{\Psi}) = \mathbf{0}, \tag{4.13}$$

where $\mathbf{R}^{\mathbf{\Psi}}$ is the adjoint residual. We can solve Equation 4.13 backward in time for the adjoint using the same techniques we used to solve for the primal state, which are documented in Appendix C.

4.2.2 Output Error Estimation

Given a particular engineering output, $\bar{J}$, the adjoint solution can be used to estimate numerical errors and drive the mesh adaption using the adjoint-weighted residual method from Subsection 3.2.2. For unsteady simulations, both the temporal and spatial errors must be quantified. Denoting by $\mathbf{U}_H(t)$ the approximate primal solution obtained from a given time integration method and time step size, we can use the unsteady adjoint solution $\mathbf{\Psi}(t)$ to estimate the error in $\bar{J}$,

$$\delta \bar{J} = \bar{J}\left(\mathbf{U}_H\right) - \bar{J}\left(\mathbf{U}\right) \approx \frac{\partial \bar{J}}{\partial \mathbf{U}}\delta\mathbf{U} \approx -\int_0^{T_f} \mathbf{\Psi}^T \bar{\mathbf{R}}(\mathbf{U}_H) dt, \tag{4.14}$$

where $\delta\mathbf{U} \equiv \mathbf{U}_H - \mathbf{U}$ is the error in the state and $\bar{\mathbf{R}}(\mathbf{U}_H) \approx \frac{\partial \bar{\mathbf{R}}}{\partial \mathbf{U}}(\delta\mathbf{U})$ is the generally nonzero unsteady residual obtained from the approximate primal. The fine-space adjoint $\mathbf{\Psi}_h$ is calculated using a high-order time integration method, denoted by the subscript h. In this work, we use DIRK3 or DIRK4. Numerical error due to the spatial discretization is measured by refining the spatial discretization. This is done by increasing the spatial approximation order of the discretization by one.

Denoting $\mathbf{U}_h^H(t)$ as the prolongation of the primal from space H to space h, the unsteady output error can be estimated through an adjoint-weighted residual,

$$\delta \bar{J} \approx -\int_0^{T_f} \mathbf{\Psi}_h^T \bar{\mathbf{R}}_h(\mathbf{U}_h^H) dt. \tag{4.15}$$

In the spatial domain, the prolongation from space H to space h is done by a prolongation to higher-order: $p \to p+1$. Note that just as was done for the steady-state problem, the fine-space adjoint $\mathbf{\Psi}_h$ has the coarse-space adjoint prolonged to the fine-space, $\mathbf{\Psi}_h^H$, removed from it. It is worth mentioning that the form of the unsteady output error estimate in Equation 4.15 is almost identical to the steady output error estimate in Equation 3.27, the only difference being the temporal integral. The integral in Equation 4.15 is a summation of integrals over time intervals, each evaluated using quadrature. Temporal reconstruction, which was already discussed in Subsection 2.3.2, of the fine-space adjoint produces the adjoint at the quadrature points.

4.2.3 Error Localization

The error estimate in Equation 4.15 is a time integral of an inner product between vectors containing spatially-local data specific to each element. The output error can be written as

$$\delta \bar{J} \approx -\int_0^{T_f} \mathbf{\Psi}_h^T \bar{\mathbf{R}}_h(\mathbf{U}_h^H) dt = \sum_{n=1}^{N_t} \sum_{e=1}^{N_e} \underbrace{\int_{t^{n-1}}^{t^n} -\mathbf{\Psi}_{h,e}^T \bar{\mathbf{R}}_{h,e}\left(\mathbf{U}_h^H\right) dt}_{\mathcal{E}_e^n}, \tag{4.16}$$

where N_t is the number of time steps, N_e is the number of elements, and the subscript e on the adjoint and residual denotes restriction to the element e. The contribution to the overall error estimate at time step n, $\mathcal{E}_e^n$, is available for each element. The temporal error contribution can be calculated using a spatially down-projected adjoint [36],

$$\mathcal{E}_e^{n,\text{time}} = -\int_{t^{n-1}}^{t^n} \left(\mathbf{I}_H^h \mathbf{\Psi}_{h,e}\right)^T \bar{\mathbf{R}}_{H,e}(\mathbf{U}_H) dt, \tag{4.17}$$

where $\mathbf{I}_H^h$ is a least-squares spatial projection operator from order p_e+1 to order p_e on element e. Note that the residual in Equation 4.17 is still computed using the fine time-integration method. The temporal error is then used to calculate the elemental spatial error at each time step via

$$\mathcal{E}_e^{n,\text{space}} \equiv \mathcal{E}_e^n - \mathcal{E}_e^{n,\text{time}}. \tag{4.18}$$

The aggregated temporal and spatial errors are then given by the sum over all of the elements and time slabs:

$$\delta\bar{J}^{\text{time}} = \sum_{n=1}^{N_t}\sum_{e=1}^{N_e}\mathcal{E}_e^{n,\text{time}}, \quad \delta\bar{J}^{\text{space}} = \sum_{n=1}^{N_t}\sum_{e=1}^{N_e}\mathcal{E}_e^{n,\text{space}} = \delta\bar{J} - \delta\bar{J}^{\text{time}}. \tag{4.19}$$

Often in adaptation conservative error estimates of the localized errors are used in order to avoid noise caused by coincidental cancellation of errors. These expressions are almost identical to those in Equation 4.19, the only difference being the inclusion of absolute values:

$$\epsilon^{\text{time}} \equiv \sum_{n=1}^{N_t}\left|\sum_{e=1}^{N_e}\mathcal{E}_e^{\text{n,time}}\right|, \quad \epsilon^{\text{space}} \equiv \sum_{e=1}^{N_e}\epsilon_e^{\text{space}}, \quad \epsilon_e^{\text{space}} = \left|\sum_{n=1}^{N_t}\mathcal{E}_e^{\text{n,space}}\right|. \tag{4.20}$$

4.2.4 Mesh Optimization

With the spatial and temporal error estimates known for a given mesh, the subsequent step is determining how to best allocate the degrees of freedom based on those error estimates. The method used in this work is based on the unsteady mesh optimization procedure introduced in [106], which relies on the aggregated output-based error estimates from Subsection 4.2.3 and a cost model based on the total number of spatial and temporal degrees of freedom. This simplistic cost model uses a measure based only on the total number of space-time degrees of freedom. It equally treats the options of increasing temporal degrees of freedom by adding more time steps or increasing the spatial degrees of freedom by increasing the resolution of the mesh, in as much as these affect the total number of degrees of freedom. For a given mesh, the total number of space-time degrees of freedom, C, is given by the product of both the temporal, C^{time}, and spatial, C^{spatial}, degrees of freedom:

$$C \equiv C^{\text{space}}C^{\text{time}}, \qquad C^{\text{space}} \equiv \sum_{e=1}^{N_e} n(p_e), \qquad C^{\text{time}} \equiv N_t n_r, \tag{4.21}$$

where $n(p_e)$ is the total number of spatial degrees of freedom per element, and n_r is the number of system solutions for non-variational time integrators. For example, $n_r = n_{\text{stages}}$ for the DIRK schemes.

The total error is found using a simple additive model of spatial and temporal errors,

$$|\delta\bar{J}| = |\delta\bar{J}^{\text{space}}| + |\delta\bar{J}^{\text{time}}|. \tag{4.22}$$

We assume an *a priori* relationship between both sets of errors and their degrees of freedom,

$$|\delta \bar{J}^{\text{space}}| \propto (C^{\text{space}})^{-(p+1)/d}, \qquad |\delta \bar{J}^{\text{time}}| \propto (C^{\text{time}})^{-r-1}, \tag{4.23}$$

where p is the average spatial order of the mesh, r is the order of accuracy of the time integration scheme, and d is the number of dimensions.

Growth factors f^{space} and f^{time} in the spatial and temporal degrees of freedom, respectively, are used for each adaptive iteration to govern the relative change in cost:

$$C^{\text{space}} = C_0^{\text{space}} f^{\text{space}}, \quad C^{\text{time}} = C_0^{\text{time}} f^{\text{time}}, \tag{4.24}$$

where the subscript 0 implies the degrees of freedom of the current mesh. The growth factors are chosen so that the marginal error to cost ratio in both space and time is relatively the same. This means that for each adaptive iteration, either the spatial or temporal degrees of freedom will increase. In addition, the number of degrees of freedom can decrease, leading to a coarsening of the mesh.

Using Equation 4.23, the corresponding changes in the errors can be written as

$$|\delta \bar{J}^{\text{space}}| = |\delta \bar{J}_0^{\text{space}}|(f^{\text{space}})^{-(p+1)/d}, \qquad |\delta \bar{J}^{\text{time}}| = |\delta \bar{J}_0^{\text{time}}|(f^{\text{time}})^{-r-1}. \tag{4.25}$$

Using equal marginal error cost ratios, $\lambda^{\text{space}} = \lambda^{\text{time}}$, we can relate f^{time} and f^{space}:

$$\lambda^{\text{space}} \equiv \frac{\partial(\delta \bar{J})}{\partial f^{\text{space}}} \left[\frac{\partial C}{\partial f^{\text{space}}}\right]^{-1}, \qquad \lambda^{\text{time}} \equiv \frac{\partial(\delta \bar{J})}{\partial f^{\text{time}}} \left[\frac{\partial C}{\partial f^{\text{time}}}\right]^{-1}. \tag{4.26}$$

To enable a solution for both f^{space} and f^{time}, a constraint is imposed on the growth of the total degrees of freedom: $f^{\text{space}} f^{\text{time}} = f^{\text{tot}}$. In [106], f^{tot} was a user-prescribed growth rate. This leads to an increase in total degrees of freedom allocation for each adaptive iteration. This make a comparison of methods at the same degrees of freedom difficult if the number of degrees of freedom drifts during adaptation.

In this work, we use a user-prescribed degrees of freedom target, rather than a user-prescribed rate. For each adaptive iteration, we calculate f^{tot} by taking the degrees of freedom target divided by the current number of degrees of freedom. This allows us to use one expression for both f^{space} and f^{time} in terms of f^{tot} to determine the cost allocation for each iteration. Whereas this leads to a relative increase in degrees of freedom for each adaptive iteration in the aforementioned reference, the total degrees of freedom always move closer to the user-prescribed target in this approach. The aforementioned expression is calculated by substituting the aforementioned error and cost models into Equation 4.26 to produce

$$f^{\text{space}} = f^{\text{time}} = \left[\frac{d(r+1)}{p+1} \frac{|\delta \bar{J}^{\text{time}}|}{|\delta \bar{J}^{\text{space}}|} (f^{\text{tot}})^{1+(p+1)/d} \right]^{\frac{1}{r+3+(p+1)/d}}. \tag{4.27}$$

4.3 Entropy Variables as Unsteady Adjoints

In Section 3.3, a detailed analysis of inviscid and viscous conservation laws demonstrated that entropy variables could be used instead of an output-based adjoint for error estimation in the context of steady-state simulations. Using entropy variables is advantageous since the computation of the entropy variables in Equation 3.44 requires no expensive fine-space adjoint problem. This same logic holds for unsteady simulations. The potential computational benefit of using entropy variables is even more significant for unsteady simulations, given that the fine-space output-based adjoint computed using Equation 4.6 must be computed at each time step.

In this section, we derive the entropy adjoint expression for both inviscid and viscous unsteady conservation laws, as was accomplished in Section 3.3 for steady-state conservation laws. The section concludes with a brief review of how entropy variables replace the output-based adjoints in the error estimation process from Section 4.2. The review of the entropy-variable approach for unsteady simulations originates from the following references [123, 124].

4.3.1 Unsteady Inviscid Conservation Laws

In the context of unsteady simulations, the scalar entropy conservation law from Equation 3.30 now has an unsteady component, as shown in the following expression,

$$U_t + \partial_i F_i = 0, \tag{4.28}$$

where $F_i(\mathbf{u})$ is the entropy flux associated with the entropy function $U(\mathbf{u})$. The quasi-linear, inviscid conservation law in Equation 3.30 can be written in an unsteady form as

$$\bar{\mathbf{r}}(\mathbf{u}) = \partial_t \mathbf{u} + \mathbf{A}_i \partial_i \mathbf{u} = \mathbf{0}. \tag{4.29}$$

For the entropy function $U(\mathbf{u})$, we can continue to define the entropy variables as $\mathbf{v} = U_{\mathbf{u}}^T$. Using the same symmetry properties and compatibility relations from Subsection 3.3.1, the unsteady, inviscid conservation law can be manipulated in the following fashion:

$$\begin{aligned}\mathbf{0} &= \partial_t \mathbf{u} + \mathbf{A}_i \partial_i \mathbf{u} \\ &= \mathbf{u}_\mathbf{v} \partial_t \mathbf{v} + \mathbf{A}_i \mathbf{u}_v \partial_i \mathbf{v} \\ &= \mathbf{u}_\mathbf{v} \partial_t \mathbf{v} + (\mathbf{A}_i \mathbf{u}_\mathbf{v})^T \partial_i \mathbf{v} \\ &= \mathbf{u}_\mathbf{v} \partial_t \mathbf{v} + \mathbf{u}_\mathbf{v}^T \mathbf{A}_i^T \partial_i \mathbf{v} \\ &= \mathbf{u}_\mathbf{v} \partial_t \mathbf{v} + \mathbf{u}_\mathbf{v} \mathbf{A}_i^T \partial_i \mathbf{v}. \end{aligned} \tag{4.30}$$

Using the property that $\mathbf{u}_\mathbf{v}$ is symmetric and positive definite, the above equation can be multiplied by $\mathbf{u}_\mathbf{v}^{-1}$ to yield

$$\partial_t \mathbf{v} + \mathbf{A}_i^T \partial_i \mathbf{v} = \mathbf{0}. \tag{4.31}$$

The output associated with the above equation can be derived by regarding the adjoint as a Green's function sensitivity to residual source perturbations, just as was done in Subsection 3.3.1. From the inviscid conservation law in Equation 4.29, changes in the state and the unsteady residual are related via

$$\delta \bar{\mathbf{r}} = \partial_t \delta \mathbf{u} + \mathbf{A}_i \partial_i \delta \mathbf{u}. \tag{4.32}$$

The output perturbation can be found using an inner product of the adjoint with the unsteady residual perturbations, which is shown in the following analysis:

$$\begin{aligned}\delta J &= \int^{T_f} \int_\Omega \mathbf{v}^T \delta \bar{\mathbf{r}} d\Omega dt \\ &= \int_\Omega \mathbf{v}^T \underbrace{(\partial_t \delta \mathbf{u} + \mathbf{A}_i \partial_i \delta \mathbf{u})}_{\delta \bar{\mathbf{r}}} d\Omega. \end{aligned} \tag{4.33}$$

Using integration by parts on Equation 4.33 and the results from Equation 4.30 produces

$$\delta J = \int^{T_f} \int_{d\Omega} \mathbf{v}^T \mathbf{A}_i \delta \mathbf{u} n_i dS dt + \left[\int_\Omega \mathbf{v}^T \delta \mathbf{u} d\Omega \right]_0^{T_f}. \tag{4.34}$$

Using the relations $\mathbf{v}^T \mathbf{A}_i \delta \mathbf{u} = (F_i)_\mathbf{u} \delta \mathbf{u} = \delta F_i$ and $\mathbf{v}^T \delta \mathbf{u} = U_\mathbf{u} \delta \mathbf{u} = \delta U$, we get an expression for the output

$$J = \int^{T_f} \int_{\partial\Omega} F_i n_i + \left[\int_\Omega U \right]_0^{T_f} + \text{constant}. \tag{4.35}$$

The integral $\int^{\partial T}$ reduces to an evaluation at $t = T_f$ up to an arbitrary constant since the initial conditions prescribe **u** at $t = 0$. The first term is the net outflow of the entropy function U from Equation 3.43 through the domain boundary integrated over the entire temporal domain. The second term is the net increase of the entropy function integrated over the spatial domain Ω from $t = 0$ to $t = T_f$.

4.3.2 Unsteady Viscous Conservation Laws

The unsteady second-order viscous conservation laws can be written in the following quasi-linear form:

$$\bar{\mathbf{r}}(\mathbf{u}) = \partial_t \mathbf{u} + \mathbf{A}_i \partial_i \mathbf{u} - \partial_i \left(\mathbf{K}_{ij} \partial_j \mathbf{u} \right) = \mathbf{0}, \tag{4.36}$$

where $\mathbf{A}_i \partial_i \mathbf{u}$ is the inviscid flux and $-\mathbf{K}_{ij} \partial_j \mathbf{u}$ is the viscous flux, just as in Equation 3.35 in Subsection 3.3.2.

All of the assumptions in Subsection 3.3.2 regarding how the entropy variables symmetrize the steady conservation laws and the viscous flux Jacobian matrix $\mathbf{K}_{ij}$ are still valid for unsteady viscous conservation laws. Substituting $\partial_i \mathbf{u} = \mathbf{u}_\mathbf{v} \partial_i \mathbf{v}$ into Equation 4.36 and taking the transpose yields the following equation for the entropy variables,

$$\partial_t \mathbf{v}^T \mathbf{u}_\mathbf{v} + \partial_i \mathbf{v}^T \mathbf{A}_i \mathbf{u}_\mathbf{v} - \partial_i \left(\partial_j \mathbf{v}^T \widetilde{\mathbf{K}}_{ji} \right) = \mathbf{0}. \tag{4.37}$$

As before, this is no longer a strict mathematical adjoint due to the sign of the third term being negative. However, the entropy variables still act as an adjoint since they represent the sensitivity of a particular output to residual perturbations.

From unsteady viscous conservation law in Equation 4.36, we can relate the changes in the state and the unsteady residual via

$$\delta \bar{\mathbf{r}} = \partial_t \delta \mathbf{u} + \mathbf{A}_i \partial_i \delta \mathbf{u} - \partial_i \left(\mathbf{K}_{ij} \partial_j \delta \mathbf{u} \right). \tag{4.38}$$

As in Subsection 4.3.1 for the inviscid conservation laws, the output perturbation can be found by starting with an inner product of the adjoint with the unsteady residual perturbations expressed in Equation 4.38 as follows:

$$\begin{aligned} \delta J &= \int^{T_f} \int_\Omega \mathbf{v}^T \delta \bar{\mathbf{r}} d\Omega dt \\ &= \int_\Omega \mathbf{v}^T \underbrace{\left(\partial_t \delta \mathbf{u} + \mathbf{A}_i \partial_i \delta \mathbf{u} - \partial_i \left(\mathbf{K}_{ij} \partial_j \delta \mathbf{u} \right) \right)}_{\delta \bar{\mathbf{r}}} d\Omega. \end{aligned} \tag{4.39}$$

Using the aforementioned properties of the entropy variables and the results from Subsections 3.3.2 and 4.3.1, the entropy variables serve as an adjoint solution for an output that takes the form of

$$J = \int^{T_f} \int_{\partial\Omega} F_i n_i dS dt + \int^{T_f} \int_{\Omega} \partial_i \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} \, d\Omega \, dt$$
$$- \int^{T_f} \int_{\partial\Omega} \mathbf{v}^T \widetilde{\mathbf{K}}_{ij} \partial_j \mathbf{v} \, n_i \, dS \, dt + \left[\int_{\Omega} U dS \right]_0^{T_f} . \quad (4.40)$$

The first term in Equation 4.40 is the net outflow of entropy (U) through the spatial domain boundary, integrated over time. The second term is the generation of entropy due to viscous dissipation within shear layers, vortices, and shocks, integrated over time. The third term is the entropy diffusion across the spatial boundary integrated over time. Finally, the last term is the net outflow of entropy from the space-time domain through the initial and final time points. The output J is zero for an exact solution to the differential equation, but it is not typically zero in a numerical simulation on a discretized space-time mesh. This is because most stable numerical schemes (though conservative) introduce spurious entropy. Adapting on J using the entropy variables as adjoints drives for the entropy-based approach for spatial and temporal refinement. Since the calculation of $\mathbf{v}$ is far cheaper than that of $\boldsymbol{\psi}$, this approach is computationally advantageous. However, it is not as reliable, as it targets all regions of the spatial domain that exhibit spurious entropy generation.

4.3.3 Error Estimation Using Entropy Variables

The unsteady output error based on entropy variables can be obtained by substituting the fine-space output-based adjoint, $\boldsymbol{\Psi}_h$, with the entropy variables computed on the fine-space, $\mathbf{v}_h$ in Equation 4.15 as follows

$$\delta \bar{J} \approx - \int_0^{T_f} \mathbf{v}_h^T \bar{\mathbf{R}}_h(\mathbf{U}_h^H) dt. \quad (4.41)$$

The process outlined in Subsection 4.2.3 can be used as before. The only difference is that the temporal error contribution uses the entropy variables instead of the output-based adjoint. This means that Equation 4.17 becomes

$$\mathcal{E}_e^{n,\text{time}} = - \int_{t^{n-1}}^{t^n} \left(\mathbf{I}_H^h \mathbf{v}_{h,e} \right)^T \bar{\mathbf{R}}_{H,e}(\mathbf{U}_H) dt. \quad (4.42)$$

4.4 Combined Approach in Unsteady Simulations

In this section, the novel combined approach is explored for unsteady simulations. Recalling from Section 3.4, the general idea behind the combined approach is to combine the error indicator obtained using an output-based adjoint with the error indicator obtained using entropy variables. However, unlike in Section 3.4 where the total error only has a spatial component, for unsteady simulations there are both spatial and temporal components of the total error. This means that each component of the error must be combined. Just as for the steady-state simulations, there are three different methodologies used to combine the indicators. The basic logic behind these three approaches is similar to the three approaches outlined in Section 3.4. However, the implementation and the location in the algorithm where the error indicators are combined is unique for unsteady problems. The details from this section are taken primarily from [123].

4.4.1 Combined Approach Using a Fine, Unsteady Output-based Adjoint

In this approach, two sets of non-conservative, aggregated spatial and temporal error indicators are calculated via Equation 4.19. The first set is calculated using the output-based adjoint $\mathbf{\Psi}$, while the second set is obtained using the entropy variables $\mathbf{v}$. The aggregated temporal error indicator based on the output-based adjoint, $\delta\bar{J}_{\mathbf{\Psi}}^{\text{time}}$, is obtained by a summation of $\mathcal{E}_{e,\mathbf{\Psi}}^{n,\text{time}}$ over all of elements and time steps via Equation 4.17. This equation is dependent on the fine-space output-based adjoint $\mathbf{\Psi}_h$. The aggregated temporal error indicator based on the entropy variables, $\delta\bar{J}_{\mathbf{v}}^{\text{time}}$, is found in an almost identical process. The only difference is that the fine-space entropy variables $\mathbf{v}_h$ are used instead of the fine-space output-based adjoint, which leads to two unique aggregated temporal error indicators.

Next, the output error based on the the output-based adjoint, $\delta\bar{J}_{\mathbf{\Psi}}$, is computed by a summation of error contributions of each element for each time step, $\mathcal{E}_{e,\mathbf{\Psi}}^{n}$, using Equation 4.16. The computation of $\mathcal{E}_{e,\mathbf{\Psi}}^{n}$ again relies on the fine-space output-based adjoint. The output error based on the entropy variables can be computed using a similar approach, the only difference being the use of the fine-space entropy variables instead of the output-based adjoint in Equation 4.16. This in turn leads to the computation of the elemental error contribution at each time step based on fine-space entropy variables, $\mathcal{E}_{e,\mathbf{v}}^{n}$.

The aggregated spatial error indicator based on the output-based adjoint, $\delta\bar{J}_{\mathbf{\Psi}}^{\text{space}}$, can be found by taking the difference between $\delta\bar{J}_{\mathbf{\Psi}}$ and $\delta\bar{J}_{\mathbf{\Psi}}^{\text{time}}$. The spatial error based on entropy variables, $\delta\bar{J}_{\mathbf{v}}^{\text{space}}$, can be computed in a similar fashion. At this point, we can compute our multiplicative combined version of the non-conservative temporal and spatial error indicators via:

$$\begin{aligned} \delta \bar{J}^{\text{space}}_{\text{comb}} &= \delta \bar{J}^{\text{space}}_{\mathbf{\Psi}} \cdot \delta \bar{J}^{\text{space}}_{\mathbf{v}}, \\ \delta \bar{J}^{\text{time}}_{\text{comb}} &= \delta \bar{J}^{\text{time}}_{\mathbf{\Psi}} \cdot \delta \bar{J}^{\text{time}}_{\mathbf{v}}. \end{aligned} \tag{4.43}$$

If we are interested in a conservative error estimate, two separate sets of conservative, aggregated spatial and temporal error estimates are computed via

$$\begin{aligned} \epsilon^{\text{time}}_{\mathbf{\Psi}} &\equiv \sum_{n=1}^{N_t} \left| \sum_{e=1}^{N_e} \mathcal{E}^{\text{n,time}}_{e,\mathbf{\Psi}} \right|, \quad \epsilon^{\text{space}}_{\mathbf{\Psi}} \equiv \sum_{e=1}^{N_e} \epsilon^{\text{space}}_{e,\mathbf{\Psi}}, \quad \epsilon^{\text{space}}_{e,\mathbf{\Psi}} = \left| \sum_{n=1}^{N_t} \mathcal{E}^{\text{n,space}}_{e,\mathbf{\Psi}} \right| \\ \epsilon^{\text{time}}_{\mathbf{v}} &\equiv \sum_{n=1}^{N_t} \left| \sum_{e=1}^{N_e} \mathcal{E}^{\text{n,time}}_{e,\mathbf{v}} \right|, \quad \epsilon^{\text{space}}_{\mathbf{v}} \equiv \sum_{e=1}^{N_e} \epsilon^{\text{space}}_{e,\mathbf{v}}, \quad \epsilon^{\text{space}}_{e,\mathbf{v}} = \left| \sum_{n=1}^{N_t} \mathcal{E}^{\text{n,space}}_{e,\mathbf{v}} \right|. \end{aligned} \tag{4.44}$$

Combined versions are again obtained by simply multiplying the two sets together:

$$\begin{aligned} \epsilon^{\text{time}}_{\text{comb}} &= \epsilon^{\text{time}}_{\mathbf{\Psi}} \cdot \epsilon^{\text{time}}_{\mathbf{v}}, \\ \epsilon^{\text{space}}_{\text{comb}} &= \epsilon^{\text{space}}_{\mathbf{\Psi}} \cdot \epsilon^{\text{space}}_{\mathbf{v}}. \end{aligned} \tag{4.45}$$

In addition to combining the aggregated spatial and temporal (conservative and non-conservative) error, the elemental spatial error indicator vectors for both the output-based and entropy-based approaches are combined via elemental multiplication. This can be expressed mathematically as

$$\epsilon^{\text{space}}_{e,\text{comb}} = \epsilon^{\text{space}}_{e,\mathbf{\Psi}} \cdot \epsilon^{\text{space}}_{e,\mathbf{v}}. \tag{4.46}$$

This combination is completed after the aggregated errors are combined. The combined elemental spatial error indicator, $\epsilon^{\text{space}}_{e,\text{comb}}$ is not used in obtaining the aggregated total error contributions. It is only used to determine the error sampling of each element that the mesh adaptation strategy MOESS relies upon to determine the relative sizing and anisotropy. The complete details of MOESS and this local sampling approach will be discussed in Section 5.4 of Chapter 5.

4.4.2 Combined Approach Using a Coarse, Unsteady Output-Based Adjoint

In Subsection 4.2.1, Equation 4.16 requires the computation of the fine-space output-based adjoint, of spatial order $p + 1$, at every time step. Given that the solution of the fine-space output-based adjoint, $\mathbf{\Psi}_h$, is quite expensive, unsteady simulations only magnify this problem. To minimize the cost of this step, an additional combined approach is implemented in which no fine-space output-based adjoint is computed. Instead, an output-based adjoint at the current approximation space,

$\mathbf{\Psi}_H$, is computed at each time step using a similar process to the one outlined in Subsection 3.4.2. Computing an output-based adjoint at a lower spatial approximation space is much less computationally expensive, but it leads to much more inaccurate error estimations as shown in Section 3.2.

To mitigate the inaccuracies brought on by not using the fine-space output-based adjoint, we combine the spatial and temporal error indicators with those obtained using fine-space entropy variables, $\mathbf{v}_h$, since fine-space entropy variables are far less computationally expensive to obtain. By combining the less accurate error estimates obtained using the coarser output-based adjoint with those obtained using fine-space entropy variables, we should get results that compare favorably to the standard combined approach, but are far less expensive to compute.

Using a coarser output-based adjoint does not change the error estimation approach outlined in Subsection 4.2.2 significantly. The main change is that Equation 4.15 is now computed at the current approximation space and uses the output-based adjoint at the current-space, $\mathbf{\Psi}_H$. Just as in Subsection 3.4.2, the current-space adjoint is projected down one order and prolonged back up to the original order using Equation 3.48 and subtracted from $\mathbf{\Psi}_H$ to eliminate p-dependence of the weak form. Defining $\mathbf{U}_H^{\tilde{H}}$ as the projection of the primal from the space H at order p, to space $\tilde{H}$ at order $p-1$, and then back up to space H at order p, the output error based on the coarser output-based adjoint can now be written as

$$\delta \bar{J} \approx -\int_0^{T_f} \mathbf{\Psi}_H^T \bar{\mathbf{R}}_H(\mathbf{U}_H^{\tilde{H}}) dt = \sum_{n=1}^{N_t} \sum_{e=1}^{N_e} \underbrace{\int_{t^{n-1}}^{t^n} -\mathbf{\Psi}_{H,e}^T \bar{\mathbf{R}}_{H,e}\left(\mathbf{U}_H^{\tilde{H}}\right) dt}_{\mathcal{E}_{e,\mathbf{\Psi}}^n}. \tag{4.47}$$

As was the case for steady-state simulations, this step is necessary since the unsteady residual $\bar{\mathbf{R}}_H$ of the current-space solution $\mathbf{U}_H$ is equal to zero. Since the output error using entropy variables is still evaluated at the fine-space, it is necessary to make another copy of the primal state that is prolonged into the finer-space h and evaluate a subsequent residual. We can write this mathematically as

$$\delta \bar{J}_{\mathbf{v}} \approx -\int_0^{T_f} \mathbf{v}_h^T \bar{\mathbf{R}}_h(\mathbf{U}_h^H) dt = \sum_{n=1}^{N_t} \sum_{e=1}^{N_e} \underbrace{\int_{t^{n-1}}^{t^n} -\mathbf{v}_{h,e}^T \bar{\mathbf{R}}_{h,e}\left(\mathbf{U}_h^H\right) dt}_{\mathcal{E}_{e,\mathbf{v}}^n}. \tag{4.48}$$

Unlike the spatial error contribution which changes due to the use of the coarser output-based adjoint, the temporal error contribution, Equation 4.17, is obtained the same way. The only difference is that no spatial projection of the fine-space output-based adjoint is necessary since the adjoint is already at the current-space order p. The combination of the errors outlined in Subsection 4.4.1 does not change, since they are not dependent on the order of the output-based adjoint or entropy variables when the errors were accumulated.

4.4.3 Combined Approach Using a Coarse, Unsteady Adjoint and Mask

As was done in Subsection 3.4.3, the last combined approach considered for unsteady simulations incorporates a mask on the elemental spatial error indicator vector based on the entropy variables, $\epsilon_{e,\mathbf{v}}^{\text{space}}$. This mask is applied to the indicator before obtaining the aggregated errors obtained in Equation. 4.44. The predominant reason is that the aggregated spatial errors can be properly computed with the effect of the mask, which leads to a connection between the allocation of spatial and temporal degrees of freedom to the effect of the mask.

The basic principles and implementation logic for the mask are very similar to the mask described in Subsection 3.4.3. The purpose of the mask is to limit the weight of the entropy indicator during mesh adaptation so that the output-based indicator carries more weight in spatial optimization. This is done by creating an elemental mask based on the spatial adaptation indicator obtained using the coarser output-based adjoint. This mask, μ_e, is a vector that is the same size as the output-based, elemental, spatial error indicator $\epsilon_{e,\mathbf{\Psi}}^{\text{space}}$. The mask is created by examining the relative magnitude of the indicator for each element. Note that this requires taking an absolute value of the indicator. Using a user-specified percentage, a subset of the output-based indicators is created.

The mask is then multiplied by the entropy-based spatial error indicator in an element-wise fashion to obtain the final indicator that ultimately governs the mesh adaptation,

$$\epsilon_{e,\text{comb}}^{\text{space}} = \mu_e \cdot \epsilon_{e,\mathbf{v}}^{\text{space}}. \tag{4.49}$$

By applying a mask on the spatial error indicator obtained using entropy variables, the mesh adaptation will focus less on refining the computational domain in locations that have little impact on the desired engineering output. It is worth emphasizing that no masking is done for the temporal error indicator. The temporal error contribution is still obtained using the combined approach outlined in Subsection 4.4.2.

CHAPTER 5

Mesh Adaptation Mechanics

The two previous chapters described different ways of obtaining a local, elemental error estimate. The ultimate purpose of that error estimate is to modify the computational mesh to decrease the total error. In this work, this is done through mesh adaptation. In high-order finite element methods, there are multiple types of adaptation. In p-adaptation, the solution approximation order is modified, whereas in h-adaptation the computational mesh is changed. These approaches can be combined to form hp-adaptation. In this work, we focus only on h-adaptation at a constant p.

There are three different strategies used for mesh adaptation in this work. The first mesh adaptation approach is a fixed-fraction hanging-node strategy, where a fixed fraction of elements with the largest error indicators is marked for uniform refinement using a series of hanging nodes. The main advantage of this approach is that it is a very simple approach to implement. However, this assumes that no mesh coarsening is involved in the implementation. Without mesh coarsening, this method has limitations.

Another approach to adapting the mesh is global re-meshing. Unlike in hanging node adaptation where the refinement occurs locally, in this approach a new mesh is generated for the entire computational domain. Unstructured meshes work well with this approach since resolution can be placed only where necessary. This allows for more flexibility compared to structured meshes. In this work, we consider two different global re-meshing strategies where we allow mesh refinement and coarsening, as well as redistribution of the mesh degrees of freedom to allow for better productivity. The two approaches are distinguishable based on how they determine anisotropy.

Often flow solutions in CFD require highly anisotropic elements to capture flow features such as shocks and boundary layers. Unfortunately, the error indicators obtained in the previous chapters do not provide any information that supports anisotropic mesh adaptation. The first global re-meshing strategy accounts for anisotropy by sizing the elements using *a priori* rate estimates and using Hessians of scalars computed from the state to determine each element's anisotropy. The other approach is a variation of mesh optimization via error sampling and synthesis (MOESS) [91, 92] where the mesh is optimized to minimize the output error using a prescribed cost. The distinguishing feature of the present implementation of MOESS is an elemental error sampling approach for determining the convergence rate tensor of the elemental error. This chapter reviews the algorithms and details for all three aforementioned mesh adaptation strategies.

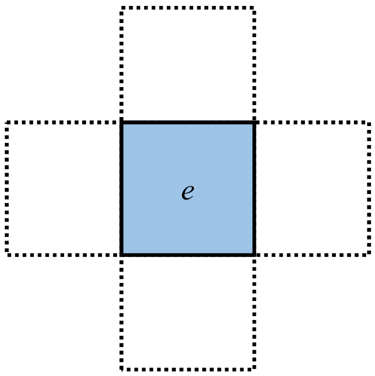
5.1 Fixed-Fraction Hanging-Node

The first mesh adaptation strategy used in this work is a fixed-fraction hanging-node adaptation strategy [125]. The driver of this approach is the elemental error indicator, $\mathcal{E}_e$. The term fixed-fraction means a certain fraction, f^{adapt}, of the elements with the largest adaptive indicators is marked for refinement. This strategy is advantageous because of the simplicity and predictability of its algorithm. The primary loop of the algorithm is over the elements targeted by the adaptive indicator. As the algorithm loops over each element, it must decide whether or not to refine the element uniformly by creating a series of hanging nodes. If two neighboring elements vary by more than one level of refinement, additional elements are flagged for refinement. Note that due to the complexity, there is no element coarsening performed while using hanging-node adaptation in this work.

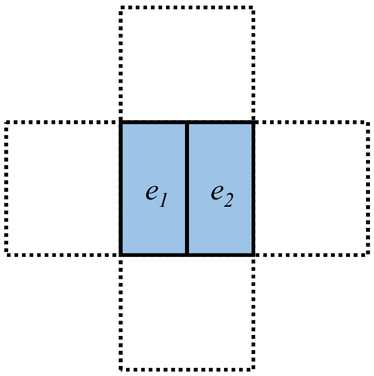
Figure 5.1 shows the discrete options for refining quadrilaterals using the fixed-fraction hanging-node adaptation strategy in two dimensions. The options are x-refinement, y-refinement, and xy-refinement. The reference-space coordinate system of x and y is arbitrary and is curved in physical space. In three dimensions, hexahedrons are refined in seven ways: isotropic refinement, three double-plane cuts, and three single-plane cuts.

The actual refinement of each element is performed in an element's reference space by using the original element's reference-to-global mapping when calculating the refined element's geometry node coordinates. When an element is refined, each subelement retains the solution approximation order, p, and quadrature rules from the original element. This ensures no loss in element quality even when the elements are curved. However, this means that the initial mesh must capture the geometry curvature well. This means that for complex curved geometry, there needs to be a sufficient amount of curved elements present in the initial mesh. As the mesh is refined, no additional geometry information will be used. Consequently, the refinement of elements on the geometry boundary does not change the geometry. For simplicity, curved elements are used throughout the domain for highly anisotropic meshes. Further details on geometry mapping from reference to global space and curved meshing can be found in Appendix B.

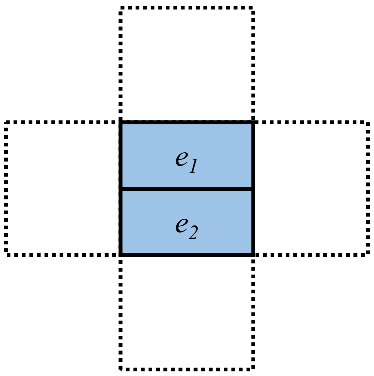
Elements created using hanging-node refinement can be refined again in future adaptive iterations. When this happens, the neighboring elements will be cut to keep one level of refinement difference between any two adjacent cells, as shown in Figure 5.2. In this image, the colored element on the left is marked for refinement. The dashed lines on the right show how the elements will be cut so that there is only one level of refinement difference between any two elements.



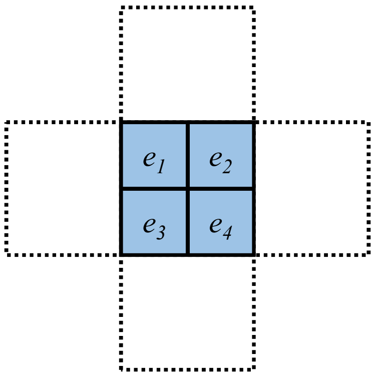
(a) Initial element (no refinement)



(b) x-refinement



(c) y-refinement



(d) xy-refinement

Figure 5.1: Refinement options for a quadrilateral element (dashed lines indicate neighbor elements).

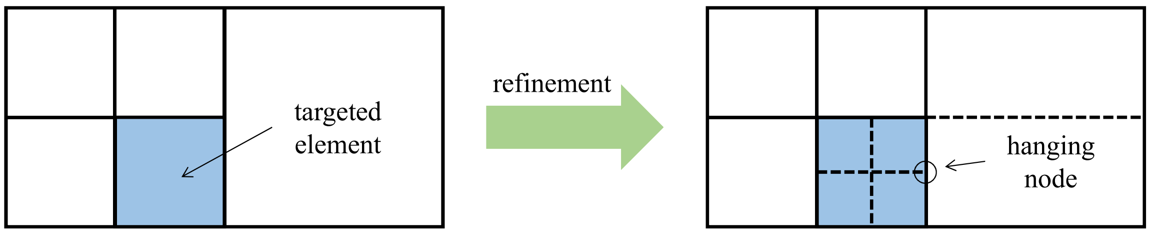


Figure 5.2: Hanging-node adaptation for a quadrilateral mesh with one level of refinement difference between any two elements.

Some applications of the hanging node mesh adaptation strategy used in this work use an additional parameter called the adaptation fixed error fraction, f^{error}. This quantity indicates the fraction of total error (sum of indicators) that will be targeted for refinement during adaptation. Specifically, the fraction of elements targeted for adaptation, $\tilde{f}$, is not constant in this approach, but instead chosen as the value that makes the sum of the error indicators in the targeted elements (starting with those possessing the highest errors) a fixed fraction of the total error indicator sum. However, the actual fraction of elements adapted will still be limited by f^{adapt}, so that

$$\text{fraction of elements adapted } = f = \min(\tilde{f}, f^{\text{adapt}}). \tag{5.1}$$

5.2 Riemannian Metric Field

Before detailing the more complex mesh adaptation strategies used in this work that take into account anisotropy, an introduction of the Riemannian metric field is necessary. In global re-meshing adaptation approaches, the original mesh, or background mesh, is used to store certain mesh characteristics during the global re-meshing step. Specifically, elemental sizing and anisotropic information are mesh characteristics that must be available during the creation of the new mesh so that the new mesh matches these desired characteristics. One way of storing this information is with a Riemannian metric field [24, 62, 91, 100, 126–129].

A Riemannian metric field is a field of symmetric positive definite (SPD) tensors that is used to store anisotropic information of the computational mesh, specifically mesh sizes and stretching directions. Mathematically, we denote the metric field by $\boldsymbol{\mathcal{M}}(\vec{x}) \in \mathbb{R}^{d\times d}$. The metric tensor, $\boldsymbol{\mathcal{M}}(\vec{x})$, provides a convenient way of computing the distance between any two points under the metric. For two arbitrary points a and b, the length of the vector $\overrightarrow{ab}$ under the metric $l_{\boldsymbol{\mathcal{M}}}(\overrightarrow{ab})$ is defined as

$$l_{\boldsymbol{\mathcal{M}}}(\overrightarrow{ab}) = \int_a^b dl = \int_a^b \sqrt{\overrightarrow{ab}^T \boldsymbol{\mathcal{M}} \overrightarrow{ab}}, \qquad \vec{ab} = \vec{x}_a - \vec{x}_b, \qquad \forall \vec{x}_a, \vec{x}_b \in \mathbb{R}^d, \tag{5.2}$$

where $\vec{x}_a$ and $\vec{x}_b$ are the coordinates of points a and b. In two dimensions, any set of points at a unit metric distance away from a point forms an ellipse as shown in Figure 5.3. For three dimensions, the points would form an ellipsoid. The eigenvectors of $\boldsymbol{\mathcal{M}}$ are the principal stretching directions and the eigenvalues are the length (stretching) of each axis. The inverse square root of the eigenvalue is equal to the magnitude of the total stretching. In Figure 5.3, $\vec{e}_1$ and $\vec{e}_2$ represent the directions of the eigenvectors of $\boldsymbol{\mathcal{M}}$, while the distances h_1 and h_2 are defined by the corresponding eigenvalues. The angle θ represents the rotation of the eigenvectors relative to the global coordinate system. For a unit circle centered at the point O, the metric can be expressed mathematically as

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1/h_1^2 & 0 \\ 0 & 1/h_2^2 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{5.3}$$

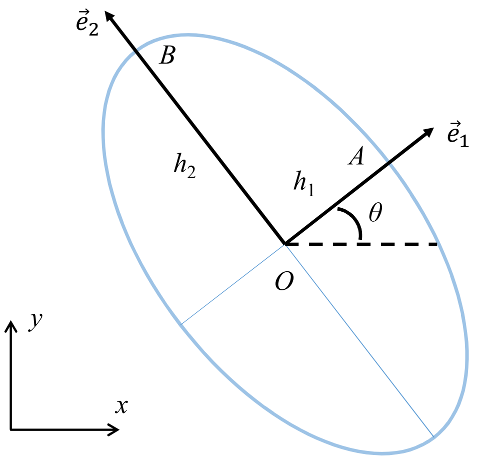


Figure 5.3: Riemannian metric field, $\boldsymbol{\mathcal{M}}$, in two dimensions.

The metric field can be modified by applying affine-invariant changes [130] using a symmetric step matrix $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d\times d}$ via

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{M}}_0^{\frac{1}{2}} \exp(\boldsymbol{\mathcal{S}}) \boldsymbol{\mathcal{M}}_0^{\frac{1}{2}}, \tag{5.4}$$

where $\boldsymbol{\mathcal{M}}_0$ is the original metric and $\boldsymbol{\mathcal{M}}$ is the modified metric. A logarithmic map from the current map to the modified one can now be defined as

$$\boldsymbol{\mathcal{S}} = \log\left(\boldsymbol{\mathcal{M}}_0^{-1/2} \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{M}}_0^{-1/2}\right). \tag{5.5}$$

Note that $\boldsymbol{\mathcal{S}} = 0$ leaves the metric unchanged, while diagonal values in $\mathcal{S}$ of $\pm 2\log 2$ either halve or double the stretching sizes of the metric. The ability to change the metric using $\boldsymbol{\mathcal{S}}$ is how metric-based optimization algorithms, such as MOESS, determine the optimal local changes for a particular element.

A mesh where each edge of the elements has approximately the same metric length, as defined by Equation 5.2, is a metric-conforming mesh. However, there are many possible metric-conforming meshes [91]. In metric-based meshing, the Riemannian metric field is used to store the mesh characteristics (such as the size and shape of each element) and to develop a continuous model of the discrete mesh [130]. The mesh-implied metric can be obtained using the reference-to-global mapping Jacobian on each element. Specifically, we solve a linear system for $d(d+1)/2$ entries for each element, which determines the Riemannian metric, $\boldsymbol{\mathcal{M}}$. Using both the metric-conforming mesh and the mesh-implied metric allows for the conversion between the Riemannian metric field and an anisotropic mesh.

Several mesh generators exist that have metric-conforming mesh generation capabilities. For adaptive meshing in two dimensions for this work, the Bi-dimensional Anisotropic Mesh Generator (BAMG) [131] is used to perform global re-meshing for each adaptive iteration. During each adaptive iteration, BAMG uses the desired metric field and background mesh to create a mesh that conforms to the desired metric. In Figure 5.4, a desired metric field and isotropic element is used to create a new anisotropic element that conforms to the desired metric field.

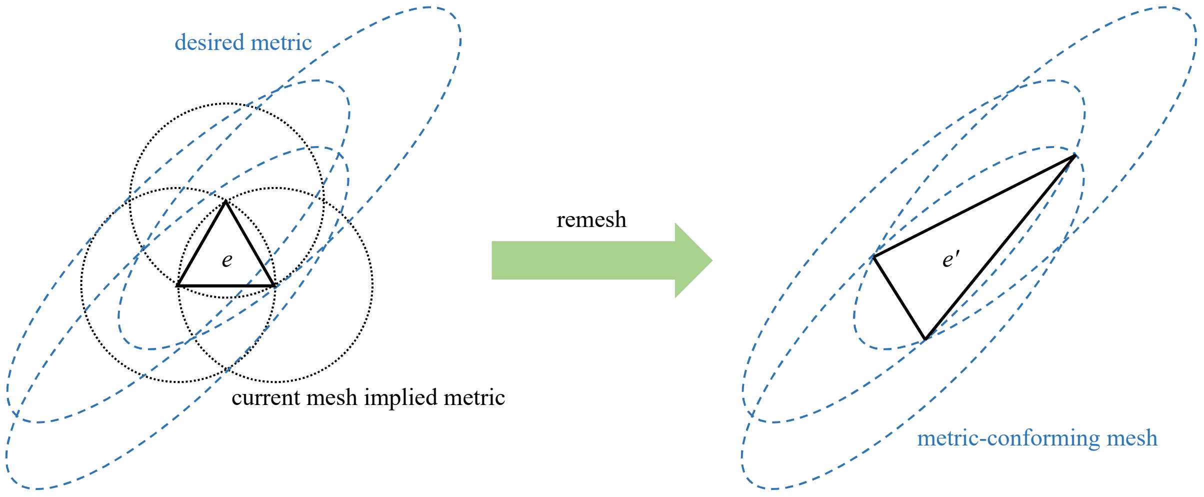


Figure 5.4: Illustration of metric-based global re-meshing in adaptation (figure based off image in [129]). The image on the left shows the background isotropic mesh element with the corresponding mesh-implied metric field. The image on the right shows the metric-confirming anisotropic mesh element.

5.3 *A Priori* Mesh Adaptation

In the *a priori* mesh adaptation approach adopted for this work, the anisotropy information for each element is found by estimating the direct interpolation error of the solution [62, 64, 127, 128, 132]. The mesh is then adapted to control a norm of the interpolation error so that areas of the computational domain that exhibit discontinuities or singularities (such as shocks) will not be excessively refined. However, this method is not aware of output sensitivities to the solution errors. To remedy this, the elemental error indicator from Chapter 3 can be included in the methodology [64, 65, 72]. This will lead to better control of the elemental sizing. The approach outlined in this section is primarily based on [101, 129].

5.3.1 Anisotropic Detection Using Hessians

For linear approximations, we define the interpolation error of a scalar solution u over an edge E in the mesh as

$$\delta_{u,E} = \left| \vec{s}^T \mathbf{H} \vec{s} \right| h^2, \tag{5.6}$$

where $\vec{s}$ is the edge's unit tangent vector, h is the length, and $\mathbf{H} \in \mathbb{R}^{d \times d}$ is the Hessian matrix of a chosen scalar,

$$H_{i,j} = \frac{\partial^2 u}{\partial x_i \partial x_j}, \ i, j \in [1, ..., d] \, . \tag{5.7}$$

Quadratic reconstruction of the linear solution is used to estimate second-order derivatives. The scalar solution chosen for this work, which is suitable for many types of flows [24, 64, 101], is the Mach number.

We can interpret Equation 5.6 geometrically by defining the interpolation error over an edge as a quantity that depends directly on the squared edge length under the measure of an SPD matrix defined by taking the absolute value of the Hessian. We define this mathematically as $|\mathbf{H}|$. Assuming the mesh conforms to the Riemannian metric field $\boldsymbol{\mathcal{M}}$, assumed constant along the edge E, the edge is length is equal to unity. This can be expressed mathematically, using Equation 5.2, as

$$l_{\boldsymbol{\mathcal{M}}} = \sqrt{\vec{s}^T \boldsymbol{\mathcal{M}} \vec{s} h^2} = 1. \tag{5.8}$$

Using Equation 5.6, the metric field can be related to the interpolation error,

$$\delta_{u,E} \propto \frac{\left| \vec{s}^T \mathbf{H} \vec{s} \right|}{\vec{s}^T \boldsymbol{\mathcal{M}} \vec{s}}. \tag{5.9}$$

Equal distribution of the errors gives

$$\frac{\vec{s}^T \boldsymbol{\mathcal{M}} \vec{s}}{\vec{s}^T \mathbf{H} \vec{s}} = k, \tag{5.10}$$

where k is a constant. A suitable choice for the metric $\boldsymbol{\mathcal{M}}$ is one so that Equation 5.10 holds in any direction,

$$\boldsymbol{\mathcal{M}}_{\mathbf{H}} = k\left|\mathbf{H}\right| = k\mathbf{Q}\left|\boldsymbol{\Lambda}_{\mathbf{H}}\right|\mathbf{Q}^T = \mathbf{Q}\boldsymbol{\Lambda}_{\boldsymbol{\mathcal{M}}}\mathbf{Q}^T, \tag{5.11}$$

where $\mathbf{Q}$ are the orthonormal eigenvectors of $\mathbf{H}$ and $\boldsymbol{\Lambda}_{\mathbf{H}}$ and $\boldsymbol{\Lambda}_{\boldsymbol{\mathcal{M}}}$ are the diagonal matrices of the eigenvalues for both $\mathbf{H}$ and $\boldsymbol{\mathcal{M}}$, respectively. Note that the eigenvectors $\mathbf{Q}$ are the same for both $\mathbf{H}$ and $\boldsymbol{\mathcal{M}}$. From Equation 5.11 we can infer the metric field shape and the metric-conforming mesh,

$$\frac{\lambda_{\boldsymbol{\mathcal{M}},j}}{\lambda_{\boldsymbol{\mathcal{M}},i}} = \left|\frac{\lambda_{\mathbf{H},j}}{\lambda_{\mathbf{H},i}}\right| \Rightarrow \frac{h_i}{h_j} = \sqrt{\frac{\lambda_{\boldsymbol{\mathcal{M}},j}}{\lambda_{\boldsymbol{\mathcal{M}},i}}} = \sqrt{\left|\frac{\lambda_{\mathbf{H},j}}{\lambda_{\mathbf{H},i}}\right|}, \tag{5.12}$$

where $\lambda_{\boldsymbol{\mathcal{M}}}$ and $\lambda_{\mathbf{H}}$ are the eigenvalues of $\boldsymbol{\mathcal{M}}$ and $\mathbf{H}$, respectively. The terms h_i and h_j represent the desired sizes of the elements in the $\vec{e}_i$ and $\vec{e}_j$ principal directions.

For high-order approximations, the interpolation error is characterized by $p+1$ derivatives and the first d largest directional derivatives are used to determine the principal directions of $\mathbf{Q}$ and the corresponding stretching of $\boldsymbol{\Lambda}_{\mathbf{H}}$ [65]. However, in this work, we only consider using the Hessian matrix of second-order derivatives, regardless of the solution approximation order. The principle of equal distribution of the error dictates the optimal mesh is found by equally distributing the squared edge length under $|\mathbf{H}|$. For two principal directions $\vec{e}_i$ and $\vec{e}_j$, equal error distribution yields the mesh stretching as [129]

$$\frac{h_i}{h_j} = \left|\frac{u_{\vec{e}_j}}{u_{\vec{e}_i}}\right|^{1/(p+1)} = \left|\frac{\lambda_{\mathbf{H},j}}{\lambda_{\mathbf{H},i}}\right|^{1/(p+1)} = \sqrt{\frac{\lambda_{\boldsymbol{\mathcal{M}},j}}{\lambda_{\boldsymbol{\mathcal{M}},i}}}, \tag{5.13}$$

where $u_{\vec{e}_i}$ is the $p+1^{\text{st}}$ derivate along the direction $\vec{e}_i$ whose magnitude is found by the i^{th} eigenvalue λ_i. The expression in Equation 5.13 can then be used to control how elements stretch relative to their initial size, while the error indicator controls the absolute mesh sizes.

5.3.2 Element Sizing Based on *A Priori* Rates

In the previous subsection, we outlined the process for obtaining a scalar-based Hessian, $\mathbf{H}$, that can be used to control the stretching information for each element in the mesh. However, the Hessian cannot be used to determine how many elements, N^f, will be in the new adaptive mesh. Instead, we will rely on *a priori* estimates of the error rates using the elemental error indicators obtained in the previous chapters. To do so, we must estimate the number of adapted elements, n_e, in element e of the original mesh. By defining the current element size as h_i^c and the desired element size as h_i^f, the number of adapted elements can be defined as

$$n_e = \prod_{i=1}^{d} \left(h_i^c / h_i^f \right), \tag{5.14}$$

where i denotes the principal directions. We then relate the increase in the number of elements to an error reduction factor through an *a priori* estimate

$$\underbrace{n_e \frac{\mathcal{E}^f}{N^f}}_{\text{allowable error}} = \underbrace{\mathcal{E}_e^c \left(\frac{h_{\text{ref}}^f}{h_{\text{ref}}^c} \right)^{\bar{p}_e+1}}_{\text{a priori estimate}} = \mathcal{E}_e^c \left[\prod_{i=1}^{d} \left(\frac{h_i^f}{h_i^c} \right) \right]^{(\bar{p}_e+1)/d}, \tag{5.15}$$

where $\mathcal{E}_e^c$ is the elemental error indicator, $\bar{p}_e = \min(p_e, \gamma_e)$, and γ_e is the lowest order of any singularity in the element. The error reduction factor, $\mathcal{E}^f/N^f$, is the ratio of the output error tolerance to the number of elements in the adapted mesh. In this work, the output error tolerance is not known. Instead, it is controlled by a target number of elements, N_f, and computed via [129]

$$\mathcal{E}^f = N^f \left(\frac{1}{N^f} \sum_{e=1}^{N_e} (\mathcal{E}_e^c)^{r'} \right)^{1/r'}, \tag{5.16}$$

where $r' = d/(p+1+d)$. With $\mathcal{E}^f$ and N^f known, the scaling factor in element size, h_i^f/h_i^c, must be computed because anisotropic information comes from the Hessian, which cannot be used to obtain element size. The mesh sizing is found through the smallest principle stretching and the corresponding aspect ratios, along other principal directions [129].

Substituting Equation 5.14 into Equation 5.15 produces a helpful relationship between n_e and N^f that form the starting point for determining the element sizing. We can write this relationship as

$$n_e \frac{\mathcal{E}^f}{N^f} = \mathcal{E}_e^c \left[\prod_{i=1}^{d} \left(\frac{h_i^f}{h_i^c} \right) \right]^{(\bar{p}_e+1)/d} = \mathcal{E}_e^c n_e^{-(\bar{p}_e+1)/d} \Rightarrow n_e^{1+(\bar{p}_e+1)/d} = \frac{\mathcal{E}_e^c}{\mathcal{E}^f/N^f}. \tag{5.17}$$

Defining the aspect ratio along the i^{th} principal direction with respect to the smallest stretching direction, h_1, as $\text{AR}_i = h_i/h_1$, we can rewrite Equation 5.17 as

$$n_e^{1+(p+1)/d} = \left[\prod_{i=1}^{d} \left(\frac{h_i^c}{h_i^f} \right) \right]^{1+(p+1)/d} = \left[\left(\frac{h_1^c}{h_1^f} \right)^{d} \prod_{i=1}^{d} \left(\frac{\text{AR}_i^c}{\text{AR}_i^f} \right) \right]^{1+(p+1)/d} = \frac{\mathcal{E}_e^c}{\mathcal{E}^f/N^f}. \tag{5.18}$$

By rearranging Equation 5.18, the desired element size can be found using

$$\frac{h_1^f}{h_1^c} = \left[\frac{\mathcal{E}^f}{N^f}\frac{1}{\mathcal{E}_e^c}\right]^{1+(p+1)/d}\left[\prod_{i=1}^{d}\left(\frac{\mathrm{AR}_i^c}{\mathrm{AR}_i^f}\right)\right]^{1/d}, \qquad \frac{h_i^f}{h_1^f} = \mathrm{AR}_i^f. \tag{5.19}$$

A new metric can now be constructed by combining the stretching information and principle directions for the desired metric with the scaling in element size found using Equation 5.19.

5.4 Mesh Adaptation Using MOESS

The *a priori* mesh adaptation approach outlined Section 5.3 has demonstrated an ability to detect solution anisotropy in many applications [24, 64]. The drawback with this approach is that it relies on a scalar-based Hessian (the Mach number in this work) to drive adaptation. Depending on the desired output, the chosen scalar may not be an adequate choice. This makes the approach somewhat arbitrary in its implementation. In addition, when the Hessian is near-zero or the identity, the consequent stretching may be insufficient. For problems that require highly anisotropic elements, the *a priori* mesh adaptation approach may not be as robust compared to other h-adaptation methods.

A more rigorous approach is mesh optimization via error sampling and synthesis (MOESS) [91]. The basic goal of MOESS is to optimize the computational mesh to minimize the output error at a prescribed computational cost. For MOESS to properly optimize the existing computational mesh, we need a model to relate how the error changes as the metric changes for a particular element Ω_e. During the optimization, the original metric, $\boldsymbol{\mathcal{M}}_0\left(\vec{x}\right)$, is modified iteratively until the optimal metric is found. This is done by searching for optimal changes of the step matrix, from Equation 5.5, that minimize the total error given a prescribed cost:

$$\begin{aligned} \min_{\boldsymbol{\mathcal{S}}}\ \mathcal{E} &= \sum_{e=1}^{N_e}\mathcal{E}_e\left(\boldsymbol{\mathcal{S}}\right) \\ \text{s.t.}\ \ \mathcal{C} &= \sum_{e=1}^{N_e}\mathcal{C}_e\left(\boldsymbol{\mathcal{S}}\right) = \text{constant}, \end{aligned} \tag{5.20}$$

where $\mathcal{C}$ and $\mathcal{E}$ are the total cost and total error associated with the original mesh. The elemental error convergence model $\mathcal{E}_e(\boldsymbol{\mathcal{S}})$ and the elemental cost model $\mathcal{C}_e(\boldsymbol{\mathcal{S}})$ are the main inputs to an algorithm that will optimize the mesh.

A distinguishing feature of this version of MOESS is the local, elemental error sampling approach for determining the convergence rate tensor of the elemental error [92, 129]. This feature is advantageous since it does not require refinement of the element or any additional residual evaluations that add computational cost. A complete review of the MOESS algorithm in the context of a Discontinuous Galerkin discretization for steady-state simulations is the focus of this section, followed by a review of changes in MOESS for unsteady problems. The section closes with a brief review of the changes in MOESS that were made to comply with the SUPG code developed for this work.

5.4.1 Error Convergence Model

One of the benefits to MOESS compared to the *a priori* mesh adaptation strategy described in Section 5.3 is that MOESS does not assume a constant *a priori* and isotropic rate of error convergence. This is because anisotropy comes from interpolation error, while the element sizes come from the error indicators obtained from Chapter 3. The isotropic error convergence from Equation 5.15 states that

$$\frac{\mathcal{E}_e}{\mathcal{E}_{e,0}} = \left(\frac{h}{h_0}\right)^r = \exp\left[r \log\left(\frac{h}{h_0}\right)\right], \tag{5.21}$$

where h/h_0 is a measure of element size change and r is the *a priori* isotropic error convergence rate. MOESS uses a more generalized error convergence model, where the error converges differently depending on which direction changes exist. In this new model, we replace $\log(h/h_0)$ with an elemental step matrix $\boldsymbol{\mathcal{S}}_e$, and r with a symmetric rate tensor $\boldsymbol{\mathcal{R}}_e$ so that

$$\mathcal{E}_e = \mathcal{E}_{e,0} \exp\left[\text{tr}\left(\boldsymbol{\mathcal{R}}_e \boldsymbol{\mathcal{S}}_e\right)\right] \Rightarrow \frac{\partial \mathcal{E}_e}{\partial \boldsymbol{\mathcal{S}}_e} = \mathcal{E}_e \boldsymbol{\mathcal{R}}_e. \tag{5.22}$$

Note the linearization of the error with respect to $\boldsymbol{\mathcal{S}}_e$ is included for use in Subsection 5.4.4. The total error over the mesh is the sum of each elemental error:

$$\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e. \tag{5.23}$$

During the optimization, the error $\mathcal{E}$ is kept small while we change the step matrices at the mesh vertices, $\boldsymbol{\mathcal{S}}_v$. The rate tensor $\boldsymbol{\mathcal{R}}_e$ is determined using a sampling process for each element. This means the anisotropy is determined *a priori* through sampling.

5.4.2 Element-Local Error Sampling

The anisotropic error model in Equation 5.22 requires computing the error convergence rate tensor $\mathcal{R}_e$ for each element e. An estimation of the rate tensor is performed a posteriori by sampling a series of different refinement options for each element. For each sample, the metric is changed from the original metric $\mathcal{M}$ to a modified metric $\mathcal{M}_0$. Figure 5.5 shows the four refinement options for a triangular element, as well as how the metric changes for each option. The corresponding step matrix can then be computed using Equation 5.5. Using this step matrix, we can approximate the rate tensor by calculating how much the error changes for each step matrix. This is identical to finding how much the error changes for each refinement option, which will eventually allow us to determine which is the optimal choice.

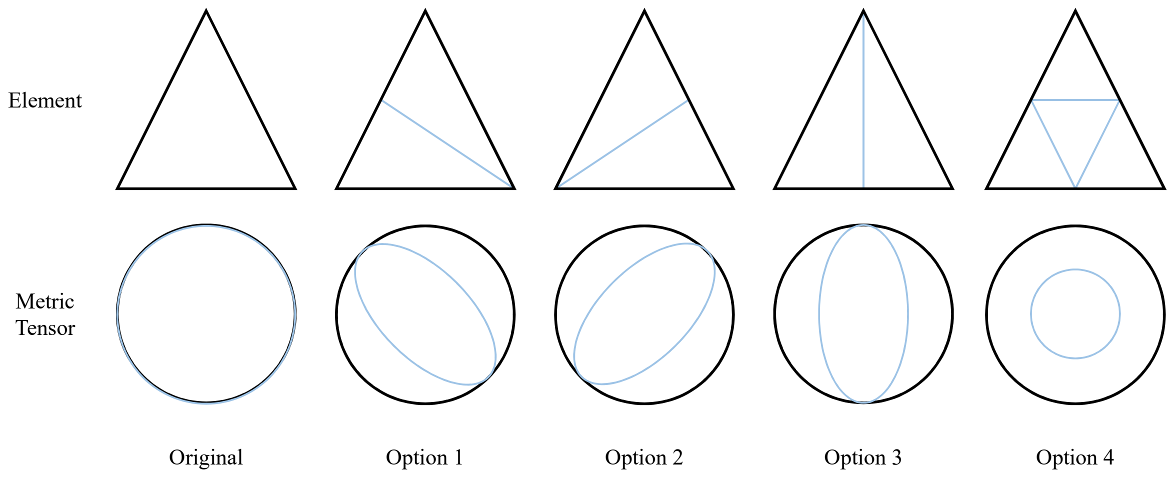


Figure 5.5: Mesh refinement options with corresponding changes in the metric for triangular elements.

Instead of computing new error estimates for each sample by refining the element with the proposed cut and solving the primal and adjoint problems each time, Yano [91] proposed solving them on just a subset of the mesh to limit the overall cost. In this work, the process is further simplified by using an element-local projection method to approximate the fine-space adjoint in semi-refined spaces associated with each element [92]. There is no solving of primal and adjoint problems in this approach.

Consider one element, Ω_e. The fine-space adjoint $\psi_h^{p+1}|_{\Omega_e}$ provides an estimate of the output error in the current order p solution. This error, $\mathcal{E}_{e,0}$ is measured relative to a *truth* solution of order $p+1$. Refining an element with option i in Figure 5.5 creates a solution space that is finer than the original space. However, this semi-refined space is not as fine as the $p+1$ space, so we will still assume the truth solution is at $p+1$. For an order p adjoint on the semi-refined space, $\boldsymbol{\psi}^p_{hi}|_{\Omega_e}$, we can compute an error indicator $\Delta\mathcal{E}_{ei}$. This estimates the error between the coarse solution and refinement option i. The remaining error associated with refinement option i is given by the difference

$$\mathcal{E}_{ei} \equiv \mathcal{E}_{e0} - \Delta\mathcal{E}_{ei}. \tag{5.24}$$

The calculation of $\Delta\mathcal{E}_{ei}$ requires projecting the fine-space adjoint $\boldsymbol{\psi}^{p+1}_{hi}|_{\Omega_e}$ down to the space of refinement option i and order p, and then back up to order $p+1$ on the original element. From there, adjoint-weighted residuals can be calculated on the original element. For the output-based adjoint from Chapter 3, mathematically this can be expressed as

$$\Delta\mathcal{E}_{ei} \equiv \left| \boldsymbol{\mathcal{R}}^{p+1}_h \left(\mathbf{u}^p_h, \delta\tilde{\boldsymbol{\psi}}^p_{hi}|_{\Omega_e} \right) \right|, \tag{5.25}$$

where $\tilde{\boldsymbol{\psi}}^p_{hi}$ is the fine-space adjoint $\boldsymbol{\psi}^{p+1}_{hi}$ projected from order $p+1$ to order p on refinement option i, and then back up to order $p+1$ on the original element. The quantity $\delta\tilde{\boldsymbol{\psi}}^p_{hi}$ is then defined as $\delta\tilde{\boldsymbol{\psi}}^p_{hi} \equiv \tilde{\boldsymbol{\psi}}^p_{hi} - \boldsymbol{\psi}^p_h$. Note that all projections are done in a least-squares sense in element-reference space. The form of Equation 5.25 that corresponds to the entropy variables is identical except that $\mathbf{v}$ replaces $\boldsymbol{\psi}$.

To summarize, the error found using refinement option i, $\Delta\mathcal{E}_{ei}$, is calculated using the adjoint-weighted residual in Equation 5.25 with all calculations occurring at order $p+1$ on the original element e. The breakdown of $\tilde{\boldsymbol{\psi}}^p_{hi}$ into the individual projections can be viewed pictorially in Figure 5.6. Using least-square projections in reference space, the combination of projections can

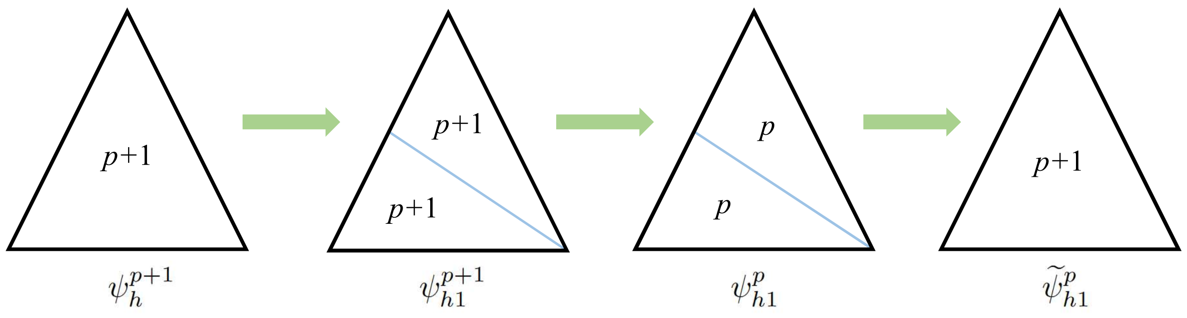


Figure 5.6: Projection of the fine-space adjoint, $\boldsymbol{\psi}^{p+1}_h$, down to the space created by refinement option i, and then back up to an order $p+1$ element.

be combined into one transfer matrix that converts $\boldsymbol{\psi}_h^{p+1}$ into $\tilde{\boldsymbol{\psi}}_{hi}^p$, both represented in the order $p+1$ space on the original element:

$$\widetilde{\boldsymbol{\psi}}_{hi}^p = \mathbf{T}_i \boldsymbol{\psi}_h^{p+1}, \tag{5.26}$$

$$\mathbf{T}_i = \left[\boldsymbol{\mathcal{M}}_0\left(\phi_0^{p+1}, \phi_0^{p+1}\right)\right]^{-1} \sum_{k=1}^{n_i} \mathbf{T}_{ik}, \tag{5.27}$$

$$\mathbf{T}_{ik} = \mathbf{M}_k\left(\phi_0^{p+1}, \phi_k^p\right)\left[\mathbf{M}_k\left(\phi_k^p, \phi_k^p\right)\right]^{-1} \mathbf{M}_k\left(\phi_k^p, \phi_k^{p+1}\right)\left[\mathbf{M}_k\left(\phi_k^{p+1}, \phi_k^{p+1}\right)\right]^{-1} \mathbf{M}_k\left(\phi_k^{p+1}, \phi_0^{p+1}\right). \tag{5.28}$$

In the above equations, n_i is the number of sub-elements within refinement option i, k is an index over the sub-elements, ϕ_k^p, ϕ_k^{p+1} are order p and $p+1$ basis functions on sub-element k, ϕ_0^p, ϕ_0^{p+1} are order p and $p+1$ basis functions on the original element, and components of the mass-like matrices are defined as

$$\mathbf{M}_k\left(\phi_l, \phi_m\right) = \int_{\Omega_k} \phi_l \phi_m d\Omega, \qquad \mathbf{M}_0\left(\phi_l, \phi_m\right) = \int_{\Omega_0} \phi_l \phi_m d\Omega, \tag{5.29}$$

where Ω_0 is the original element and Ω_k is the sub-element k. To reduce the computational cost, we calculate the transfer matrix, $\mathbf{T}_i$, in reference space just one time for each refinement option. It is then used for all elements to calculate $\Delta\mathcal{E}_{e,i}$ so that no solutions or residual evaluations are required for that element.

At this point in the process, the remaining error after each refinement option i, $\mathcal{E}_{e,i}$, can be used to estimate the rate tensor, $\boldsymbol{\mathcal{R}}_e$. This can be accomplished using a least-squares regression with the original error indicator $\mathcal{E}_{e,0}$. Using Equation 5.22, we can derive a regression to minimize the following error,

$$\sum_i \left[\log\frac{\mathcal{E}_{e,i}}{\mathcal{E}_{e,0}} - \operatorname{tr}\left(\boldsymbol{\mathcal{R}}_e \boldsymbol{\mathcal{S}}_{e,i}\right)\right]^2, \tag{5.30}$$

where the summation is over all of the refinement options. In Equation 5.30, $\boldsymbol{\mathcal{S}}_{e,i}$ is the step matrix that is associated with the refinement option i, given by (from Equation 5.5),

$$\boldsymbol{\mathcal{S}}_{e,i} = \log\left(\boldsymbol{\mathcal{M}}_0^{-1/2} \boldsymbol{\mathcal{M}}_i \boldsymbol{\mathcal{M}}_0^{-1/2}\right). \tag{5.31}$$

where $\boldsymbol{\mathcal{M}}_i$ is the affine-invariant metric average of the mesh-implied metrics of all sub-elements within refinement option i. Differentiating Equation 5.30 with respect to components of the rate tensor $\boldsymbol{\mathcal{R}}_e$ produce a linear system for each component. These can also be viewed as the normal equations for the least-square problem.

5.4.3 Cost Model

Elemental degrees of freedom, `dof`, are used to measure the cost of refinement. Using Equation 5.4 and properties of the metric tensor, assuming the step matrix $\boldsymbol{\mathcal{S}}_e$ is applied to the metric of element e, the area of the element decreases by $\exp\left[\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\mathcal{S}}_e\right)\right]$. This means that the number of new elements that occupied the original area Ω_e increases by that factor. Under these assumptions, the element cost model can be defined as

$$\mathcal{C}_e = \mathcal{C}_{e,0} \exp\left[\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\mathcal{S}}_e\right)\right] \Rightarrow \frac{\partial \mathcal{C}_e}{\partial \boldsymbol{\mathcal{S}}_e} = \mathcal{C}_e \frac{1}{2}\mathcal{I}, \tag{5.32}$$

where $\mathcal{C}_e$ is the expected cost over the element area after applying $\boldsymbol{\mathcal{S}}_e$ to the elemental metric and $\mathcal{I}$ is the identity matrix. Just as with the total error, the total cost over the entire computational domain is the sum of each elemental cost, which we define as

$$\mathcal{C} = \sum_{e=1}^{N_e} \mathcal{C}_e. \tag{5.33}$$

5.4.4 Algorithm for Mesh Optimization

The goal of the metric optimization algorithm is to determine the step matrix $\boldsymbol{\mathcal{S}}$ that minimizes the error given a fixed cost. Given the original mesh with its mesh-implied metric $(\boldsymbol{\mathcal{M}}(\vec{x}))$ and the elemental error indicators $\mathcal{E}_{e,0}$, once we have determined $\mathcal{E}_e$, $\mathcal{C}_e$, and $\boldsymbol{\mathcal{R}}_e$ we can proceed with this algorithm.

The step matrix field is approximated at mesh vertices of $\boldsymbol{\mathcal{S}}_v$, which are arithmetically-averaged to adjacent elements:[1]

$$\boldsymbol{\mathcal{S}}_e = \frac{1}{|V_e|} \sum_{v \in V_e} \boldsymbol{\mathcal{S}}_v, \tag{5.34}$$

where V_e is the set of vertices that are adjacent to the element e ($|V_e|$ is the total number of them). The optimization problem is to determine $\boldsymbol{\mathcal{S}}_v$ so that the error $\mathcal{E}$ is at a minimum for a given cost $\mathcal{C}$. The first-order optimality condition requires derivatives of the error and cost with respect to $\boldsymbol{\mathcal{S}}_v$, which can be expressed mathematically as

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{\mathcal{S}}_v} + \lambda_v \frac{\partial \mathcal{C}}{\partial \boldsymbol{\mathcal{S}}_v} = \mathbf{0}, \tag{5.35}$$

where λ_v is a Lagrange multiplier that is associated with the fixed-cost constraint. The error and

[1]Affine-invariant averages are not necessary because the entries of $\boldsymbol{\mathcal{S}}$ are coordinate-system independent.

cost derivatives in Equation 5.35 can be found using the chain rule on Equation 5.22 and Equation 5.32 with Equation 5.34 to produce

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{\mathcal{S}}_v} = \sum_{e \in E_v} \underbrace{\frac{\partial \mathcal{E}_e}{\partial \boldsymbol{\mathcal{S}}_e}}_{\text{Eqn.5.22}} \underbrace{\frac{\partial \boldsymbol{\mathcal{S}}_e}{\partial \boldsymbol{\mathcal{S}}_v}}_{1/|V_e|}, \qquad \frac{\partial \mathcal{C}}{\partial \boldsymbol{\mathcal{S}}_v} = \sum_{e \in E_v} \underbrace{\frac{\partial \mathcal{C}_e}{\partial \boldsymbol{\mathcal{S}}_e}}_{\text{Eqn.5.22}} \underbrace{\frac{\partial \boldsymbol{\mathcal{S}}_e}{\partial \boldsymbol{\mathcal{S}}_v}}_{1/|V_e|}, \tag{5.36}$$

where E_v is the set of elements adjacent to vertex v. We note that the total cost only depends on the trace of the step matrix, which means that the trace-free part of $\boldsymbol{\mathcal{S}}_e$ does not alter the area of the element: it just allows for stretching. We can then divide the step matrix into trace and trace-free parts as

$$\boldsymbol{\mathcal{S}}_v = \underbrace{s_v \boldsymbol{\mathcal{I}}}_{\text{trace}} + \underbrace{\widetilde{\boldsymbol{\mathcal{S}}_v}}_{\text{trace-free}}, \tag{5.37}$$

where $s_v = \text{tr}(\boldsymbol{\mathcal{S}}_v)/d$. Derivatives of the error with respect to s_v and $\widetilde{\boldsymbol{\mathcal{S}}_v}$ are

$$\frac{\partial \mathcal{E}}{\partial s_v} = \text{tr}\left(\frac{\partial \mathcal{E}}{\partial \boldsymbol{\mathcal{S}}_v}\right), \qquad \frac{\partial \mathcal{E}}{\partial \widetilde{\boldsymbol{\mathcal{S}}_v}} = \frac{\partial \mathcal{E}}{\partial \boldsymbol{\mathcal{S}}_v} - \frac{\partial \mathcal{E}}{\partial s_v}\frac{\boldsymbol{\mathcal{I}}}{d}. \tag{5.38}$$

The optimization algorithm presented by Yano [91] is as follows:

1. Given a mesh, solution, and adjoint, calculate $\mathcal{E}_e$, $\mathcal{C}_e$, and $\boldsymbol{\mathcal{R}}_e$ for every element e.
2. Set $\delta s = \delta s_{\text{max}}/n_{\text{step}}$, $\boldsymbol{\mathcal{S}}_v = \mathbf{0}$.
3. Start loop: $i = 1 \dots n_{\text{step}}$
   - (a) Calculate $\boldsymbol{\mathcal{S}}_e$ from Equation 5.34, $\partial \mathcal{E}_e/\partial \boldsymbol{\mathcal{S}}_e$ from Equation 5.22, and $\partial \mathcal{C}_e/\partial \boldsymbol{\mathcal{S}}_e$ from Equation 5.32.
   - (b) Calculate derivatives of $\mathcal{E}$ and $\mathcal{C}$ with respect to s_v and $\widetilde{\boldsymbol{\mathcal{S}}_v}$ using Equation 5.38.
   - (c) At each vertex create the ratio $\lambda_v = \frac{\partial \mathcal{E}/\partial s_v}{\partial \mathcal{C}/\partial s_v}$ and
     - Refine the metric for 30% of the vertices with the largest $|\lambda_v|$: $\boldsymbol{\mathcal{S}}_v = \boldsymbol{\mathcal{S}}_v + \delta s \boldsymbol{\mathcal{I}}$
     - Coarsen the metric for 30% of the vertices with the smallest $|\lambda_v|$: $\boldsymbol{\mathcal{S}}_v = \boldsymbol{\mathcal{S}}_v - \delta s \boldsymbol{\mathcal{I}}$
   - (d) Update the trace-free part of $\boldsymbol{\mathcal{S}}_v$ to enforce stationarity with respect to shape changes at fixed area: $\boldsymbol{\mathcal{S}}_v = \boldsymbol{\mathcal{S}}_v + \delta s(\partial \mathcal{E}/\partial \widetilde{\boldsymbol{\mathcal{S}}_v})/(|\partial \mathcal{E}/\partial s_v|)$.
   - (e) Rescale $\boldsymbol{\mathcal{S}}_v \to \boldsymbol{\mathcal{S}}_v + \beta \boldsymbol{\mathcal{I}}$, where β is a global constant calculated using Equation 5.32 to constrain the total cost to the desired `dof` value: $\beta = \frac{2}{d} \log \frac{\mathcal{C}_{\text{target}}}{\mathcal{C}}$, where $\mathcal{C}_{\text{target}}$ is the target cost.

Constant values in the above algorithm that work well in most instances are $n_{\text{step}} = 20$ and $\delta s_{\max} = 2\log 2$. In other words, λ_v is a ratio of the marginal error to the marginal cost of a step matrix increase (i.e. mesh refinement). In practice, the mesh optimization and flow/adjoint solution are performed multiple times for a given target cost $\mathcal{C}_{\text{target}}$ until the error stops changing or settles about a mean. From there, the target cost is increased until the error reaches a desirable level.

5.4.5 MOESS with Unsteady Problems

For this work, all unsteady adaptive simulations relied on MOESS for adaptation of the spatial problem. In this section, we briefly review the changes to the process outlined in the previous subsections, when using MOESS in the context of unsteady problems.

Given the cost allocations defined in Subsection 4.2.4, we can adapt both the temporal and spatial problems individually. Time is one dimension, so we simply use uniform refinement where the factor f^{time} changes N_t and all of the time steps are of equal size, $\Delta t = T/N_t$. It is unnecessary to use MOESS (or any other adaptive approach outlined in this chapter) to refine the temporal problem.

In this work, we use MOESS to adapt the spatial domain given that we can focus on the spatial problem individually. In the context of unsteady problems, the general MOESS process is mostly the same as outlined in the previous subsections. The main difference is that the error indicator used to sample the various refinement options is now based on unsteady adjoint-weighted residual. This means that we rewrite Equation 5.25 as

$$\Delta\epsilon_{ei}^{\text{space}} \equiv \int_0^T \left| \bar{\boldsymbol{\mathcal{R}}}_h^{p+1} \left(\mathbf{u}_h^p, \delta\tilde{\boldsymbol{\psi}}_{hi}^p|_{\Omega_e} \right) \right| dt, \tag{5.39}$$

where $\Delta\epsilon_{ei}^{\text{space}}$ is the spatial error found by refining spatially with refinement option i. Outside of this change, the rest of the MOESS process is identical to the steady case.

5.4.6 MOESS with SUPG

There are a few changes that need to be made to MOESS when using a continuous finite element method such as SUPG. In DG, the approximation is discontinuous across elements, so it is straightforward to apply a global subtraction over all of the elements of the coarse-space adjoint from the fine-space adjoint. However, for SUPG, information among elements is shared at the element boundaries. Therefore, before calculating $\Delta\mathcal{E}_{ei}$ in Equation 5.25, the coarse-space adjoint must again be subtracted from the fine-space adjoint for each element. This is not a global subtraction over all elements. Note that to obtain the coarse-space adjoint, the projection of the fine-space adjoint from $p + 1$ to p and then back up to $p + 1$ on refinement option i is done for each element individually. This eliminates any issues caused by the sharing of information across elements.

When using the SUPG discretization, elemental error indicators computed using Equation 3.18 need to be restricted based on how many elements each degree of freedom is in contact with. This is because the fine-space residuals and adjoints are not tied to each element individually. Instead, they are shared across all elements. Consequently, the contributions to the error indicators used to govern the local sampling approach, $\Delta\mathcal{E}_{ei}$, also must be equally distributed across elements using the approach shown in Figure 3.4.

CHAPTER 6

Results

The overarching message of the results presented in this chapter is how the various combined approaches for error estimation and mesh adaptation perform for various flow simulations relying on the sets of governing equations outlined in Section 2.1. This is done by directly comparing the numerical output error obtained using the combined approaches to the output-based adjoint approach and the entropy-variable approach. Sample meshes are shown for all cases, in order to show where the various adaptive error indicator approaches refine the computational domain.

Besides varying what types of governing equations are used, the different types of mesh adaptation strategies reviewed in Chapter 5 also vary across the cases. Results using both types of discretizations outlined in Chapter 2 are also directly compared against one another. Steady-state and unsteady simulations are considered in this chapter as well.

The results are broken up into three sections, with each one having a distinct theme. The first section shows five different cases using a variety of governing equations, the number of spatial dimensions, and types of combined approaches. All cases in this section are steady-state simulations only. The Discontinuous Galerkin (DG) discretization is used for all cases in this section as the focus is more on how does the combined approach perform for various applications.

The second section focuses on how the combined approaches using the Streamline-Upwind Petrov-Galerkin (SUPG) discretization compare to using DG. Two types of applications are shown in this section as the SUPG code developed for this work is limited to Euler equations reviewed in Subsection 2.1.2. However, each case is analyzed in great detail using both types of discretizations, different solution approximation orders, and different mesh adaptation strategies.

The purpose of the last section is to analyze how the combined approach performs for an unsteady simulation. In this case, a vertical gust is imposed on a NACA 0012 airfoil. The various adaptive error indicators approaches are tested with this case using both conservative and non-conservative error estimates.

6.1 Steady-State Simulations Using DG

The goal of this section is to compare adaptation using output-based adjoints, the entropy variables, and various applications of the combined approach for both two-dimensional and three-dimensional cases using various geometries. All cases are steady-state simulations and use the DG discretization. Some of the geometries, initial meshes, and solution parameters are similar to those used in previous works[86, 87, 92], while other cases were designed especially for this research. In total there are five different cases, each using multiple error indicator approaches, shown in Table 6.1. The results in this section were previously published in the following works [121, 133].

	NACA 0012 Airfoil (Inviscid)	**NACA 0012 Wing (Inviscid)**	**Diamond Airfoil (Inviscid)**	**NACA 0012 Airfoil (Viscous)**	**NACA 0004 Airfoil (RANS)**
Uniform	X				
Output	X	X	X	X	X
Entropy	X	X	X	X	X
Combined	X	X	X	X	X
Coarse Comb.			X		
Comb. w/ Mask			X		

Table 6.1: Error indicators used for each steady-state case using DG.

6.1.1 NACA 0012 Airfoil in Inviscid Flow: $M_\infty = 0.95$, $\alpha = 0°$

The first case of a steady-state simulation shown in this section is two-dimensional, inviscid flow over the NACA 0012 airfoil with a closed trailing edge and a farfield approximately 40 chord lengths away. The flow is transonic with a freestream Mach number of $M_\infty = 0.95$ and angle of attack of $\alpha = 0°$. The flow produces a fishtail shock at the trailing edge, as shown in Figure 6.1. This shock makes it necessary to add artificial viscosity to stabilize the flow in the shock's immediate vicinity. The presence of the shock makes this case an instructive example of where the entropy-variable approach will overly target regions of the domain consumed by the shock. Since shocks are prevalent in many flow simulations in an engineering setting, the tendency of the entropy-based approach to overly target these regions for refinement is a serious issue. However, this issue can be mitigated by combining the indicator obtained from the output-based adjoint with that obtained from the entropy variables.

The starting mesh of 572 elements is shown in Figure 6.2. It is an unstructured (initially structured but not after adaptation), quadrilateral mesh with quartic, $q = 4$, curved elements repre-

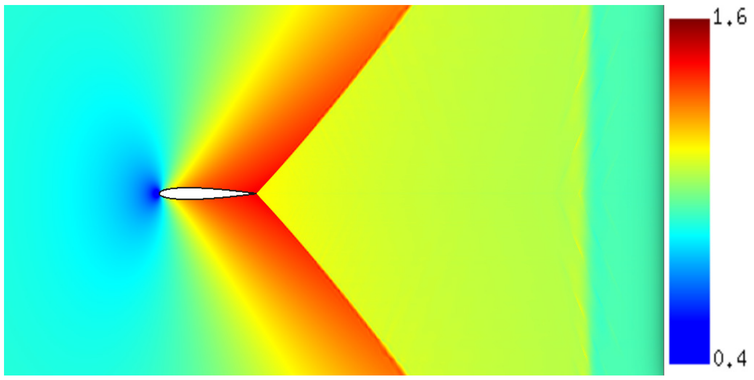


Figure 6.1: NACA 0012 $M_\infty = 0.95$, $\alpha = 0°$: Mach number contours (range: 0.4-1.6).

senting the airfoil geometry. The mesh adaptation strategy used in this case is the hanging node approach outlined in Section 5.1. The solution interpolation order for all of the cases is $p = 2$ and the adaptation fraction is set to $f^{\text{adapt}} = 0.1$ for each of the seven adaptation iterations. This means that for each adaptive iteration, only 10% of the elements are refined. To illustrate the effect this term has on the error convergence for the various methods, an additional set of runs were done with $f^{\text{adapt}} = 0.075$.

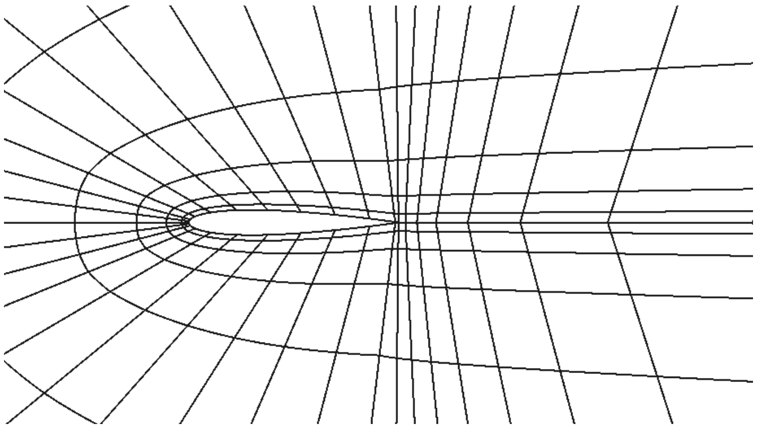


Figure 6.2: Initial NACA 0012 airfoil mesh.

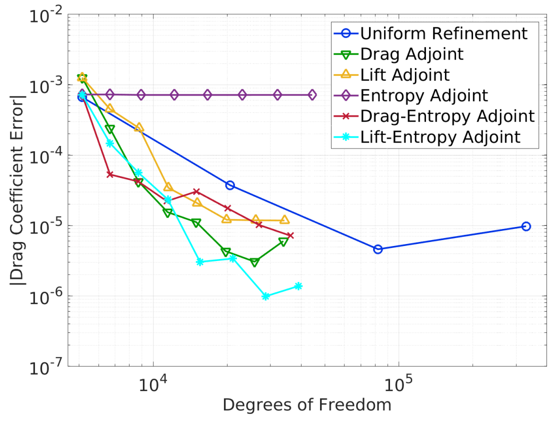
For this case, only one engineering output, the drag coefficient, was analyzed. Integrals of the inviscid momentum flux, i.e. the pressure on the airfoil surface, were used to obtain drag. Adaptation was driven using adjoints associated with the drag coefficient, lift coefficient, and entropy variables. In addition, combined output-based and entropy-variable adaptive solutions were generated. The combined approach considered for this case is the most straightforward approach, outlined in Subsection 3.4.1, where the error indicators using the fine-space output-based adjoint are multiplied by those obtained using entropy variables for each element.

The presence of shocks in inviscid flows leads to the creation of entropy in the regions where shocks are present. This creates a problem for the entropy-variable indicator since it will target regions of the mesh where spurious entropy is generated. The shocks may cause the entropy indicator to continue to refine regions of the mesh to which an engineering output function $J(\mathbf{u})$ may be insensitive. This not only leads to unnecessary mesh refinement in these regions, but also prevents regions of the mesh near the geometry to which $J(\mathbf{u})$ is sensitive from being refined, since only a certain percentage of cells are refined in each adaptive iteration. By combining the output-based adjoint indicator with the entropy-variable indicator through direct, elemental multiplication, this issue can be alleviated, since the output-based adjoint does not unnecessarily target regions of spurious entropy generation that do not affect $J(\mathbf{u})$.

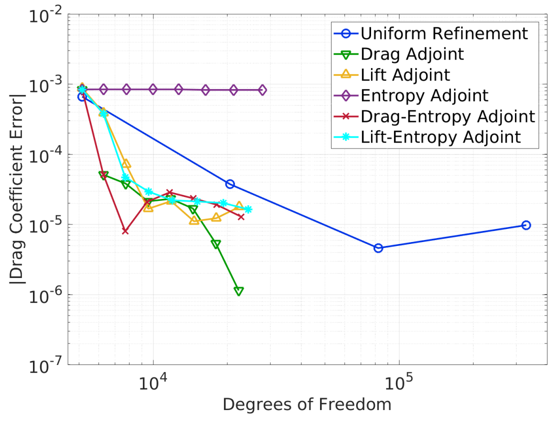
Figure 6.3 shows the convergence of the drag coefficient for various error indicator approaches and uniform refinement using both $f^{\text{adapt}} = 0.1$ and $f^{\text{adapt}} = 0.075$. One truth drag coefficient was obtained by averaging cases obtained using a solution approximation order of $p = 5$ on a mesh obtained by uniformly refining the finest drag coefficient adapted mesh twice. A second truth solution was obtained using the same uniformly refined mesh with $p = 4$. The difference in the drag coefficient between these two solutions is 6×10^{-9}.

The creation of entropy due to the fishtail shocks prevents the entropy-variable approach from properly refining the mesh to produce a more accurate drag coefficient. However, the combined approaches converge as the mesh is refined, since they are not as susceptible to unnecessary refinement in regions that do not affect the drag coefficient. When $f^{\text{adapt}} = 0.1$, the combined approach that incorporates the lift-based adjoint produces a better drag coefficient, while the combined approach that used the drag-based adjoint does not perform as well as the drag-based adjoint approach. However, this is not a universal result for this case, since the errors are sensitive to solver parameters, including coefficients in the artificial-viscosity-based shock-capturing strategy [134]. That is, adjustments to the shock-capturing smoothness indicator parameters can make the combined approach perform better. Despite changes to the shock capturing methods, however, the drag coefficient error convergence using the entropy-based indicator never improves. In addition, when $f^{\text{adapt}} = 0.075$, the combined lift-entropy case does not perform nearly as well. This is the only method that changes significantly when f^{adapt} is changed. It is reasonable to expect the error convergence to change since f^{adapt} will effectively change the relative meshes at each adaptive iteration. Thus, the significant change in the error convergence for the combined lift-entropy case is noteworthy, but not entirely surprising.

Figure 6.4 compares the meshes after seven adaptation iterations for the error indicator strategies in Figure 6.3 that use $f^{\text{adapt}} = 0.1$. The output-based adjoint methods do not target the fishtail shock, outside of the region directly near the trailing edge, whereas the entropy-variable approach focuses solely on the fishtail shock and does not refine the mesh anywhere else. This explains why in Figure 6.3, there is no change in drag coefficient error with each adapted mesh for the entropy-



(a) $f^{\text{adapt}} = 0.1$



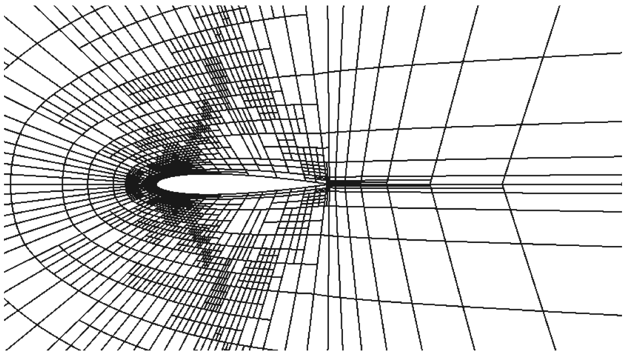
(b) $f^{\text{adapt}} = 0.075$

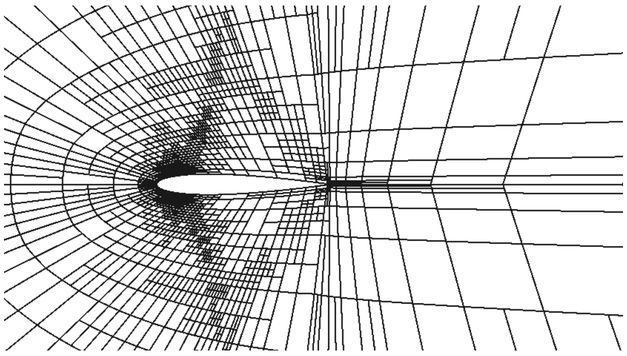
Figure 6.3: NACA 0012 $M_\infty = 0.95$, $\alpha = 0°$: Comparison of drag coefficient convergence histories for various error indicators using DG.

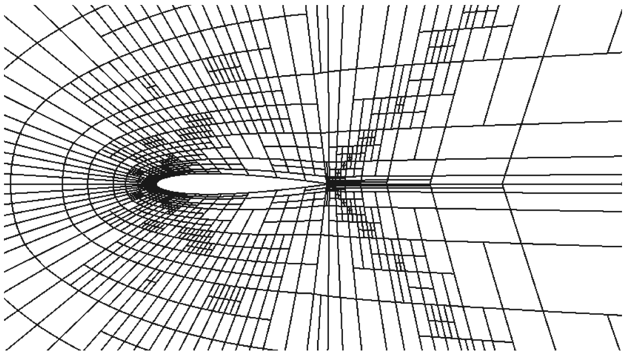
based indicator case. Since the mesh is barely refined near the airfoil surface (particularly the leading edge) the drag coefficient error never improves. The combined drag and entropy-variable approach yields a much-improved mesh compared to just using the entropy variables alone. However, there is still some unnecessary refinement due to the fishtail shock, because of how strong the entropy-based indicator is in that region. This is likely the reason why this combined approach does not perform as well as the drag-based alone approach. The combined lift and entropy-variable approach does not have as much refinement in the fishtail shock, which might explain why it performs much better than the combined approach that uses the drag-based adjoint. One option to remedy the issue with the drag-based combined approach is to use a mask on the entropy indicator so that its effect is not as pronounced. This option, outlined in Subsection 3.4.3, will be investigated for the results in Subsection 6.1.3.

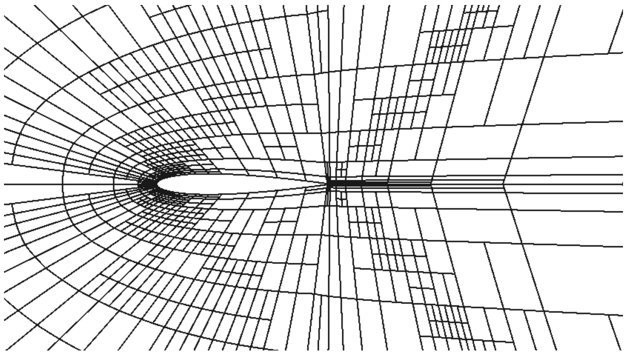
6.1.2 NACA 0012 Wing in Inviscid Flow: $M_\infty = 0.4$, $\alpha = 3°$

To demonstrate whether the combined approach is effective at higher dimensions, the next case is a NACA 0012, untapered, untwisted wing with a closed trailing edge and a rounded wing tip. The freestream Mach number is $M_\infty = 0.4$ and the angle of attack is $\alpha = 3°$. The wing has an aspect ratio of 10 and is inside a computational domain that has a farfield 40 chord lengths away. The unstructured mesh is composed of cubic, $q = 3$, hexahedral elements that curve to match the wing geometry. Quadratic solution interpolation, $p = 2$, was used in the discretization, but for these runs the adaptation fixed fraction was lowered to $f^{\text{adapt}} = 0.05$. Artificial viscosity stabilization [134] was added to the solution to provide stability in the vicinity of the trailing-edge vortices. The


(a) Drag Adaptation


(b) Lift Adaptation


(c) Drag-Entropy Adaptation


(d) Lift-Entropy Adaptation

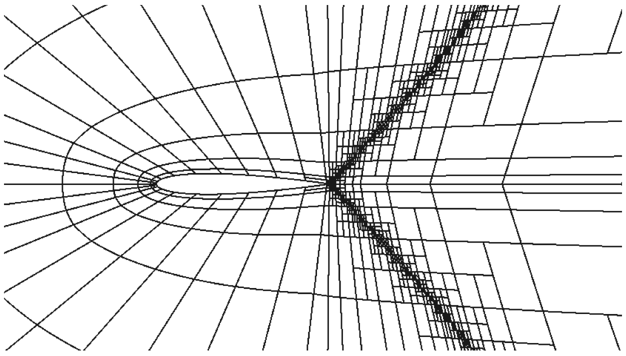
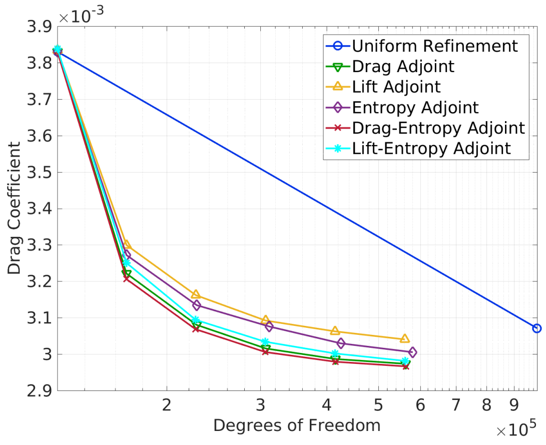

(e) Entropy Adaptation

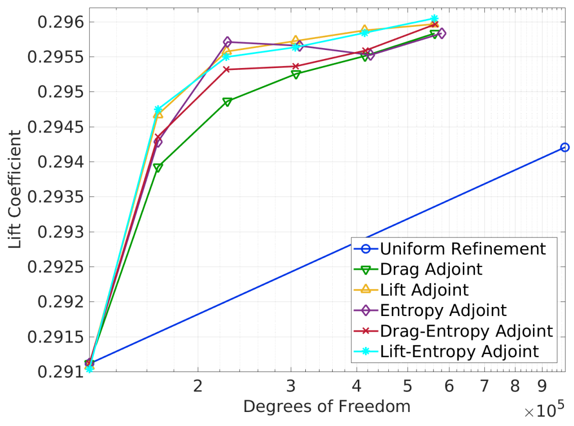
Figure 6.4: NACA 0012 $M_\infty = 0.95$, $\alpha = 0°$: Meshes after seven adaptation iterations for various error indicators that use $f^{\text{adapt}} = 0.1$ and DG.

cores of the trailing vortices experience very low pressures and become singularities in the inviscid limit, which is more closely approximated as the mesh is adapted in these regions. Without the addition of artificial viscosity stabilization, these regions then cause convergence problems for the high-order solver [87].

The two engineering outputs that are considered for this case are drag and lift coefficients. Adjoint solutions associated with these two outputs were used to drive separate adaptation runs. Additionally, the entropy-variable indicator, as well as the standard combined entropy and fine-space output-based adjoint indicator approach were tested. Figure 6.5 shows the results of these adaptation runs, along with a uniform refinement run for comparison. Note that one iteration of uniform refinement produced a mesh larger than any of the finest adapted meshes. This is why the uniform refinement convergence history in Figure 6.5 is linear between the two mesh sizes. Therefore, the difference in drag between using uniform refinement and any of the adaptive approaches for the intermediate mesh sizes may not be as large as it appears. No truth solutions were obtained due to the computational expense of such runs. Consequently, what is shown in these figures is not the output error, but the actual lift and drag coefficients.



(a) Drag Output



(b) Lift Output

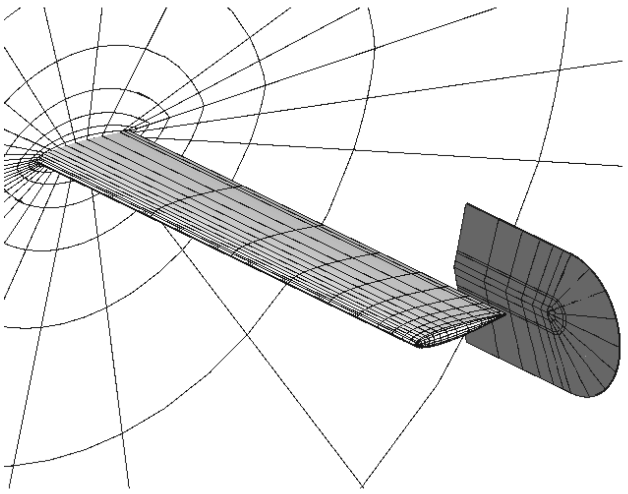
Figure 6.5: NACA 0012 Wing $M_\infty = 0.4$, $\alpha = 3°$: Comparison of output convergence histories for various error indicators using DG.

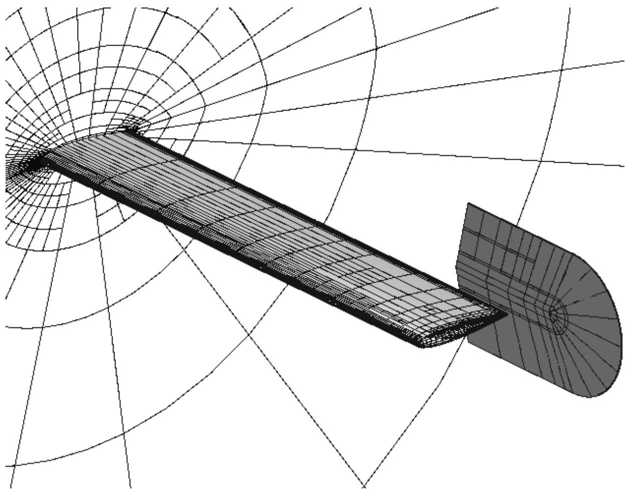
For the drag coefficient, the drag-based adjoint approach and combined drag and entropy-variable approach behave comparably and converge to a very similar value, assuming this value is close to the exact solution. It is possible that the combined approach might be slightly outperforming the output-based approach. However, as previously discussed, output-based adaptation is not always optimal because of adjoint singularities at the leading and trailing edges that can distract the adaptation. The entropy-variable approach does not converge to the same degree and particularly underperforms for the coarser meshes. Similar to what was observed for the drag coefficient, the lift-based adjoint approach and combined lift and entropy-variable approach perform comparably regarding their ability to predict the lift coefficient output. The entropy-variable indicator by itself is much more erratic and does not perform nearly as well. Not unexpectedly is that the lift-based adjoint approach does not produce as accurate of a drag coefficient as the drag-based adjoint and vice versa. Since the drag-based adjoint specifically refines the mesh to produce the most accurate drag coefficient, it is reasonable to assume that the lift-based adjoint approach, while producing the better lift coefficient, does not produce as accurate a drag coefficient. This same logic is the reason the combined approach that uses the drag-based adjoint does not produce as accurate of a lift coefficient compared to the combined approach that uses the lift-based adjoint.

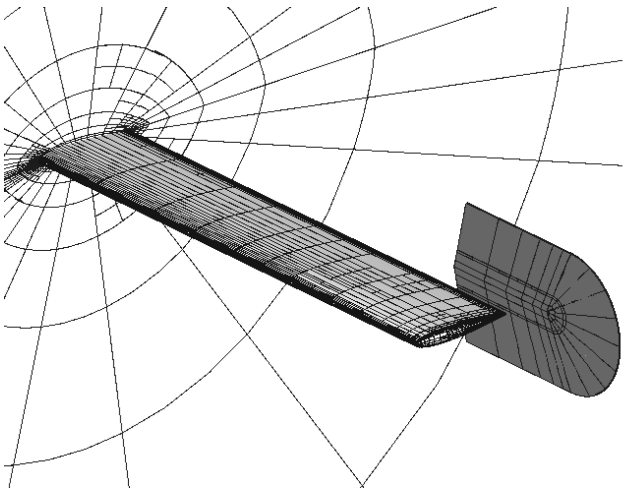
An explanation for why the entropy-variable indicator does not perform as well is its propensity to refine the mesh in areas consumed by the wing-tip vortex. From theory and experimental work [135], it is known that wakes form behind wings. This occurs due to the natural extension of the boundary layer from the wing surface, as well as streamwise vorticity that is shed due to lift distribution variation. This case is inviscid so there is no boundary layer. However, there still is shedding of streamwise vorticity, particularly at the wing tip. The numerical results presented in this paper show a very concentrated vortex emanating from the wing tip. Since the under-resolved wing-tip vortex leads to spurious entropy generation, the entropy-variable indicator targets areas of the mesh through which the vortex travels.

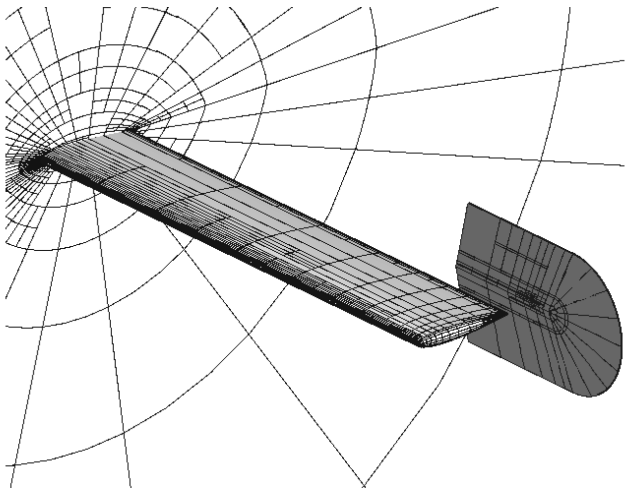
Figure 6.6 shows the initial mesh, as well as the final meshes, after five adaptive iterations for the various adaptive strategies. The cut plane in these images is not far behind the wing trailing edge. The output-based indicators refine the mesh near the wing surface and do not significantly refine the mesh downstream of the wing trailing edge. However, the entropy-variable indicator refines the regions of the mesh engulfed by the wake, especially where the wing-tip vortex is. While areas of the mesh, particularly those near the leading edge, are also refined, the entropy-variable indicator sacrifices refinement on the upper surface of the wing due to the excessive refinement in regions of the mesh where the wing-tip vortex is. The combined output-based and entropy-based adjoint indicators have some refinement due to the wing-tip vortex as to be expected, but it is not excessive enough to lead to poor adaptation performance, as illustrated by the results in Figure 6.5.

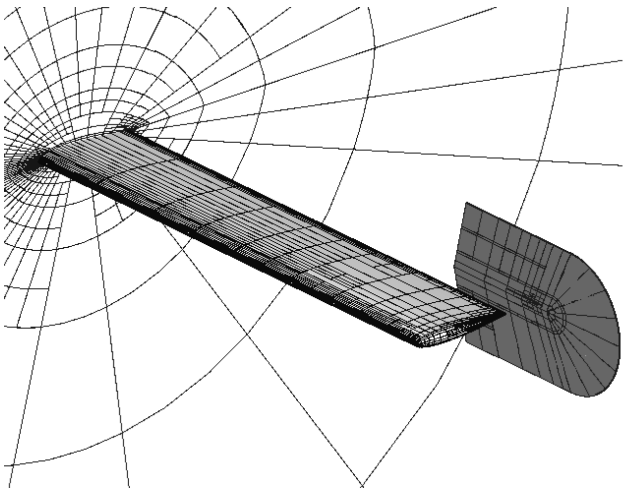
The impact of the wing-tip vortex can be further examined via the results in Figure 6.7, which shows cuts of the mesh at the wing tip. Again, the output-based adjoint indicators focus all of the refinement near the wing-tip surface, while the entropy-variable indicator focuses refinement downstream of the wing trailing edge due to the wing-tip vortex. The combined drag and entropy-variable indicator refines the region of the mesh consumed by the wing-tip vortex near the wing-tip surface, but does not excessively refine the mesh further downstream. It is reasonable to assume that some refinement due to the wing-tip vortex is necessary to accurately predict the drag coef-


(a) Initial Mesh


(b) Drag Adaptation


(c) Lift Adaptation


(d) Entropy Adaptation


(e) Drag-Entropy Adaptation

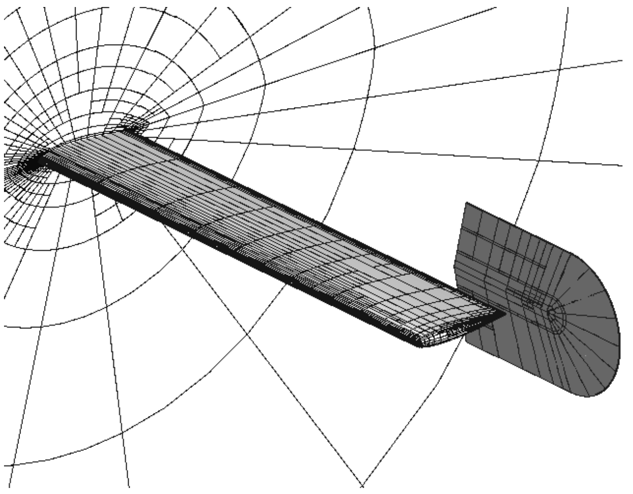
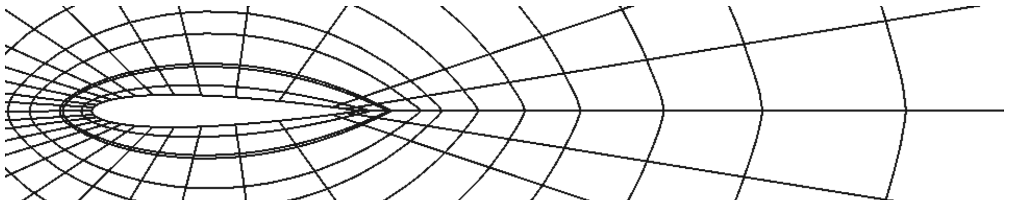
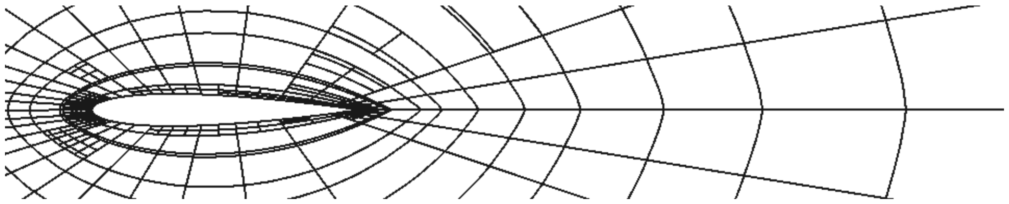
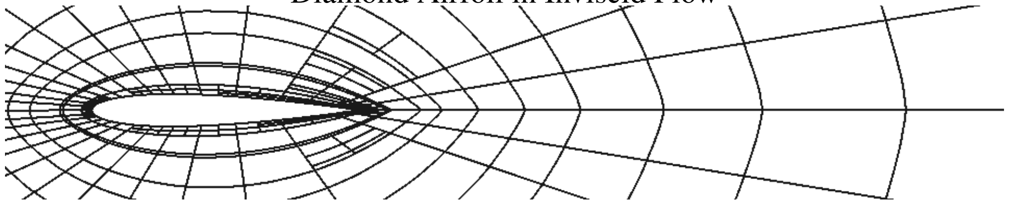

(f) Lift-Entropy Adaptation

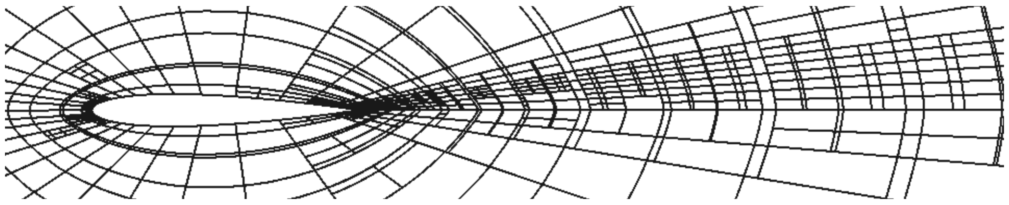
Figure 6.6: NACA 0012 Wing $M_\infty = 0.4$, $\alpha = 3°$: Initial mesh and meshes after five adaptation iterations for various error indicator strategies using DG.

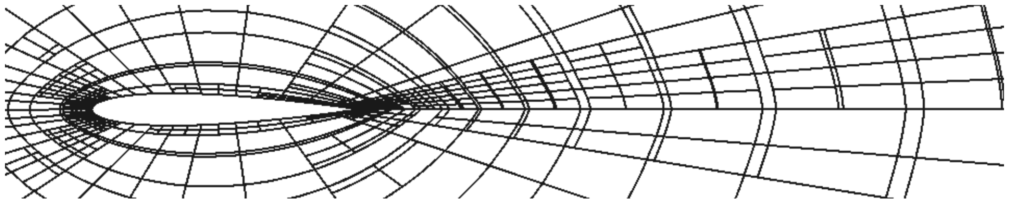

(a) Initial Mesh


(b) Drag Adaptation

Diamond Airfoil in Inviscid Flow


(c) Lift Adaptation


(d) Entropy Adaptation


(e) Drag-Entropy Adaptation

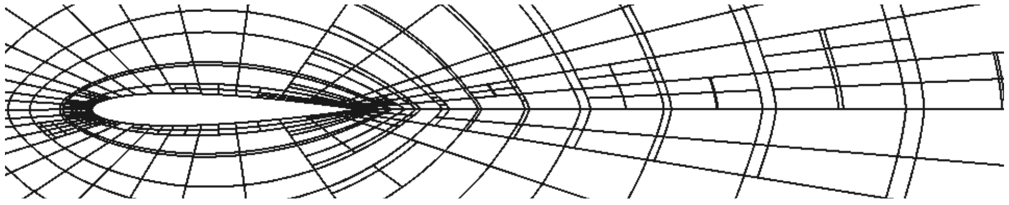

(f) Lift-Entropy Adaptation

Figure 6.7: NACA 0012 Wing $M_\infty = 0.4$, $\alpha = 3°$: Cut-plane at wing tip from initial mesh and meshes after five adaptation iterations for various error indicator strategies using DG.

ficient, but not nearly to the extent the entropy-variable approach indicates. The combined drag and entropy-variable indicator does a much better job balancing refinement between regions in the wake and near the airfoil surface. This is even more evident for the combined lift and entropy-variable indicator case, where the refinement due to the wing-tip vortex is even less pronounced since the wing-tip vortex under-resolution does not contribute to the lift coefficient error to the same degree as to the drag coefficient error.

6.1.3 Diamond Airfoil in Inviscid Flow: $M_\infty = 1.5$, $\alpha = 2°$

The next case is two-dimensional, inviscid flow over a thin diamond airfoil with a thickness ratio of 0.05. The freestream Mach number is $M_\infty = 1.5$ and the angle of attack is $\alpha = 2°$. The purpose of this simulation is to test the various modifications to the combined approach outlined in Subsections 3.4.2 and 3.4.3, particularly the masking on the entropy-based indicator. This case is also a relatively challenging case for the adaptive indicator since there are numerous flow discontinuities present. The grid is a square with a side length of 10 chord lengths. The supersonic freestream flow produces shocks emanating from both the top and bottom surfaces of the airfoil, as visible in Figure 6.8. However, because the airfoil is thin enough to produce weak shocks, no artificial viscosity was added to the solution. This is ideal since it is desirable to not have artificial viscosity present in the simulation, as it might adversely affect the entropy-based indicator.

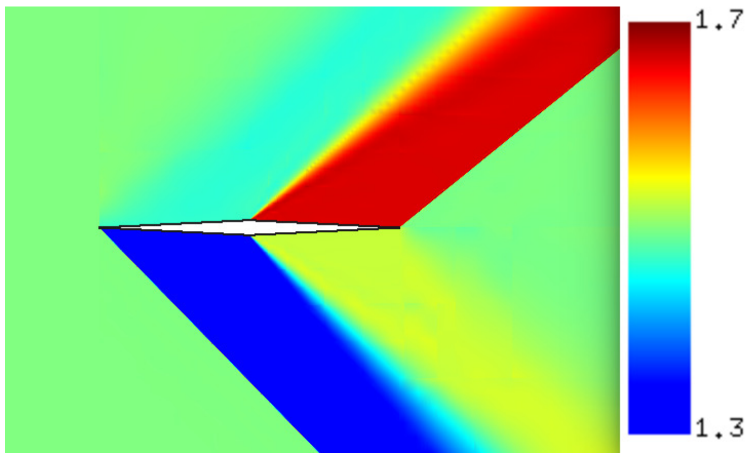


Figure 6.8: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Mach number contours (range: 1.3-1.7).

The airfoil is situated slightly off from the center of the domain so that the shocks emanating from the airfoil do not directly impact the corners of the mesh domain. Figure 6.9 shows the initial, unstructured, quadrilateral mesh made up of linear, $q = 1$, elements. As evident from the figure, the initial mesh is quite coarse. Just as with the inviscid NACA 0012 case, the solution approximation order for all of the cases is $p = 2$. All of the cases ran for exactly fourteen adaptive

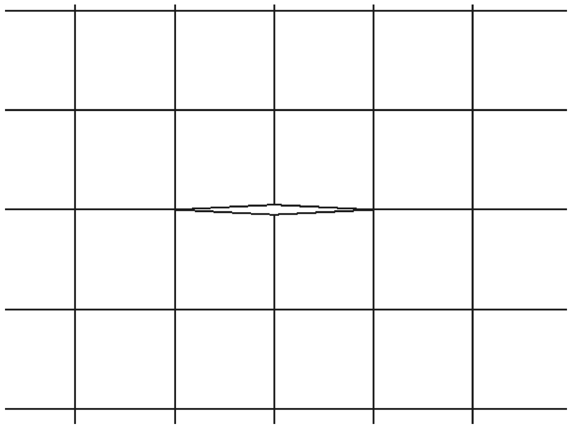
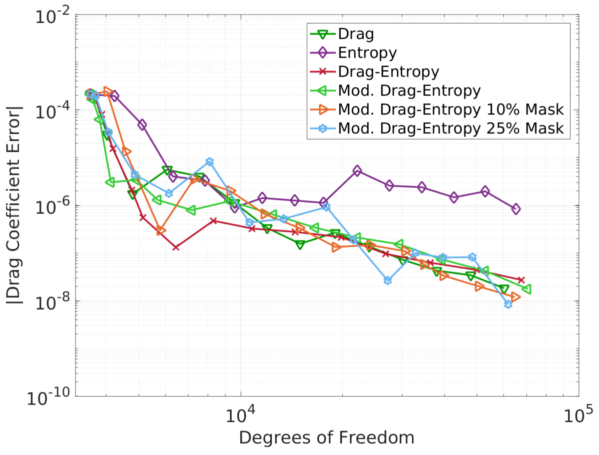


Figure 6.9: Initial diamond airfoil mesh made up of quadrilateral elements.

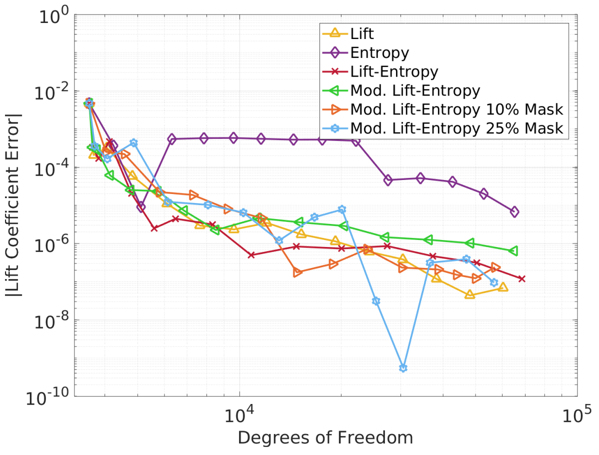
iterations, with the last iteration resulting in a mesh made up of approximately 70,000 degrees of freedom. This was done by using the hanging node adaptation approach and varying f^{adapt} for each case. In addition, for these runs the fixed-error fraction was set to $f^{\text{error}} = 0.95$.

For this case, the drag coefficient and the lift coefficient were again used as outputs to govern the mesh adaptation. Adaptation runs using adjoints based on these two outputs, as well as runs using both the entropy-based adjoint and the combined entropy and output-based adjoint were generated. In addition, three modified forms of the combined approach were considered for this case. The first modified combined approach, described in Subsection $3.4.2$, does not entail solving for the fine-space output-based adjoint. Instead, the adjoint and the state are projected down one order, to $p = 1$ for this case, so that the output-based indicator can be obtained at the $p = 2$ space. The second and third modified combined approaches build off of the first modified approach by adding a mask on the entropy-based indicator (see Subsection $3.4.3$) using different masking percentages, in this case 10% and 25%.

Figure 6.10 shows the drag and lift coefficient convergence results for all of the various error indicator strategies. As previously discussed, by varying f^{adapt} for each case, we were able to run each case with fourteen adaptive iterations that produced similar mesh sizes. The truth outputs for both lift and drag coefficient were obtained from a case at $p = 3$ on a mesh obtained by uniformly refining the finest output-adapted mesh. Another set of truth solutions was obtained by uniformly refining the initial truth solution and running a new case at $p = 3$. The difference in lift and drag coefficient between these two truth solutions was on the order of 2×10^{-9}. The modified combined approach, in which no fine-space output-based adjoints were solved for and the fine-space primal problem was approximated with an iterative solver, shows slightly downgraded performance, especially in terms of predicting the lift coefficient. However, the performance of this modified approach improves significantly with the addition of the mask on the entropy-based indicator. Unfortunately, no one particular masking percentage yields superior results for both lift and drag error estimation. The 10% mask is generally superior in terms of measuring drag coefficient. However, for lift coefficient error estimation, the 25% mask appears to perform better. This approach crosses the truth solution, which explains the sudden drop in error magnitude.



(a) Drag Output



(b) Lift Output

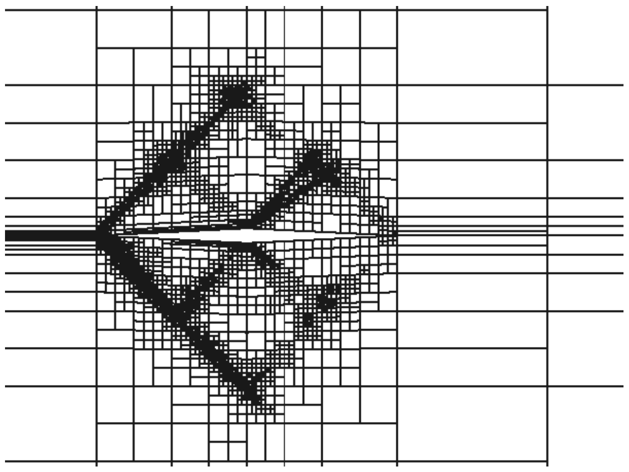
Figure 6.10: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Comparison of output convergence histories for various error indicator strategies using DG.

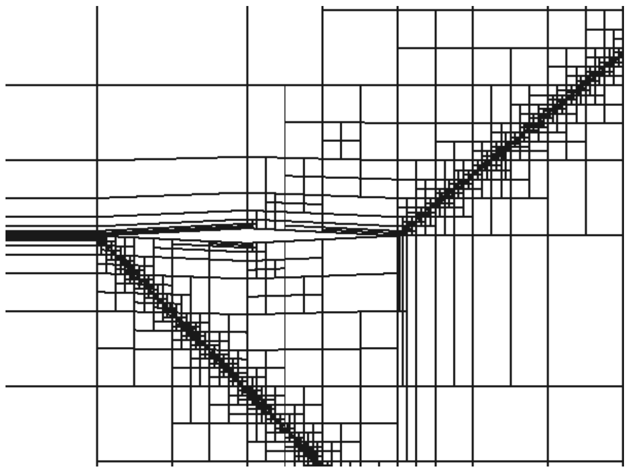
Figure 6.11 shows the meshes after the final adaptation iteration for the various error indicator strategies that, outside of the entropy-variable approach, all include indicators based on the drag-based adjoint. The entropy-variable indicator mostly focuses on the discontinuities emanating from the leading edge and trailing edge of the diamond. This refinement extends unnecessarily to the farfield boundaries, which accounts for why the error convergence of the entropy-based approach produces the least desirable error convergence. However, the entropy-variable approach still produces a better error estimate compared to the example in Subsection 6.1.1 where the error never improved as the mesh was refined. In that case, the shocks were nowhere near the leading edge, leading to no refinement there. For this supersonic case, there are shocks/expansions close to the airfoil that are refined. Thus, the error estimate predicted by the entropy-based approach is much better for this case. The drag-based adjoint indicator does not suffer from this issue, instead producing regions of refinement that (when examined as a whole) resemble the shape of a diamond. The regions of refinement that originate near the airfoil and then travel downstream are necessary for good error estimation. However, based on the four meshes obtained using a variation of the combined approach, the regions of refinement that travel upstream may not be necessary for accurate error estimation. These regions are refined because of the singularities in the output-based adjoint in the vicinity of the leading-edge stagnation streamline. The four combined approaches produce fairly similar drag coefficient error estimates, despite their meshes looking quite a bit different. The two masking cases, in particular, show a significant difference in refinement on the top of the airfoil. The refinement from the shocks above the airfoil is necessary since only at the last adaptive iteration does the 25% mask case outperform the 10% case. Of course, the discontinuity emanating from the nose that is below the airfoil is far more important since this region of the mesh is refined regardless of the method. Overall, a lower masking percentage appears to produce much more reliable meshes.

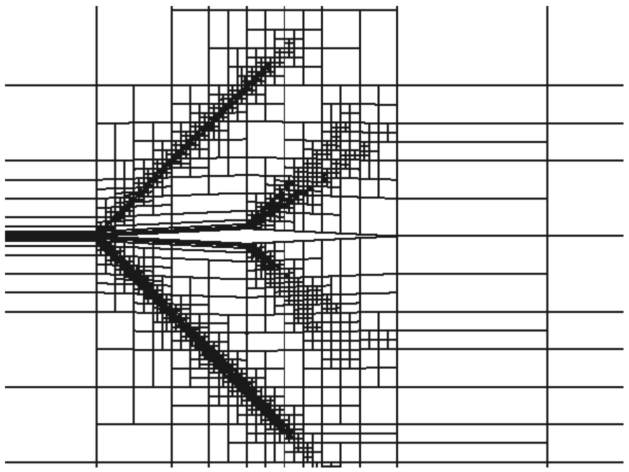
Cell-wise adaptive indicator plots on meshes after seven adaptive iterations are shown in Figure 6.12. The cases shown in this figure are identical to those shown in Figure 6.11, the main difference being that these meshes are after seven adaptive iterations, not fourteen. For all cases, the area of the mesh that gets refined first is the shock emanating from the nose that is below the airfoil. After seven iterations, this region is not nearly as refined in the entropy-variable case, compared to the drag-based adjoint case. This is because the mesh is refined in regions where the discontinuity is far from the airfoil. This region appears to be the most refined for the masking cases, which might explain their slight performance improvement. However, most of the cases produce similar drag coefficient error levels overall, despite the visible differences in these cell-wise adaptive indicator plots.

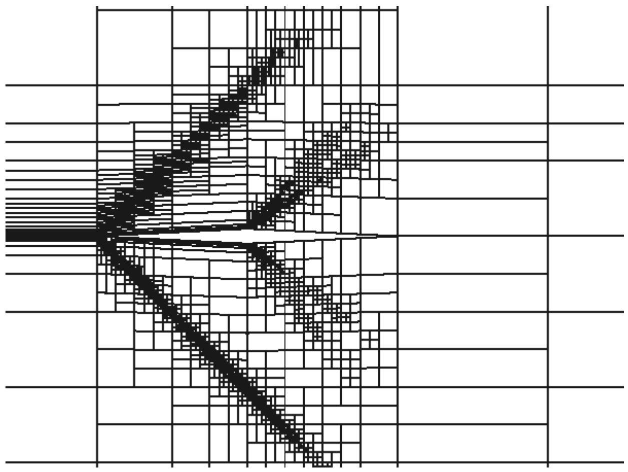
Figure 6.13 shows meshes after the final adaptation iteration for the various error indicator strategies that rely on the lift-based adjoint. For lift coefficient estimation, the modified combined approach does not perform nearly as well without the mask. The issue with this case is that many of the elements (particularly those in the region above the airfoil) have high aspect ratios. Since hanging node adaptation requires certain ratios between cell refinements, the very small grid cells near the top point and leading tip of the diamond airfoil lead to the propagation of large aspect ratio cells above the airfoil. These cells do a poor job of capturing the very thin discontinuity. The combined case with the 25% mask has many poor aspect ratio cells too, which is likely why the lift coefficient estimation is so erratic with each adaptive iteration. The high aspect ratio cells are not as prevalent in the output-based case and the combined case with the 10% mask, hence the improvement in the estimation of the lift coefficient.

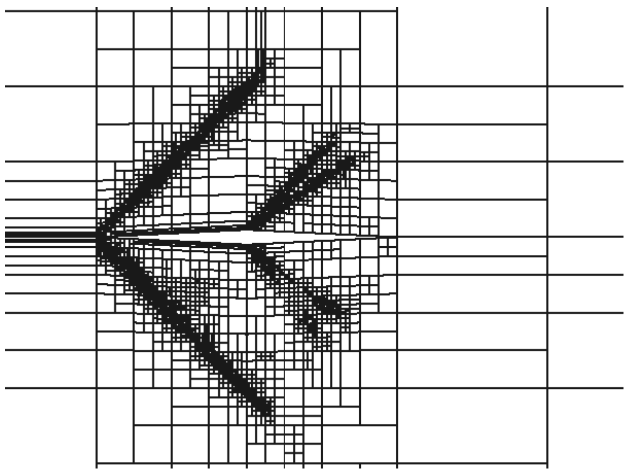
It is important to note that many masking percentages, as well as various values for f^{adapt} and f^{error}, were studied using this case. Additionally, masking was attempted on the output-based indicator as opposed to the entropy-based indicator. We have chosen to include combinations of inputs that yielded the best results for this particular case. However, no one particular combination of inputs always yields the best error convergence. Part of the issue in determining the optimal approach is the inherent limitations of the hanging-node adaptation discussed in the previous paragraph. To eliminate these limitations, the subsequent results in this section will focus on mesh adaptation using MOESS.


(a) Drag Adaptation


(b) Entropy Adaptation


(c) Drag-Entropy Adaptation


(d) Mod. Drag-Entropy Adaptation


(e) Mod. Drag-Entropy Adaptation 10% Mask

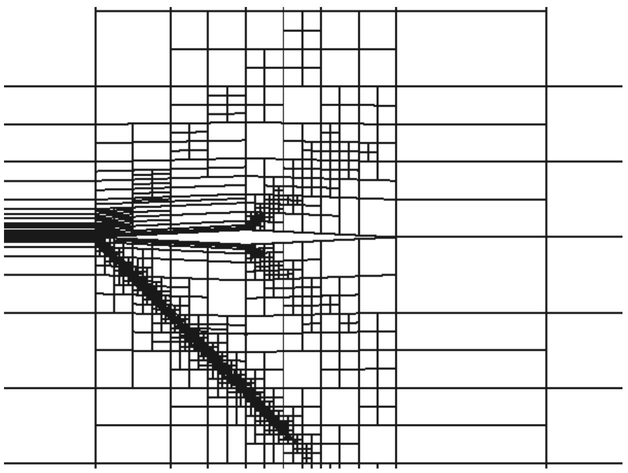
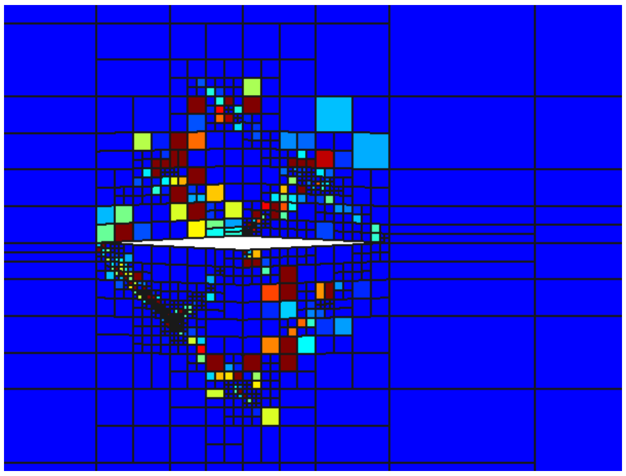
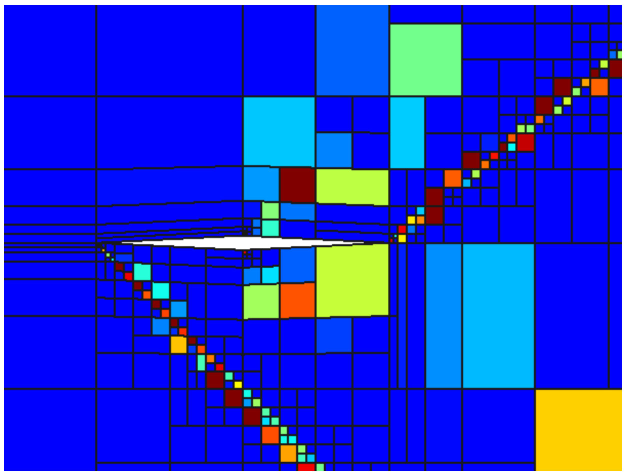
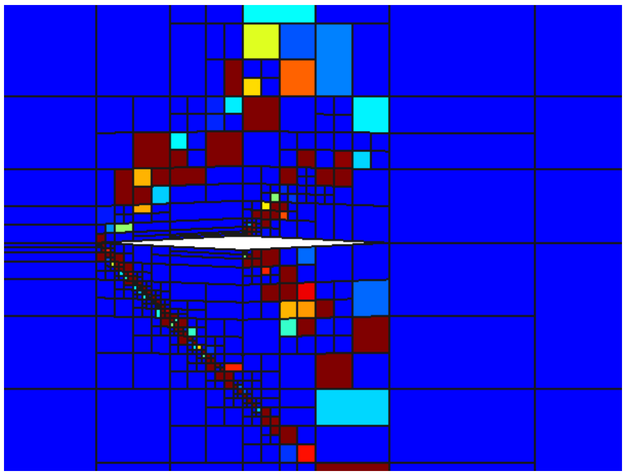
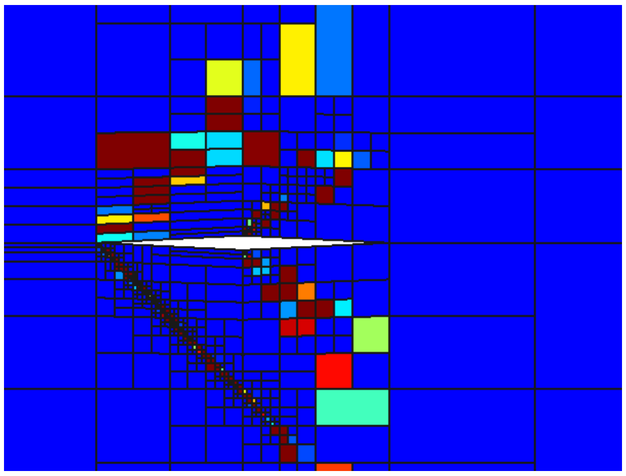

(f) Mod. Drag-Entropy Adaptation 25% Mask

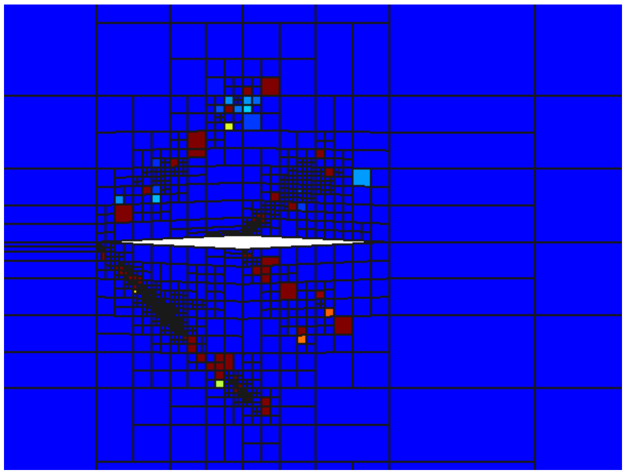
Figure 6.11: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Final mesh after adaptation iterations for various error indicator strategies using DG, most of which require the drag coefficient adjoint.


(a) Drag Adaptation


(b) Entropy Adaptation


(c) Drag-Entropy Adaptation


(d) Mod. Drag-Entropy Adaptation


(e) Mod. Drag-Entropy Adaptation 10% Mask

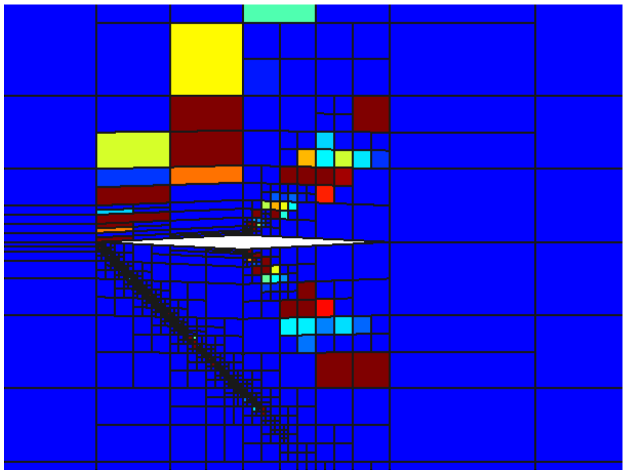
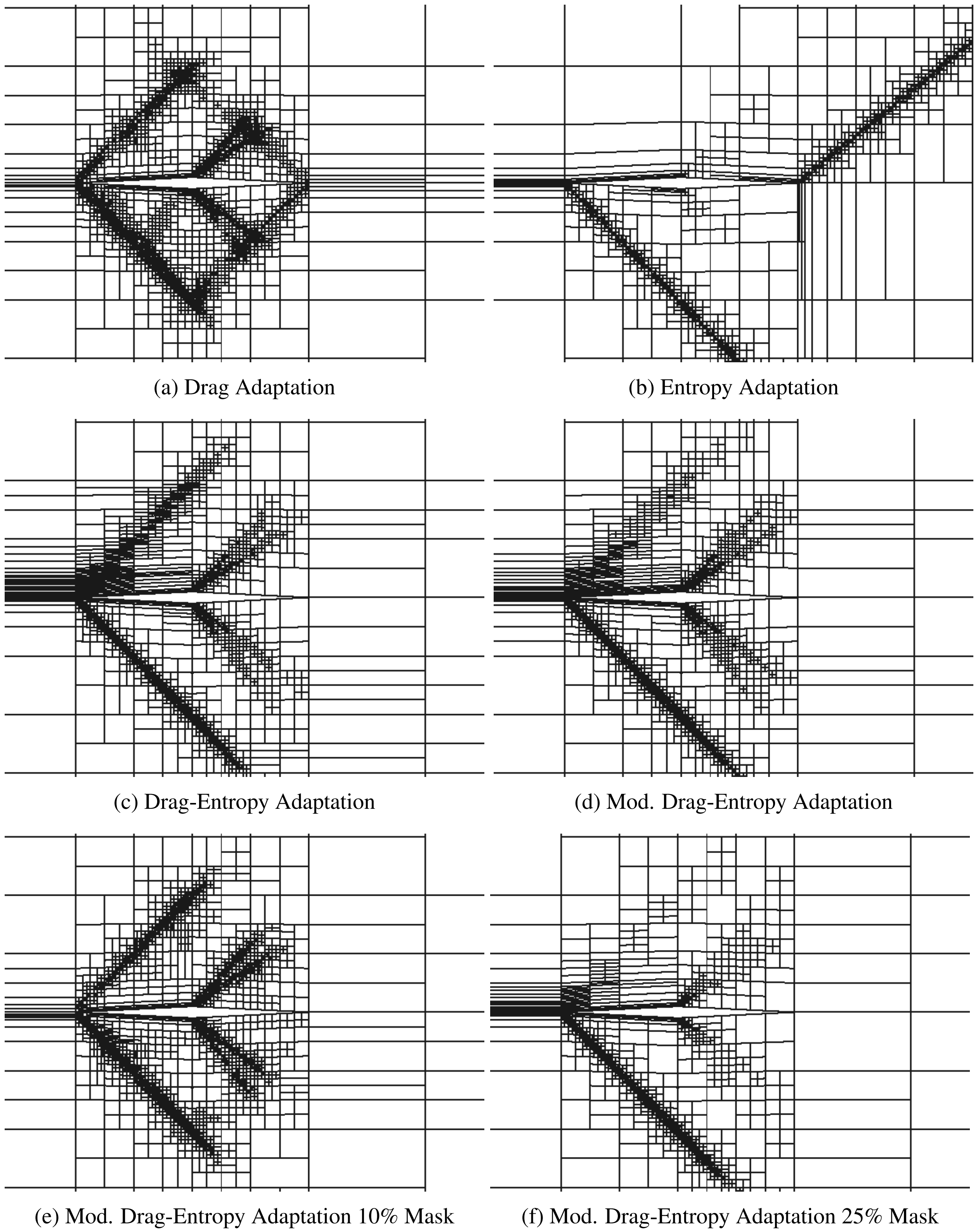

(f) Mod. Drag-Entropy Adaptation 25% Mask

Figure 6.12: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Adaptive indicator at seven adaptation iterations for various error indicator strategies using DG, most of which require the drag coefficient adjoint.



(a) Drag Adaptation

(b) Entropy Adaptation

(c) Drag-Entropy Adaptation

(d) Mod. Drag-Entropy Adaptation

(e) Mod. Drag-Entropy Adaptation 10% Mask

(f) Mod. Drag-Entropy Adaptation 25% Mask

Figure 6.13: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Final mesh after adaptation iterations for various error indicator strategies using DG, most of which require the lift coefficient adjoint.

6.1.4 NACA 0012 Airfoil in Viscous Flow: $M_\infty = 0.5$, $\alpha = 5°$, $Re = 5000$

To illustrate how using MOESS, outlined in Section 5.4, compares to hanging node adaptation, we consider two-dimensional, viscous flow over the NACA 0012 airfoil with a closed trailing edge. This case differs from the previous examples in that it is a viscous simulation. This allows us to demonstrate the effectiveness of the combined approach for a flow simulation with much more challenging governing flow equations. The initial meshes are shown in Figure 6.14. The freestream Mach number is $M_\infty = 0.5$, the angle of attack is $\alpha = 5°$, and the Reynolds number is set to $Re = 5000$.

The initial mesh used with hanging node adaptation is a quadrilateral mesh with quartic, $q = 4$, curved elements representing the geometry. The mesh is made up of 818 elements and the farfield is approximately 40 chord lengths away. The extra refinement near the surface is necessary to enable a solution on the coarsest mesh. The adaptation fraction was set to $f^{\text{adapt}} = 0.1$ and the adaptation fixed error fraction was set to $f^{\text{error}} = 1$ for each of the nine adaptation iterations. Finally, the solution was set to $p = 2$ for all cases.

The initial mesh used with MOESS consists of 356 unstructured triangles. The farfield is over 2000 chord-lengths away, and curved elements of geometry order $q = 4$ are used to represent the airfoil. The cases are run with a solution interpolation order of $p = 2$ with the following six degrees of freedom targets: 2000, 4000, 8000, 16000, 32000, and 64000. At each `dof` target, fifteen solution iterations are performed before moving to the next `dof` target.

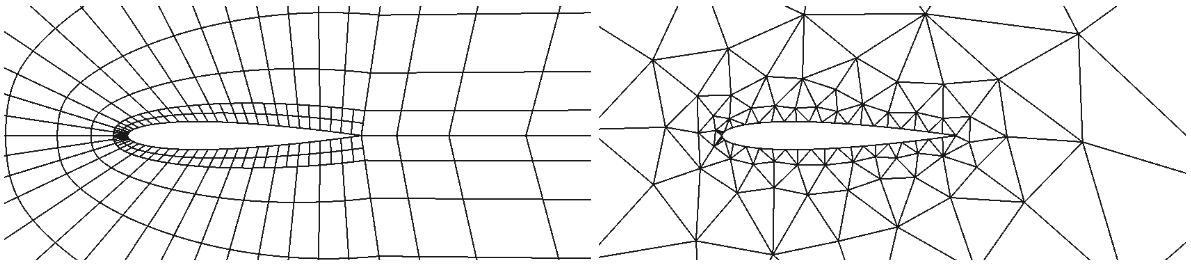
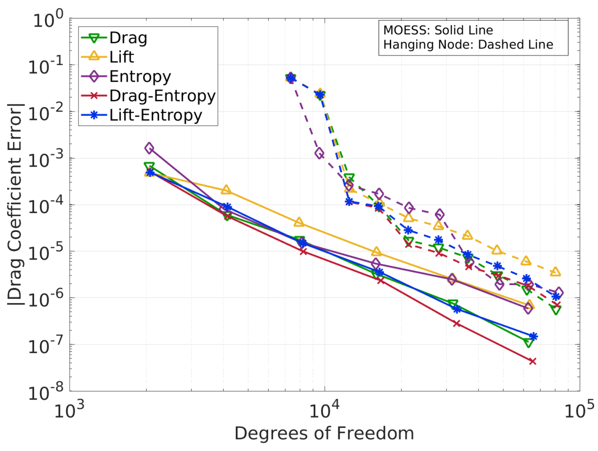
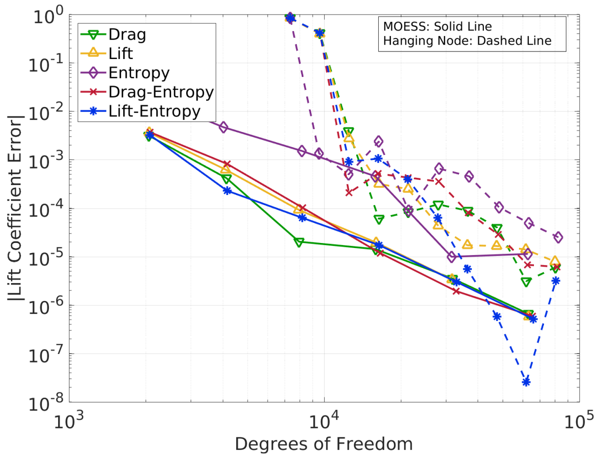

(a) Hanging Node (818 elements) (b) MOESS (356 elements)

Figure 6.14: Initial viscous NACA 0012 airfoil meshes.

For both sets of adaptation strategies, the drag coefficient and lift coefficients were used as outputs to drive the mesh adaptation. Adaptation runs using adjoints based on these outputs, the entropy-variable approach, and the standard combined entropy and fine-space output-based adjoint were considered. None of the previously aforementioned modifications were made to the combined approach in this case. Figure 6.15 shows the convergence results for all of the error indicators using both sets of mesh adaptation strategies. The solid lines represent solutions using MOESS, while the dashed lines represent solutions using hanging node adaptation. The output and `dof` values reported from the cases incorporating MOESS are averages over the last 5 solution iterations at each target `dof`. For the viscous case, the truth outputs for both the lift and drag coefficients were obtained by refining the final output-adapted mesh and running cases at orders $p = 4$ and $p = 5$. The difference in drag coefficient between these two sets of cases was of the order 4×10^{-8}, while the difference in lift coefficient was of the order 8×10^{-8}. For the MOESS case, truth solutions were run at $p = 3$ and $p = 4$. The difference in outputs between these two sets of cases was of the order 1×10^{-8}.



(a) Drag Output

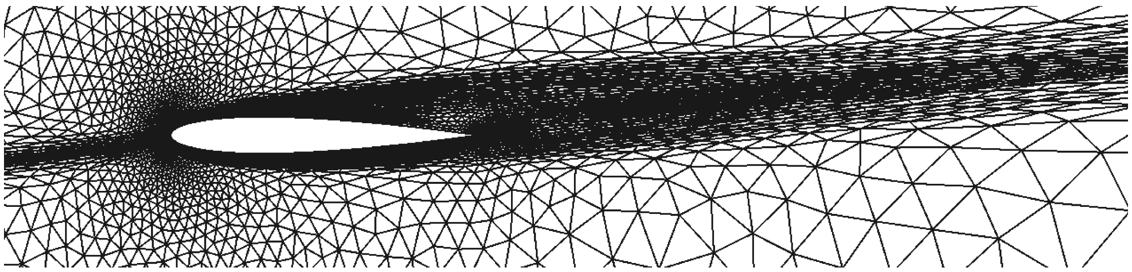


(b) Lift Output

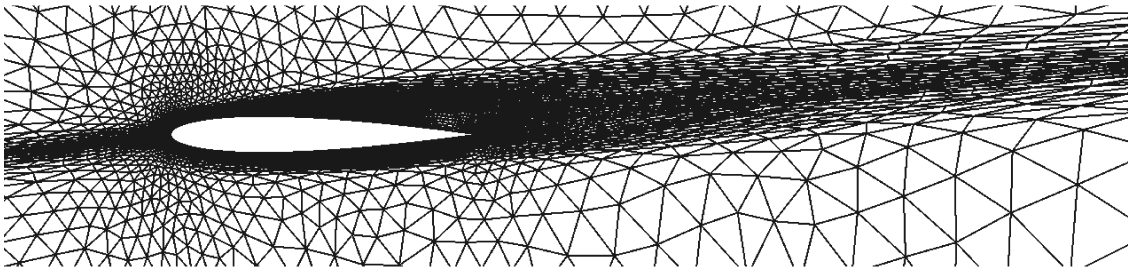
Figure 6.15: NACA 0012 $M_\infty = 0.5$, $\alpha = 5°$, $Re = 5000$: Comparison of output convergence histories for various error indicators using both hanging node and MOESS adaptation with DG.

Unsurprisingly, the MOESS adapted cases mostly perform much better than the hanging node adapted cases, since MOESS can coarsen regions of the mesh that do not affect the output error estimation. MOESS also considers elemental anisotropic information, which leads to greater accuracy. When implementing hanging node refinement, the error indicator strategies' drag coefficient error convergence does not vary significantly, outside of the lift-based adjoint approach whose relative error estimation is far inferior. However, for MOESS a clear distinction is present. The entropy variable-based adaptation solution is quite poor, comparable to the lift-based adjoint approach, while the combined approach that incorporates the drag-based adjoint yields better performance than the drag coefficient-based adaptation solution. In addition, the combined approach that uses the lift-based adjoint compares very well to the drag-based approach. For the lift coefficient error convergence, the combined approach using hanging node adaptation is superior to output-based hanging node adaptation. However, for MOESS there is little difference between the combined approach and the output-based approach.

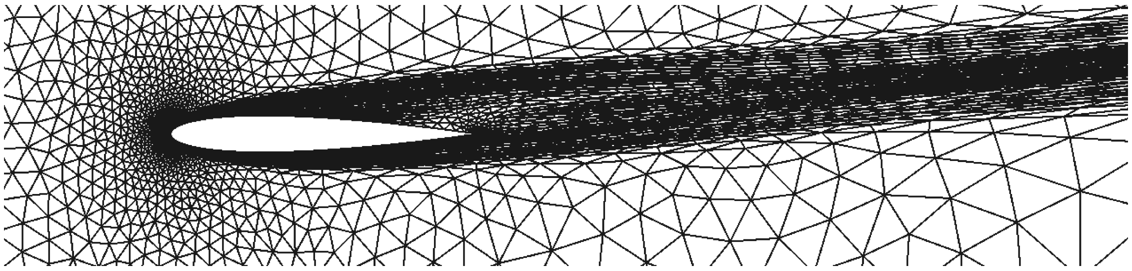
Figure 6.16 presents the meshes at 64000 `dof` for the cases that used MOESS adaptation. The



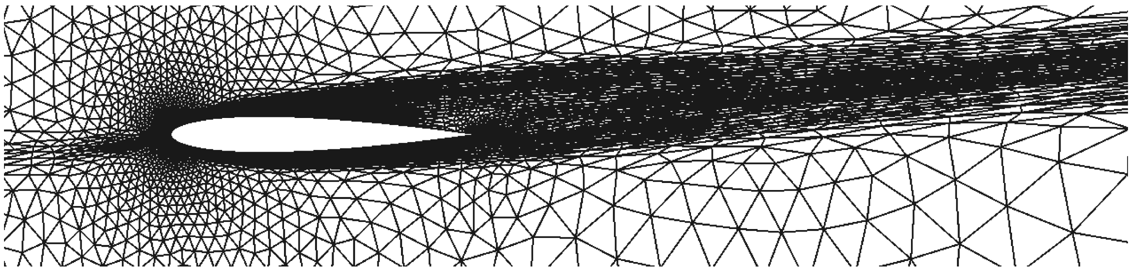
(a) Drag Adaptation



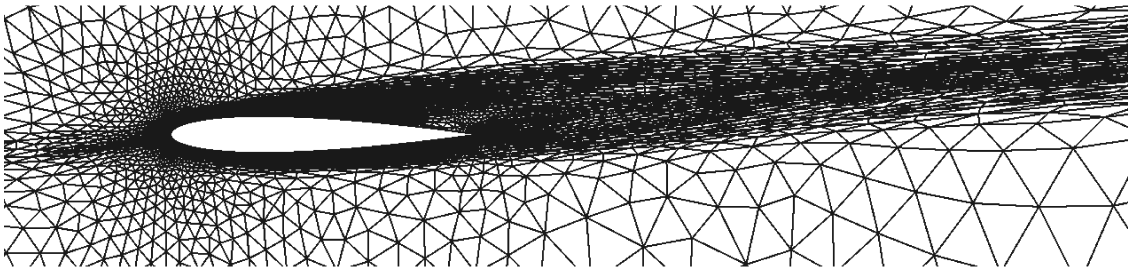
(b) Lift Adaptation



(c) Entropy Adaptation



(d) Drag-Entropy Adaptation



(e) Lift-Entropy Adaptation

Figure 6.16: NACA 0012 $M_\infty = 0.5$, $\alpha = 5°$, $Re = 5000$: Meshes after final adaptation iteration (approximately 64000 `dof`) for various error indicator strategies using DG.

individual output-based approaches target regions of the domain near the airfoil, as well as the portions of the wake near the airfoil and areas near the stagnation streamline. In fact, there is substantial refinement ahead of the stagnation streamline that is multiple chord lengths long. The entropy-based adjoint approach does not target regions of the mesh well ahead of the stagnation point. Instead, it targets the entire wake (all the way to the farfield boundary) since there is substantial spurious entropy generation in an under-resolved wake. The entropy indicator also leaves the aft portion of the upper surface of the airfoil relatively unrefined. The combined approach achieves a balance between refining ahead of the airfoil and in the wake. The meshes obtained from these approaches are refined in the wake, but not to the excessive extent present in the mesh produced from the entropy-based approach. In addition, these meshes have refinement ahead of the stagnation streamline, but the refinement does not extend well ahead of the airfoil as it does for the output-based approaches. This balance in refinement for this particular case clearly shows the benefit of using a combined approach to govern the mesh adaptation.

6.1.5 NACA 0004 Airfoil in RANS Flow: $M_\infty = 0.5$, $\alpha = 3°$, $Re = 1 \times 10^4$

To test the combined approach's effectiveness for even more challenging numerics, the final case is the NACA 0004 airfoil in RANS flow with the version of the Spalart-Allmaras turbulence model outlined in Subsection 2.1.3. The angle of attack, in this case, is $\alpha = 3°$ and the Reynolds number is set to $Re = 1 \times 10^4$. A relatively low freestream Mach number of $M_\infty = 0.5$ is used so that the Mach number in the domain never approaches sonic conditions. The rationale for this is the concern that the present shock-capturing techniques, through artificial viscosity stabilization, can adversely affect the entropy variables. Additionally, the wall distance function interpolation order, which is the order of polynomials used to approximate the wall distance within each element [136], was set to 2. For curved geometries, this order needs to be sufficiently high to not introduce low-order errors in the wall distance calculation, which is required for the source terms of the turbulence model. Experiments with linear wall distance function approximation (in an element's reference space) showed degradation of the output convergence, whereas higher wall distance function orders did not appreciably affect the results for the solution orders used in this example.

Figure 6.17 shows the initial coarse mesh for this case, which consists of 660 triangular elements. Curved elements of order $q = 4$ represent the airfoil geometry, and the farfield boundary is 2000 chord lengths away. The cases were run with a solution approximation order of $p = 2$ with the following five degrees of freedom targets: 5000, 10000, 20000, 40000, and 80000. Just as with the previous MOESS simulations, for each `dof` target, fifteen solution iterations were performed before moving to the next `dof` target.

As before, the drag coefficient and lift coefficients were used as outputs to govern the mesh

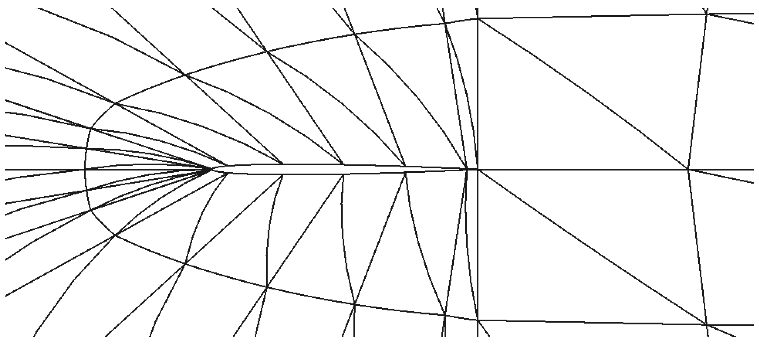


Figure 6.17: Initial NACA 0004 airfoil mesh.

adaptation. Adaptation runs using adjoints based on these outputs, the entropy-variable approach, and the standard combined entropy and output-based adjoint solution were generated. Two modified versions of the combined approach were also investigated for this case. The first is a modification to the combined approach in which only the output-based indicator is used to target which elements get refined while using MOESS. Therefore the output-based indicator alone governs the local sampling approach outlined in Subsection 5.4.2. The second modification involves using a modified residual made up of only the convective terms to obtain the entropy-based adjoint when using the combined approach.

The following reference [137] indicates that for RANS cases, the optimal drag coefficient error estimate using entropy variables requires two inviscid residual evaluations, one with the approximate solution and one with the exact solution. This is motivated by relating the output J from Equation 3.42 directly to one output of interest, the drag coefficient, through Oswatitsch's formula [138] via

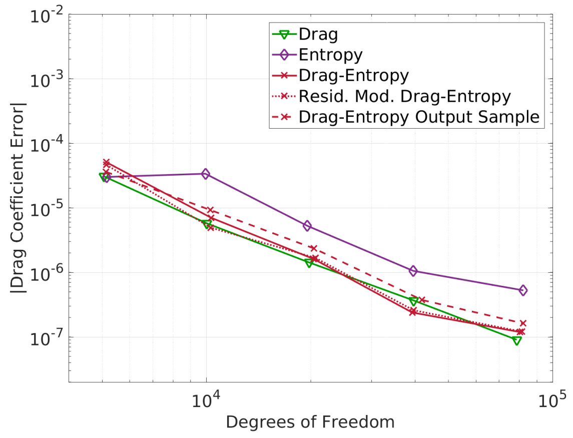
$$\delta c_{d,\mathrm{osw}} \approx K\delta J^{\mathrm{inv}} = K\left(J^{\mathrm{inv}}\left(\mathbf{u}_H\right) - J^{\mathrm{inv}}\left(\mathbf{u}\right)\right) = K\int_{\Omega} \mathbf{v}^T\left(\mathbf{r}^{\mathrm{inv}}\left(\mathbf{u}_H\right) - \mathbf{r}^{\mathrm{inv}}\left(\mathbf{u}\right)\right) d\Omega, \tag{6.1}$$

where the superscript "inv" denotes inviscid residuals and fluxes. For RANS cases, the term $\mathbf{r}^{\mathrm{inv}}\left(\mathbf{u}\right)$ needs to be approximated. The discrete analogue of Equation 6.1 is

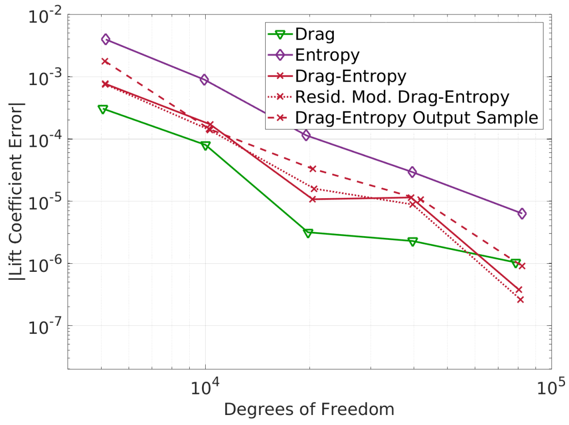
$$\delta c_{d,\mathrm{osw}} \approx K\mathbf{V}_h^T\left[\mathbf{R}_h^{\mathrm{inv}}\left(\mathbf{U}_h^H\right) - \mathbf{R}_h^{\mathrm{inv}}\left(\mathbf{U}_h\right)\right], \tag{6.2}$$

where $\mathbf{U}_h^H$ is the discrete coarse-space state vector prolonged into the fine space, and $\mathbf{U}_h$ is the discrete fine-space state. The residual operators in Equation 6.2 include only the convective terms of the RANS equations.

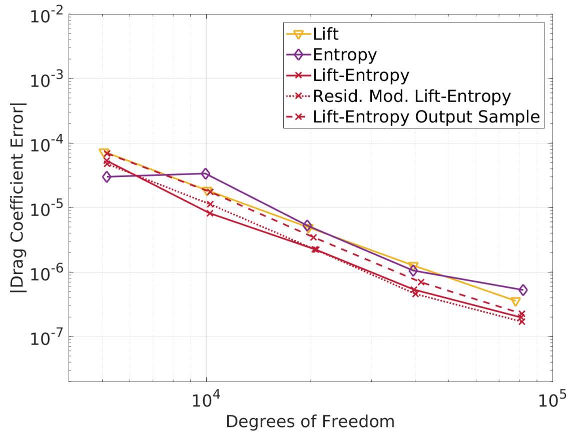
Figure 6.18 shows the convergence results for all of the error indicator strategies. In this figure, we also present how the drag-based indicators perform in predicting lift coefficient (and vice versa), which yields a total of four output error convergence plots. Again, the output and `dof` values reported from these results are averages over the last 5 solution iterations at each target `dof`. The truth outputs for both the drag and lift coefficients were obtained by uniformly refining the final output-adapted mesh and running at $p = 3$. An identical set of truth solutions was obtained for both truth solutions using $p = 4$. The difference in the outputs between these two solutions is of the order 1×10^{-9}. The dashed red line corresponds to the case where the output-based adjoint alone controls the local sampling. The dotted red line corresponds to the combined approach that incorporates the modified residual approach to obtain the entropy-based indicator.



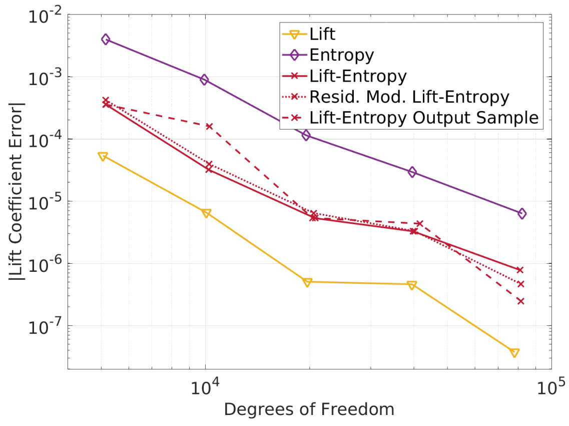
(a) Drag Output Using Drag-based Adjoints



(b) Lift Output Using Drag-based Adjoints



(c) Drag Output Using Lift-based Adjoints



(d) Lift Output Using Lift-based Adjoints

Figure 6.18: NACA 0004 $M_\infty = 0.5$, $\alpha = 3°$, $Re = 1 \times 10^4$: Comparison of output convergence histories for various error indicator strategies using DG.

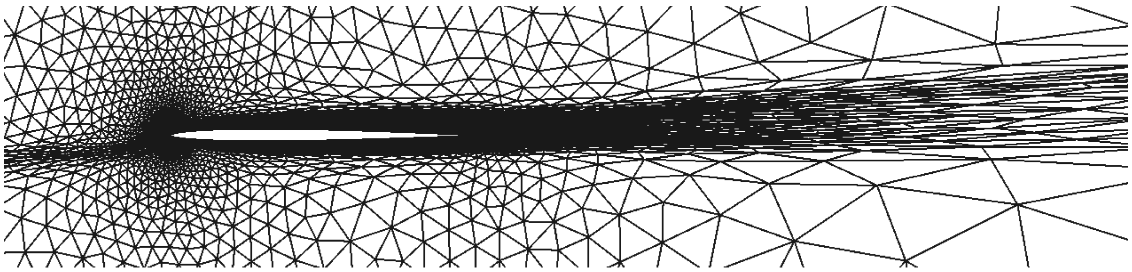
When it comes to predicting the drag coefficient, none of the combined approaches yield significantly different results compared to the drag-based approach. In fact, several other RANS simulations were analyzed with multiple geometries and flow conditions. It was found that the drag-based adjoint indicator performs similarly, or sometimes a little better than the combined approach indicators. However, this is not overly discouraging, since these are steady-state simulations. For unsteady simulations, the cost of fine output adjoint calculations will be dramatically larger, and we expect more benefit from using the much cheaper entropy variables.

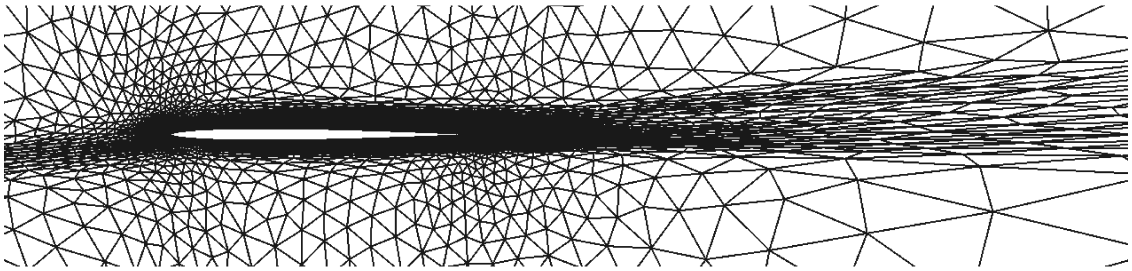
With regards to predicting lift coefficient, there is much more variety among the different adaptive indicator approaches. The lift-based indicator predicts a much more accurate lift coefficient compared to the combined approach. This is not surprising, given that we have observed the reliability of the entropy-based indicator with regards to its prediction of drag coefficient, since regions of the flow field that produce spurious entropy generation are those that subsequently produce more drag. On the other hand, the combined approach using the drag-based indicator obtains more accurate lift coefficients than the drag-based indicator for the final degrees of freedom target. Unfortunately, since the combined approaches do not produce as accurate lift coefficient errors at the lower degrees of freedom targets, it is difficult to definitively state that the combined approaches are better.

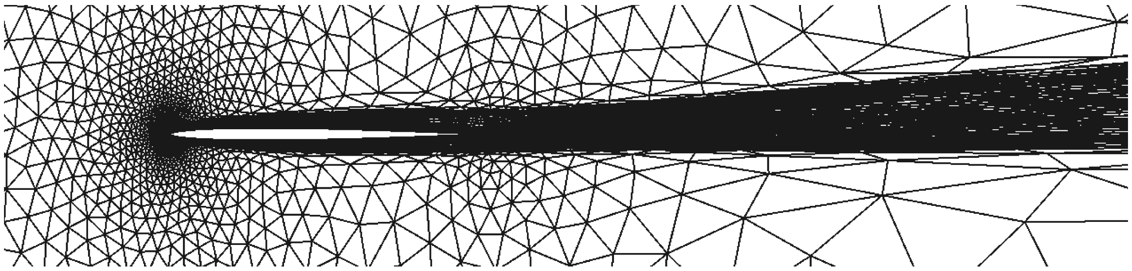
Figure 6.19 presents the meshes at 80000 `dof` for the output-based adjoint approaches, the entropy-based approach, and the combined approaches with no modifications. Although the drag coefficient errors at this `dof` are not that different among all of the adaptation indicator strategies, the meshes do show visible differences consistent with those in Figure 6.16. Clearly, the output-based adjoint approaches focus refinement ahead of the airfoil and less in the wake, while the entropy-based approach focuses significant refinement in the wake. Interestingly, the drag coefficient error levels specifically do not vary significantly among the different strategies, which suggests that either the set of meshes that yield nearly optimal drag predictions is large, or that an optimal mesh has not yet been identified by the adaptive approaches considered. The resolution of this question is an area of ongoing research.

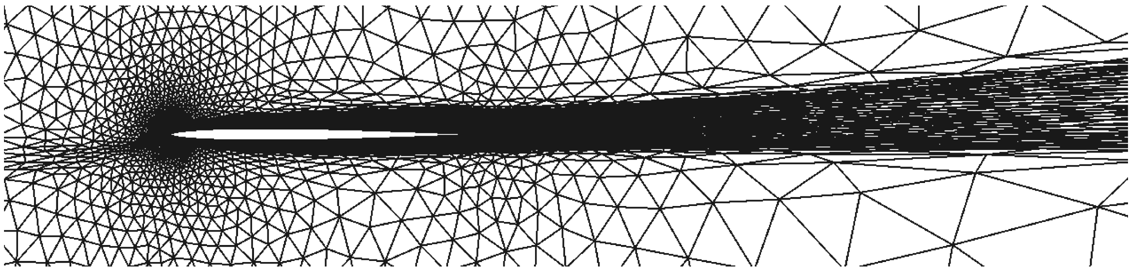
6.2 Steady-State Inviscid Cases Comparing SUPG to DG

The results in this section showcase how the combined approach performs for the SUPG discretization relative to DG discretization used for all results in Section 6.1. All cases in this section are again steady-state simulations. In addition, since the SUPG code written for this work is based on a discretization of the Euler equations reviewed in Subsection 2.1.2, the cases in this section are limited to those equations. In total there are two different cases presented in this work, each using various error indicators approaches, shown in Table 6.2. Many of the results in this section were previously published in the following work [107].


(a) Drag Adaptation


(b) Lift Adaptation


(c) Entropy Adaptation


(d) Drag-Entropy Adaptation

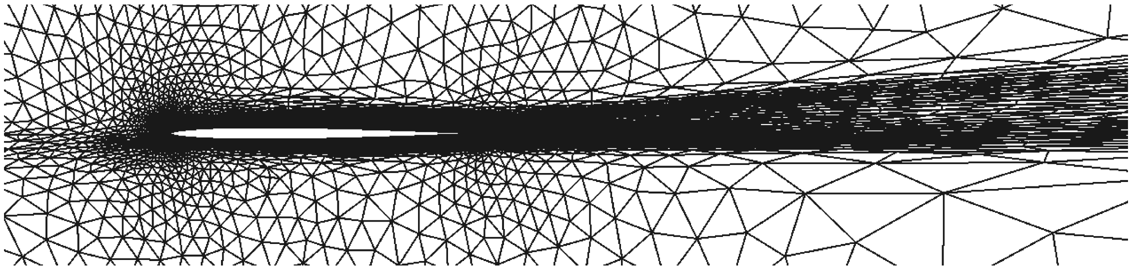

(e) Lift-Entropy Adaptation

Figure 6.19: NACA 0004 $M_\infty = 0.5$, $\alpha = 3°$, $Re = 1 \times 10^4$: Meshes after final adaptation iteration (approximately 80000 DOF) for various error indicator strategies using DG.

	NACA 0012 Airfoil (Inviscid)	**Diamond Airfoil (Inviscid)**
Output	X	X
Entropy	X	X
Combined	X	X
Comb. w/ Mask		X

Table 6.2: Error indicators used for each steady-state case comparing SUPG to DG.

6.2.1 NACA 0012 Airfoil: $M_\infty = 0.3,\ \alpha = 5°$

The first case in this section is a two-dimensional, inviscid flow over the NACA 0012 airfoil with a closed trailing edge. The initial mesh, shown in Figure 6.20, is made up of 533 triangular elements with a far-field located 100 chord lengths away from the airfoil. Curved elements, $q = 3$, were used to represent the geometry and solution approximation orders of $p = 1$ and $p = 2$ were used. The following degrees of freedom targets were run with $p = 1$: 750, 1500, 3000, and 6000. In addition, the following degrees of freedom targets were run with $p = 2$: 1500, 3000, 6000, and 12000. At each `dof` target, multiple solution iterations were performed so that an average of five solutions could be obtained.

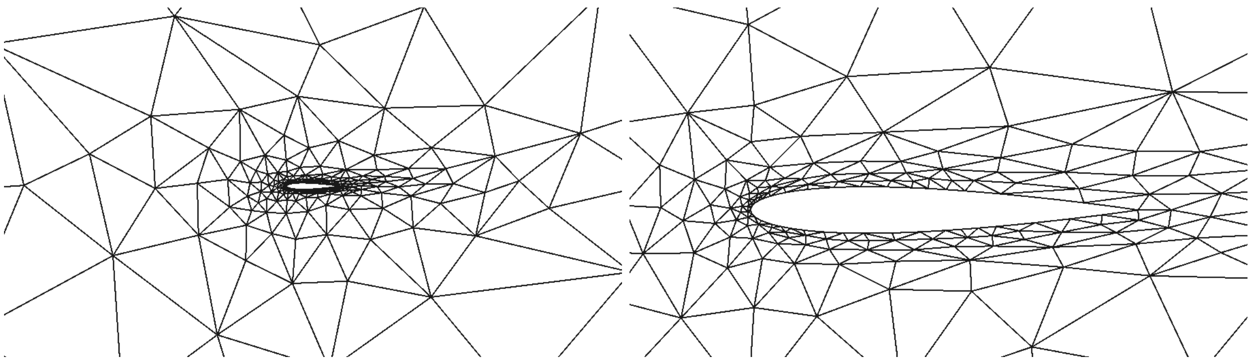

(a) Initial Mesh (b) Initial Mesh (Zoom)

Figure 6.20: Initial NACA 0012 airfoil mesh.

The freestream Mach number is $M_\infty = 0.3$ and the angle of attack is $\alpha = 5°$. The flow is subsonic everywhere in the computational domain. Mach number contours are shown in Figure 6.21. The drag coefficient and lift coefficient were chosen as the engineering outputs. Adaptation was governed by adjoints using either the output-based adjoint or the entropy variables. In addition, the standard combined approach that uses the fine-space output-based adjoint and entropy-variable approach was considered.

Figure 6.22 shows the convergence of the drag coefficient for the various adaptation strategies using both SUPG and DG. The mesh adaptation strategy used for these cases was MOESS. One

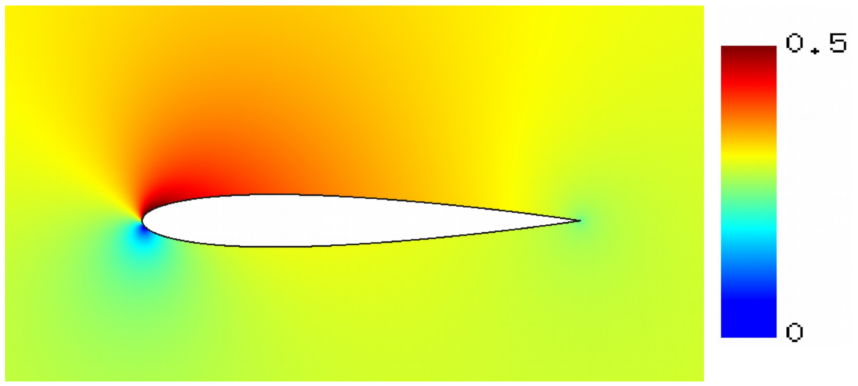
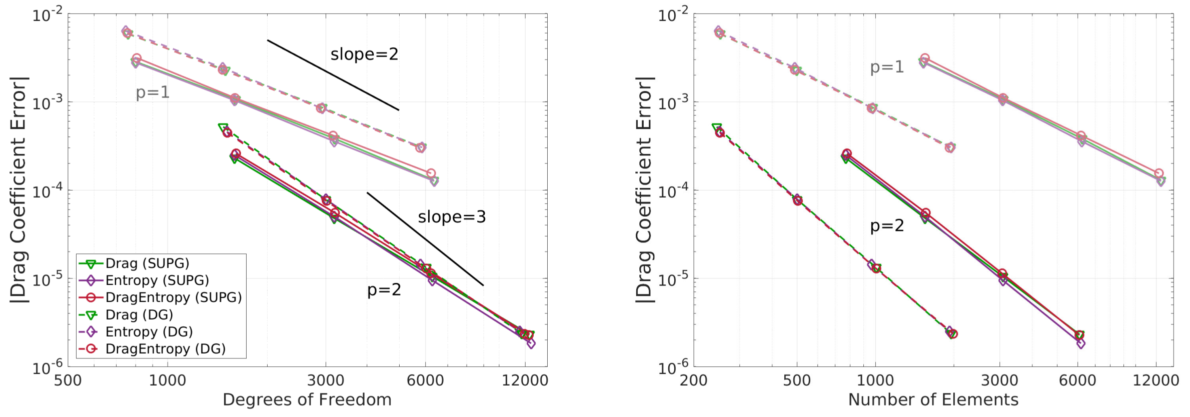


Figure 6.21: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Mach number contours (range: 0-0.5).

convergence plot compares the drag coefficient error versus degrees of freedom, while the other shows the error versus the number of elements. The drag truth solution was obtained by taking the finest mesh obtained using the drag-based adjoint with the DG discretization at $p = 2$, uniformly refining it twice, and solving it with a solution approximation order of $p = 3$. A second truth solution was obtained by taking the finest mesh obtained using the drag-based adjoint with the SUPG discretization and running with a solution approximation order of $p = 3$. The difference between these two drag coefficients was 3×10^{-8}. The observed convergence rate for both DG and SUPG at $p = 1$ is approximately 2, while at $p = 2$ the rate is 3. This indicates we are seeing a convergence rate of $p+1$ for these cases, and not the expected super-convergent rate of $2p+1$ [95]. When isolated singularities are present in the solution, adaptive refinement can focus refinement to such a degree in these regions that refinement of the smooth regions is not resolved at the optimal rate. For this case, a significant focus of the refinement is singularities at the stagnation point and trailing edge.

For the same number of degrees of freedom, cases using a solution approximation order of $p = 2$ produced smaller errors compared to cases that used $p = 1$. This conclusion holds regardless of whether SUPG or DG was used. At $p = 1$, there is a much greater difference in error levels between SUPG and DG than at $p = 2$. At each degrees of freedom target, SUPG produces drag coefficient error levels that are about one-third an order of magnitude less than DG. For $p = 2$, the relative difference in error between SUPG and DG is much less, especially for the higher degrees of freedom. Since each element run with DG has its own unique set of degrees of freedom, there are far fewer overall elements in DG compared to simulations using SUPG. At $p = 1$, there are three unique degrees of freedom for each element in DG. This leads to inferior performance relative to SUPG where the approximations are continuous and there are more elements in the mesh. For $p = 2$, the number of degrees of freedom in DG increases to six, which is enough to better compensate for the fact that the solution is discontinuous across elements. For the same



(a) Drag Error Versus Degrees of Freedom

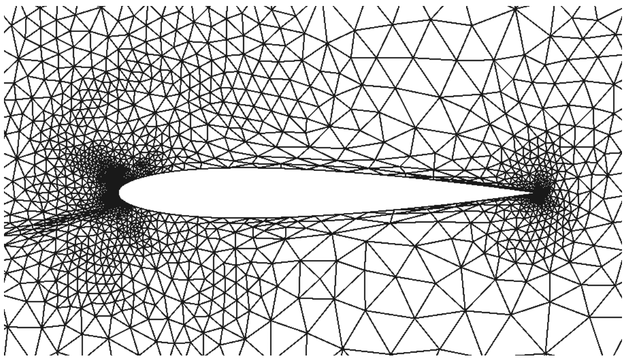
(b) Drag Error Versus Number of Elements

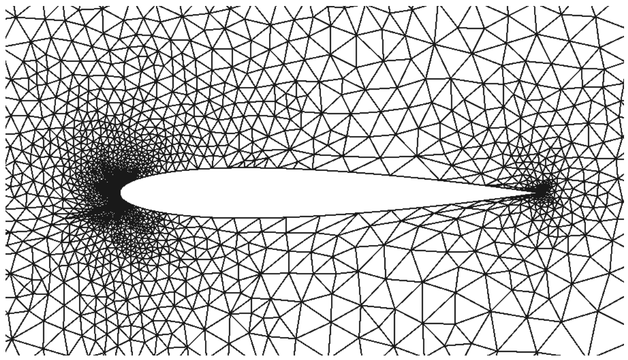
Figure 6.22: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Comparison of drag coefficient convergence histories for various error indicator strategies using SUPG and DG.

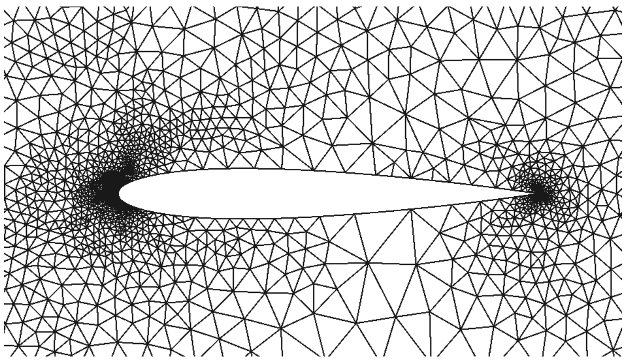
number of elements, DG performs much better than SUPG. However, that is not a fair comparison as the number of degrees of freedom per element is much less using SUPG, and it is the number of degrees of freedom and the consequent structure of the matrices (not the number of elements) that adds to the computational expense.

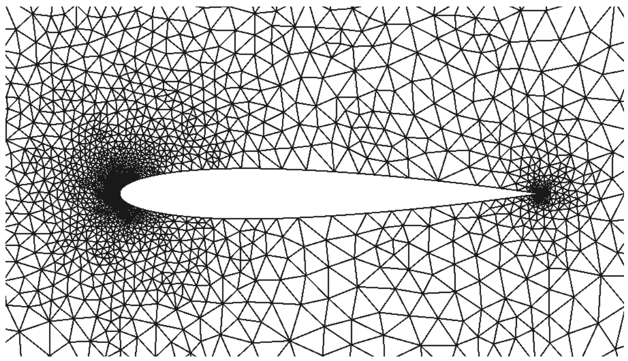
This case is a benign simulation with no major flow discontinuities. Consequently, there is not too much difference between the three different error estimation approaches, especially for DG. For SUPG, there is a small benefit in accuracy using just the entropy-adjoint compared to using just the drag-based adjoint for $p = 2$. This is a particular case that shows no loss in accuracy using entropy variables to drive adaptation compared to the typical output-based adjoint approach. However, as was demonstrated numerous times for the cases in Section 6.1, this is only because there are no regions in the domain that generate spurious entropy that are far from the airfoil. Since there is not a major difference in performance between using the drag-based adjoint versus entropy variables, the combined approach does not provide much of an impact. However, it does not negatively impact the performance.

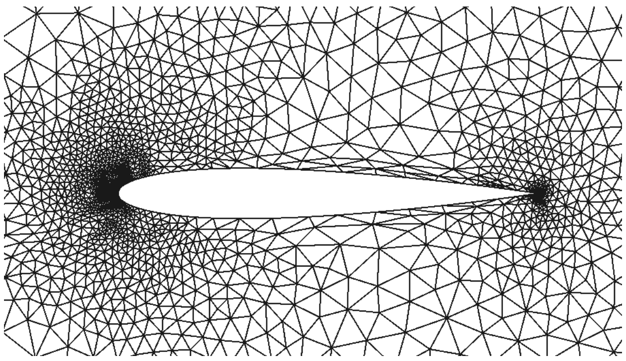
Sample meshes generated using SUPG with the various error indicator approaches for both $p = 2$ and $p = 1$ are shown in Figure 6.23. All of these meshes have approximately 6000 elements, but the meshes generated using $p = 2$ have far more degrees of freedom. Both meshes obtained using the drag-based adjoint allocate a large number of elements to the stagnation streamline region well in front of the airfoil, where the adjoint has a weak singularity [78]. However, at $p = 2$, significantly more elements are allocated to this region. This feature was present in meshes generated using the output-based adjoints for the NACA 0012 airfoil in viscous flow from the previous section, as shown in Figure 6.16. The meshes generated using the entropy variables or the combined approach do not demonstrate this propensity to allocate elements to this region. This explains why these approaches produce slightly lower error levels at $p = 2$. For meshes generated using $p = 2$, there is also more refinement near the surface of the airfoil in the middle region, except for meshes that use the entropy-variable approach. When using entropy variables, all refinement is focused on the nose and tip of the airfoil where it is more critical to accurately predict drag.


(a) Drag Adaptation ($p = 2$, Elem≈6000)


(b) Drag Adaptation ($p = 1$, Elem≈6000)


(c) Entropy Adaptation ($p = 2$, Elem≈6000)


(d) Entropy Adaptation ($p = 1$, Elem≈6000)


(e) Drag-Entropy Adaptation ($p = 2$, Elem≈6000)

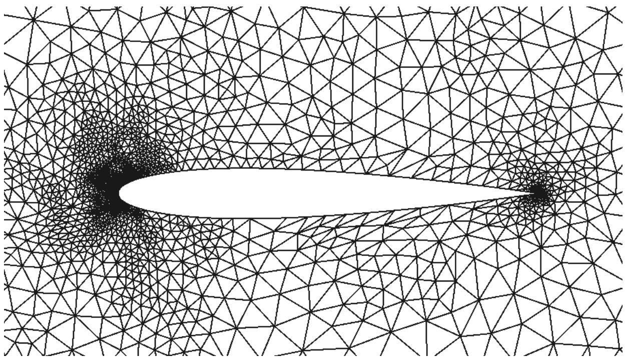

(f) Drag-Entropy Adaptation ($p = 1$, Elem≈6000)

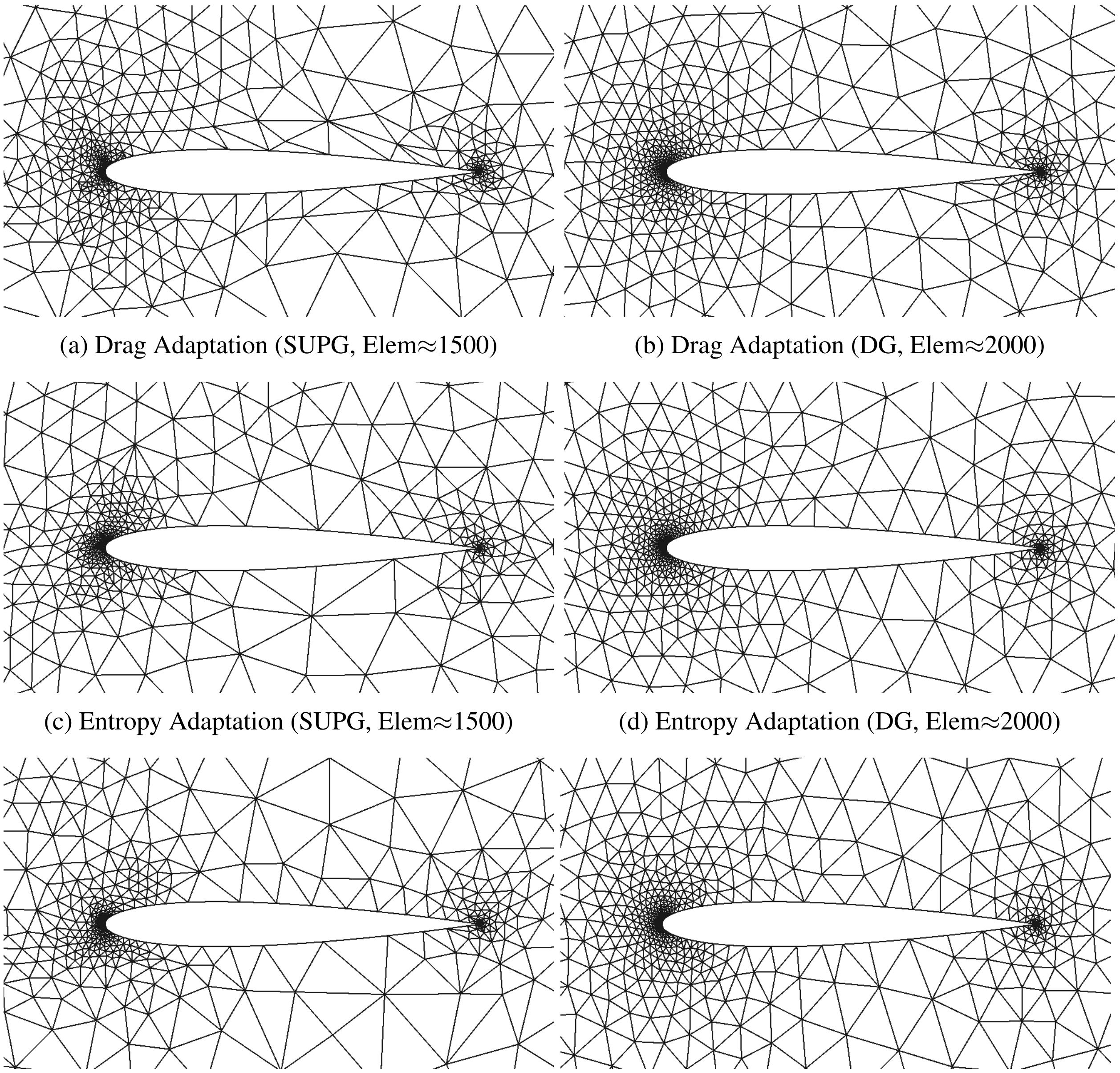
Figure 6.23: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Meshes generated using SUPG for various error indicators.

A comparison of meshes generated using SUPG and DG with the various error indicator approaches for $p = 2$ is shown in Figure 6.24. For SUPG, there are approximately 1500 elements in each mesh, while there are approximately 2000 elements in the DG meshes. Despite the scarcity in the number of elements in each mesh compared to the previous figure, these are the finest DG meshes shown in Figure 6.22, since the number of degrees of freedom per element is much higher compared to SUPG. Overall, the meshes are pretty comparable between DG and SUPG for a given error strategy. Since the meshes are much coarser from an element perspective, all of the extra refinement in the stagnation streamline region is not present for the drag-based adjoint cases. This explains where there was minimal difference between the entropy variables and output-based approach for DG. These meshes are not fine enough to warrant the unnecessary stagnation streamline refinement. A reasonable conclusion is that for this case the benefit of the entropy-variable approach is much stronger in SUPG compared to DG, since SUPG requires far more elements at the same degrees of freedom level. When there are more elements in a given mesh, they are more likely to be placed in unnecessary regions for the drag-based adjoint approach.

Figure 6.25 shows the convergence of the lift coefficient for the various adaptation strategies using both SUPG and DG. Instead of using the drag-based adjoint for the output-based adjoint and combined approaches, the lift-based adjoint is used instead. Finding the lift truth solution was done in the same manner as was done for the drag truth solution. The difference in truth solutions between SUPG and DG is 3×10^{-7}. The observed convergence rate for both DG and SUPG at $p = 1$ is slightly less than 2, while at $p = 2$ the rate is slightly less than 3. Unlike for the drag coefficient convergence plots in Figure 6.22, for lift coefficient we are not observing a convergence rate of $p + 1$. This is because the focus of the stagnation point is even greater for estimating the lift coefficient. This increased focus on refining the stagnation streamline region reduces the convergence rate for the remaining smooth regions in the domain.

When it comes to predicting lift coefficient, the relative difference in error between $p = 2$ and $p = 1$ is still significant for both SUPG and DG. However, there is much more variability in predicting lift coefficient among the different error indicator approaches compared to predicting the drag coefficient, where all of the different approaches were pretty similar. For $p = 1$, all of the approaches produce lower lift coefficient error levels when using SUPG compared to DG. The entropy-variable approach, in particular, outperforms all of the other approaches. However, for $p = 2$, the results obtained using DG are much more comparable, if not slightly better, compared to SUPG. For the higher order of $p = 2$, the entropy-variable approach suffers (especially for DG), while the combined approach generally produces lower error levels for both SUPG and DG. Since there is more variability between the output-based and entropy-variable approaches when calculating lift coefficient, the combined approach makes much more of a positive impact.

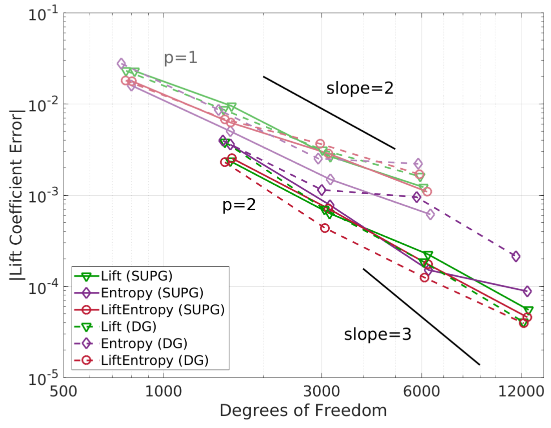
The benefits of the combined approach are much more obvious upon closer examination of the


(a) Drag Adaptation (SUPG, Elem≈1500) (b) Drag Adaptation (DG, Elem≈2000)

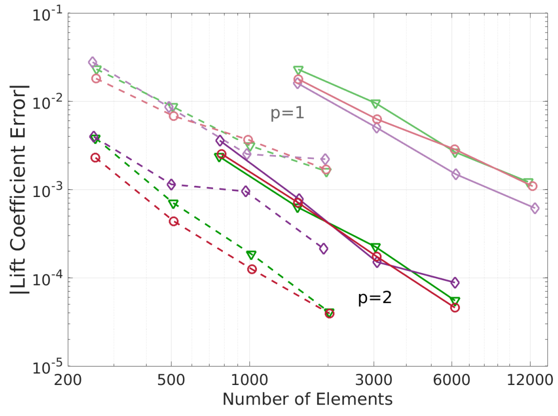
(c) Entropy Adaptation (SUPG, Elem≈1500) (d) Entropy Adaptation (DG, Elem≈2000)

(e) Drag-Entropy Adaptation (SUPG, Elem≈1500) (f) Drag-Entropy Adaptation (DG, Elem≈2000)

Figure 6.24: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Meshes generated using SUPG and DG for various error indicators at $p = 2$.



(a) Lift Error Versus Degrees of Freedom

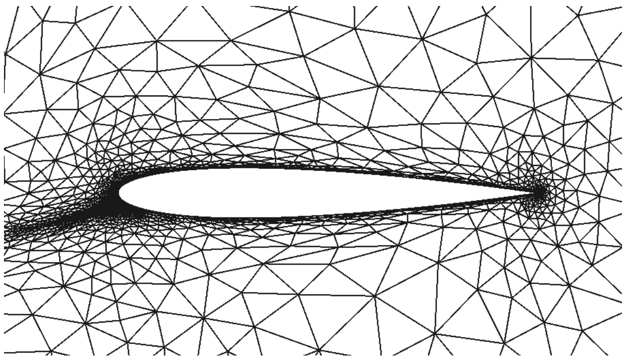


(b) Lift Error Versus Number of Elements

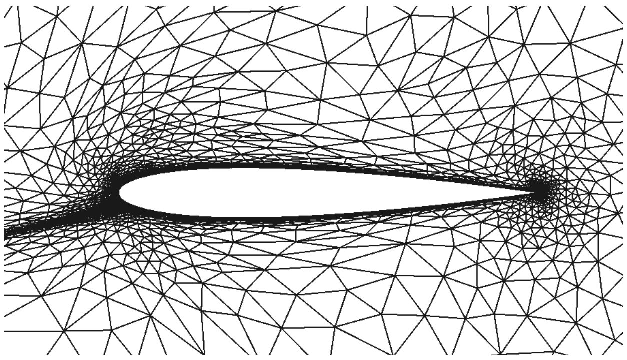
Figure 6.25: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Comparison of lift coefficient convergence histories for various error indicator strategies using SUPG and DG.

meshes. A comparison of meshes for both $p = 2$ and $p = 1$ generated using SUPG is shown in Figure 6.26. The meshes generated using the lift-based adjoints refine the stagnation streamline area to an even greater degree compared to the meshes generated using the drag-based adjoints. The entropy-variable approach does not refine this region, nor does it refine near the surface of the airfoil to the same degree as the output-based adjoint approaches do. This was not an issue for calculating the drag coefficient, but this may be more of an issue for calculating the lift coefficient. The entropy-variable approach in particular lacks refinement near the surface close to the tip and nose. The combined approach may strike a better balance between refining near the surface of the airfoil, near the nose and trailing edge especially, while not refining the stagnation streamline as much as the output-based approach. The combined approach at $p = 2$ does not refine the stagnation streamline region far from the airfoil as much as $p = 1$, which explains why it performs better for $p = 2$.

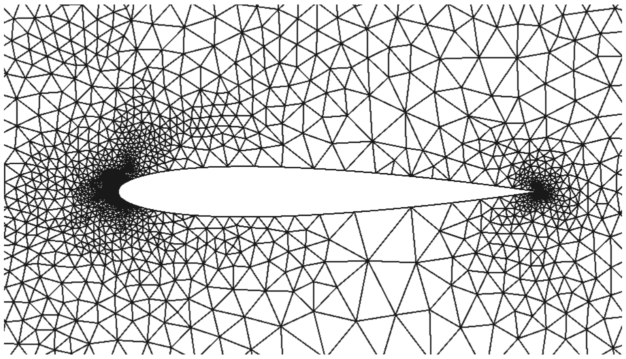
Meshes at $p = 2$ generated using DG and SUPG are directly compared in Figure 6.27. Just as in Figure 6.24, for SUPG there are approximately 1500 elements in each mesh, while there are approximately 2000 elements in the DG meshes. As before, the meshes between DG and SUPG are pretty comparable for the same error indicator approach. Unlike in Figure 6.24, there are much more noticeable differences in the meshes among the different error indicator approaches despite the meshes being coarser than in Figure 6.26. The stagnation streamline region and regions near the surface of the airfoil are significantly more refined in the output-based approach compared to the entropy-variable approach. While the extra refinement in the stagnation streamline region may not be necessary, the entropy-variable approach is not refining the regions near the surface of the



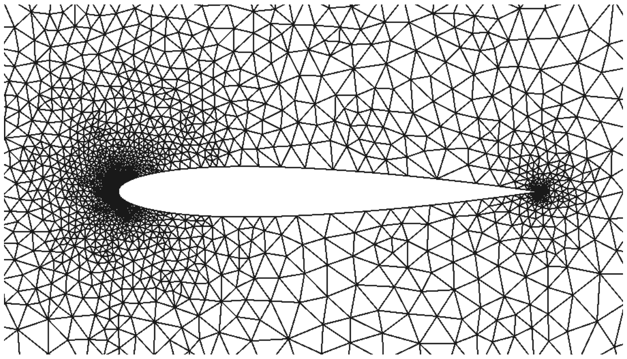
(a) Lift Adaptation ($p = 2$, Elem≈6000)



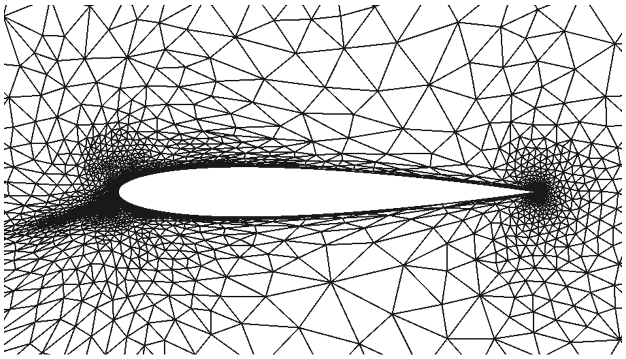
(b) Lift Adaptation ($p = 1$, Elem≈6000)



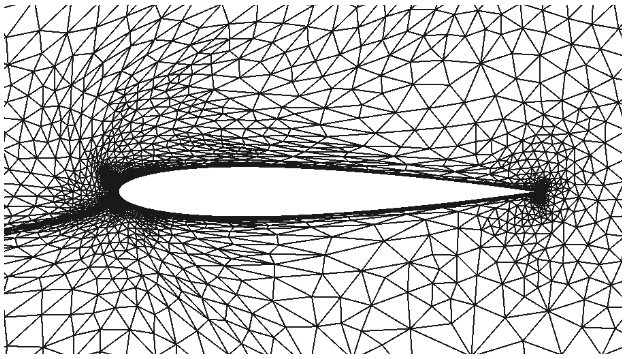
(c) Entropy Adaptation ($p = 2$, Elem≈6000)



(d) Entropy Adaptation ($p = 1$, Elem≈6000)



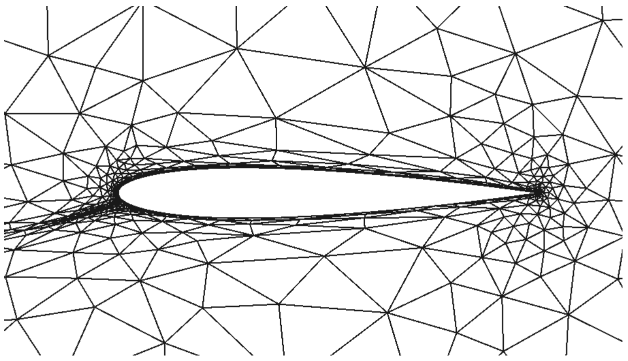
(e) Lift-Entropy Adaptation ($p = 2$, Elem≈6000)

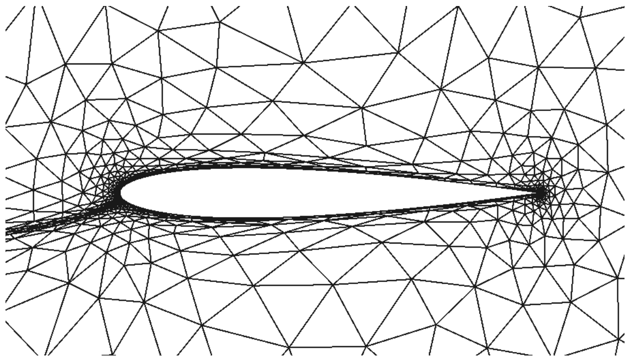


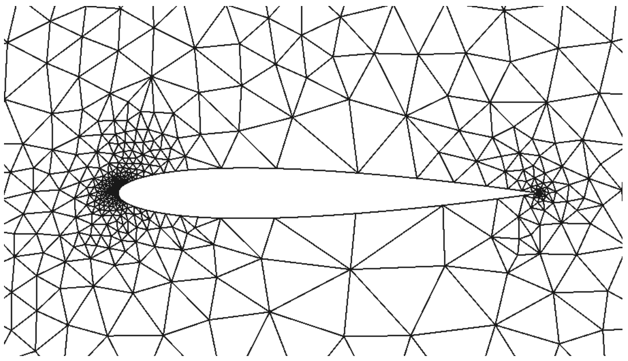
(f) Lift-Entropy Adaptation ($p = 1$, Elem≈6000)

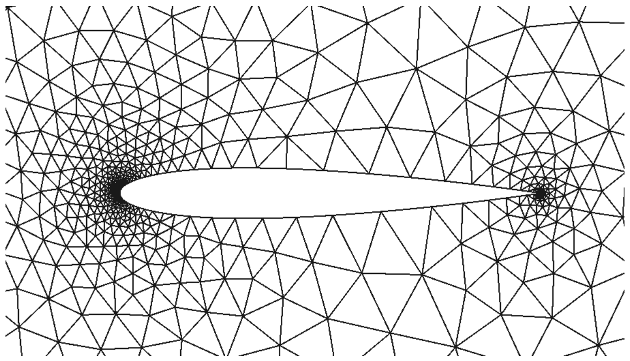
Figure 6.26: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Meshes generated using SUPG for various error indicators.

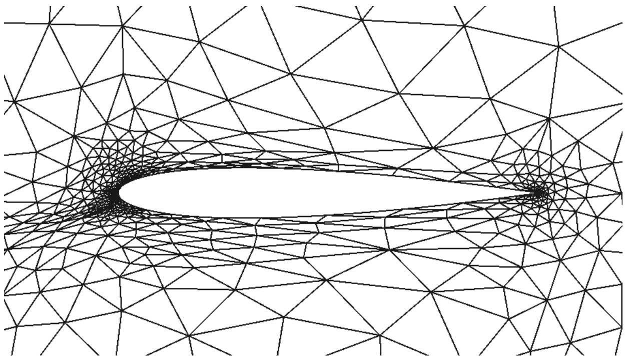
airfoil enough. This is why the combined approach works so well for calculating lift coefficient, as both the output-based approach and entropy-variable approach have limitations in refining the appropriate regions for calculating lift.


(a) Lift Adaptation (SUPG, Elem≈1500)


(b) Lift Adaptation (DG, Elem≈2000)


(c) Entropy Adaptation (SUPG, Elem≈1500)


(d) Entropy Adaptation (DG, Elem≈2000)


(e) Lift-Entropy Adaptation (SUPG, Elem≈1500)

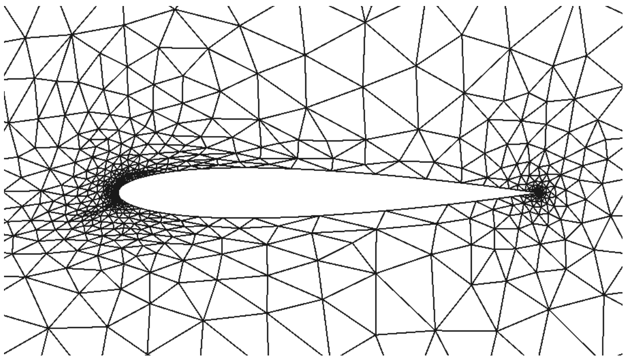
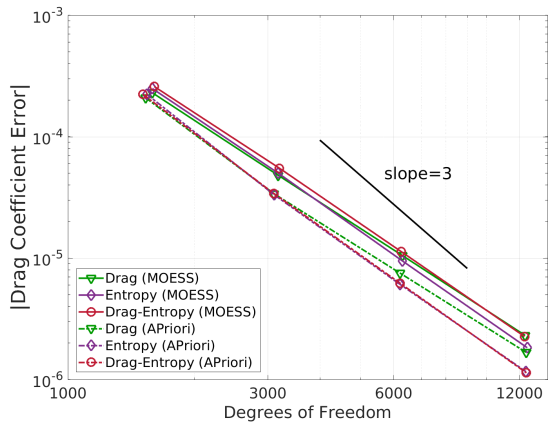

(f) Lift-Entropy Adaptation (DG, Elem≈2000)

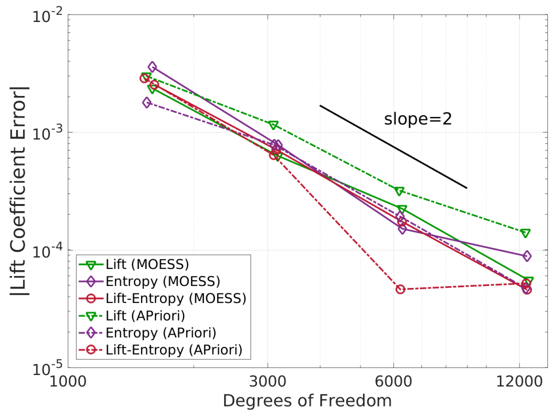
Figure 6.27: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Meshes generated using SUPG and DG for various error indicators at $p = 2$.

All of the results documented up to this point use MOESS for the mesh adaptation strategy. In Figure 6.28, the SUPG results at $p = 2$ from the previous figures are compared to similar simulations that use the *a priori* mesh adaptation approach from Section 5.3 instead of MOESS.

Overall, the *a priori* approach yields lower drag coefficient error levels compared to MOESS, regardless of the error indicator approach. However, the lift coefficient error performance between MOESS and *a priori* is dependent on chosen error indicator approach. The combined approach and the entropy-variable approach yield similar performance between MOESS and *a priori*, whereas the lift-based adjoint approach performs much better when using MOESS for mesh adaptation.



(a) Drag Error Versus Degrees of Freedom

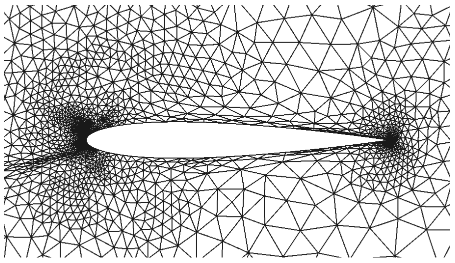


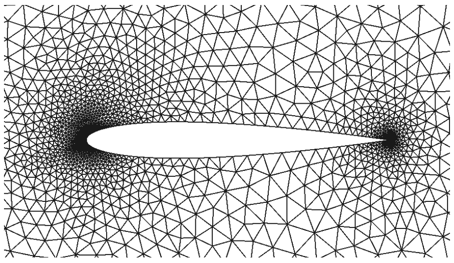
(b) Lift Error Versus Degrees of Freedom

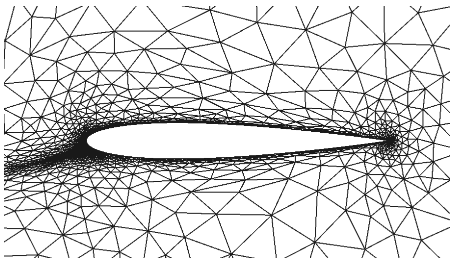
Figure 6.28: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Comparison of drag and lift coefficient convergence histories for various mesh adaptation strategies using SUPG at $p = 2$.

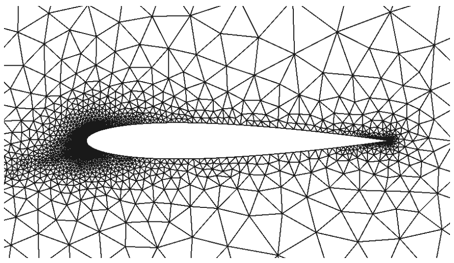
Figure 6.29 presents sample meshes from the various approaches shown in Figure 6.28. For both the drag-based and lift-based adjoint approaches, MOESS focuses more refinement in the stagnation streamline region and near the surface of the airfoil. This is because MOESS detects anisotropic refinement much better than *a priori*, as previously discussed in Chapter 5. Having refinement near the surface of the airfoil in the middle section of the airfoil is not as essential for accurately predicting the drag coefficient, which is why the *a priori* approach outperforms MOESS. Conversely, the extra refinement in the mid-board section of the airfoil leads to a better estimation of the lift, since having a good estimation of the pressure on both the bottom and top surface of the airfoil is critical for calculating the lift coefficient. It is obvious (based on the sample mesh generated with the lift-based adjoint approach) that the *a priori* approach is unable to produce ideal anisotropic elements on the boundary of the airfoil. This leads to suboptimal performance compared to MOESS. A reasonable conclusion is that if the desired output requires highly anisotropic elements, using MOESS is a more desirable approach compared to the *a priori* approach.

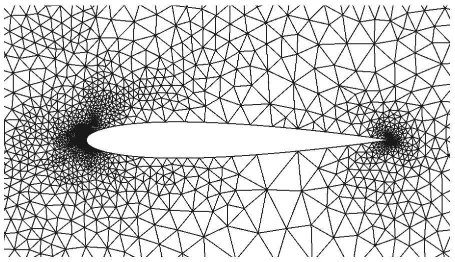
The same conclusions regarding the relative performance between MOESS and *a priori* for the output-based adjoint approach also apply to the combined approaches. However, the contributions

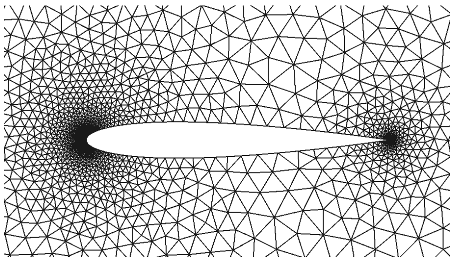

(a) Drag Adaptation (MOESS, Elem≈6000)

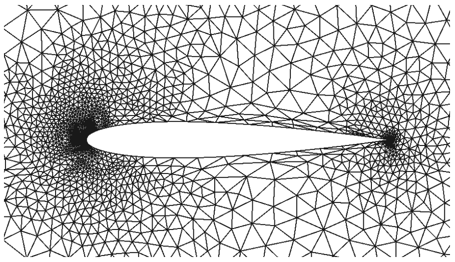

(b) Drag Adaptation (*A Priori*, Elem≈6000)

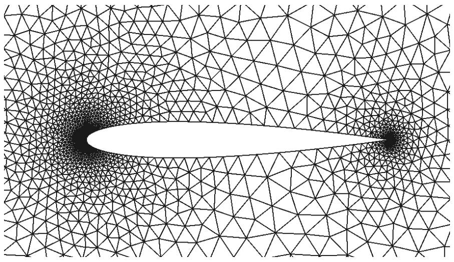

(c) Lift Adaptation (MOESS, Elem≈6000)

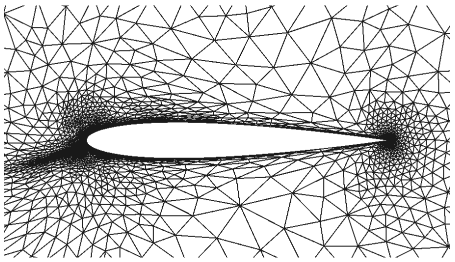

(d) Lift Adaptation (*A Priori*, Elem≈6000)


(e) Entropy Adaptation (MOESS, Elem≈6000)


(f) Entropy Adaptation (*A Priori*, Elem≈6000)


(g) Drag-Entropy Adaptation (MOESS, Elem≈6000)


(h) Drag-Entropy Adaptation (*A Priori*, Elem≈6000)


(i) Lift-Entropy Adaptation (MOESS, Elem≈6000)

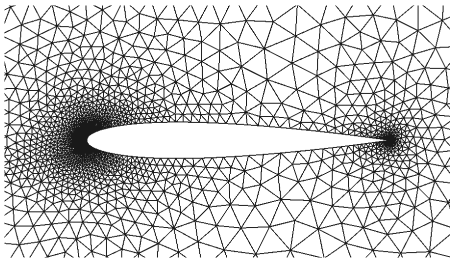

(j) Lift-Entropy Adaptation (*A Priori*, Elem≈6000)

Figure 6.29: NACA 0012 $M_\infty = 0.3$, $\alpha = 5°$: Comparing mesh adaptation strategies using SUPG at $p = 2$.

from the entropy variable indicator in the combined approach limit the dramatic differences present using the output-based approach. While the stagnation streamline region is still targeted for refinement in both combined approaches using MOESS, the refinement is not as fine as it was for the output-based approaches. This is why for estimating lift coefficient in particular, the combined approaches outperform the output-based approaches in both MOESS and *a priori*. The entropy-variable approach using *a priori* produces lower drag and lift errors compared to MOESS because the mesh is much coarser near the surface of the airfoil. While the output-based approaches in MOESS produced too much refinement in this region, the entropy-variable approach does not refine this region enough. Since the entropy variables and output-based adjoint approaches target different regions of the domain for both MOESS and *a priori*, there can be a clear benefit of using the combined approach so that all regions of the domain required to accurately calculate a particular output are adequately refined.

6.2.2 Diamond Airfoil: $M_\infty = 1.5,\ \alpha = 2^\circ$

The next case was chosen specifically because there are multiple flow discontinuities present in the computational domain. In this case, a thin diamond airfoil, with a thickness-to-chord ratio of 0.05, is subjected to a supersonic flow of $M_\infty = 1.5$ at an angle of attack of $\alpha = 2^\circ$. The initial conditions and airfoil geometry are identical to the case described in Subsection 6.1.3. Even though the flow is supersonic, the simulation can be solved without stability viscosity because the shocks are weak. The weak shocks are due to the small thickness of the diamond airfoil.

Just as in Subsection 6.1.3, the grid is a square with a side length of ten chords. The diamond is placed slightly off from the center of the domain so that the shocks emanating from the airfoil do not directly impact the corners of the domain. For this case, the initial linear, $q = 1$, mesh is made up of unstructured, triangular elements, as shown in Figure 6.30. This differs from the initial mesh in Figure 6.9 where the elements were made up of quadrilaterals.

Cases run with a solution interpolation order of $p = 1$ used the following three degrees of freedom targets: 375, 750, and 1500. Cases that had a solution interpolation order of $p = 2$ used the following degrees of freedom targets: 750, 1500, and 3000. As for the previous case in Subsection 6.2.1, multiple solution iterations were performed at each target so that an average could be done over five separate iterations. The drag coefficient was chosen as the analyzed engineering output for this case. Consequently, the drag-based adjoint approach was analyzed along with the entropy-variable approach and the standard combined approach that uses the fine-space drag-based adjoint.

Figure 6.31 presents the drag coefficient error convergence for the adaptation strategies using both DG and SUPG. The mesh adaptation strategy used for these cases was MOESS. The truth

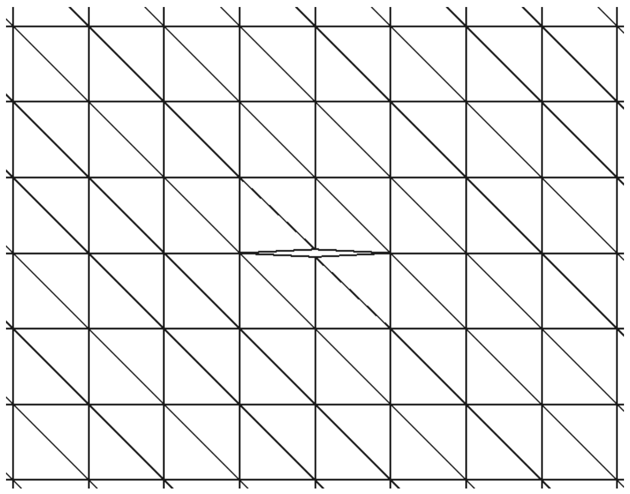
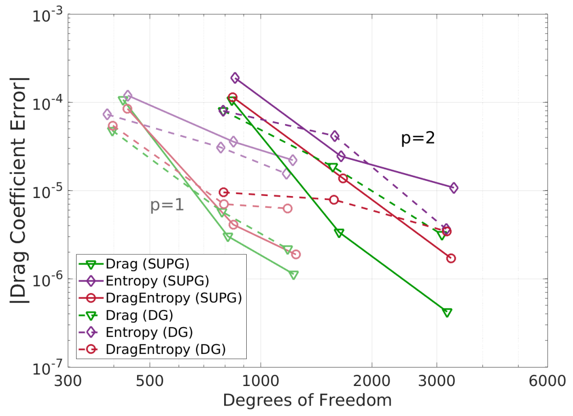
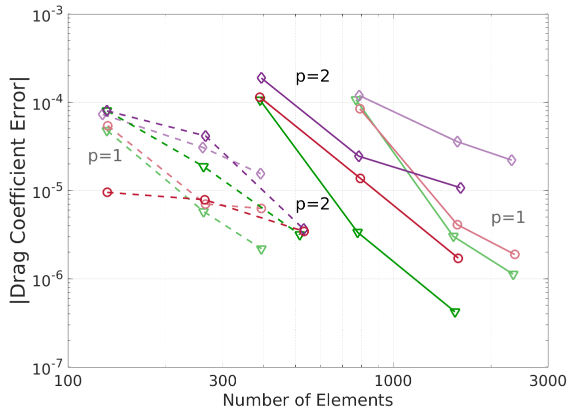


Figure 6.30: Initial diamond airfoil mesh made up of triangular elements.

solution was obtained by generating an adaptive mesh with approximately 3000 degrees of freedom using the drag-based adjoint with a DG formulation, uniformly refining that mesh, and generating a solution at $p = 3$. A similar truth solution was obtained using SUPG. The difference in the lift coefficient truth solutions between SUPG and DG was 2×10^{-7}.



(a) Drag Error Versus Degrees of Freedom



(b) Drag Error Versus Number of Elements

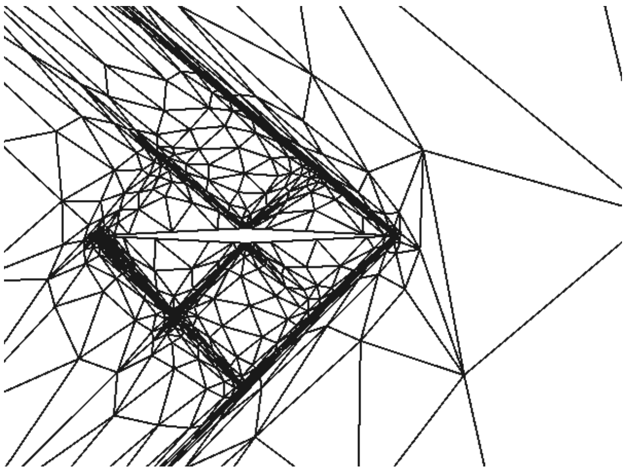
Figure 6.31: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Comparison of drag coefficient convergence histories for various error indicator strategies using SUPG and DG.

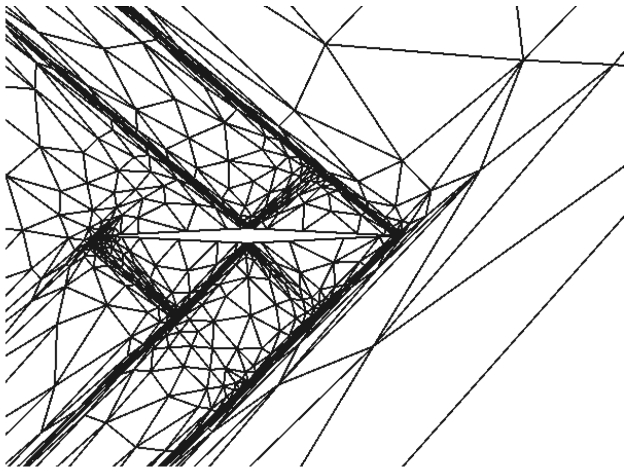
A noteworthy observation from Figure 6.31 is that generally there is not much of a benefit running this simulation at $p = 2$. Unlike in subsonic airfoil case in Subsection 6.2.1 where simulations at $p = 2$ always produced lower output errors at the same degree of freedom level, regardless of the discretization and adaptation approach used, for this case it is more advantageous to allocate degrees of freedom to the number of elements and not to higher elemental orders. Capturing the flow discontinuities with more elements produces better drag error estimates than using fewer higher-order elements. For the same number of elements, using $p = 2$ does provide a benefit for SUPG. However, that benefit is not as significant as it was for the previous cases.

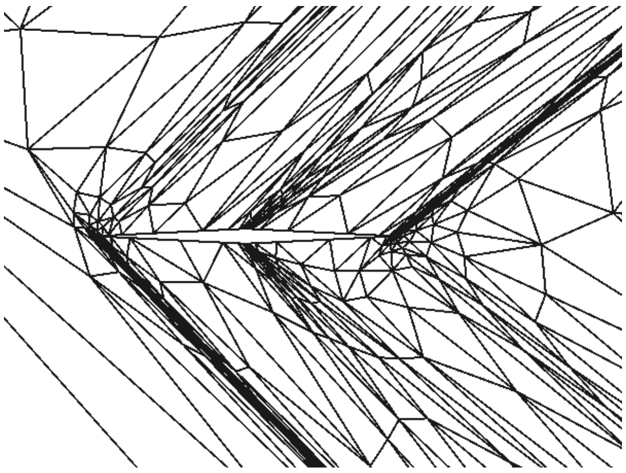
Another observation, consistent with the results from Subsection 6.1.3, is that for this case the entropy-variable approach is inferior to the output-based adjoint approach, regardless of whether SUPG or DG is used. This is because the entropy-variable approach targets only the shocks where a large amount of entropy is generated. Since the entropy-variable approach performs poorly, it negatively impacts the combined approach. This is a situation where a more sophisticated way of combining the indicators, weighted more towards those generated from the output-based adjoint, would likely improve the error estimation. In terms of comparing SUPG versus DG, using the combined approach and the drag-based adjoint approach, respectively, in SUPG produces lower drag error estimates compared to DG. Unlike in the previous airfoil case at subsonic flow, there is much more variation in drag error levels between SUPG and DG. If we consider our previous observation that for this case the number of elements is more essential than the number of degrees of freedom for estimating drag coefficient because outside of the shocks the domain is pretty uniform, it is reasonable to surmise that SUPG performs better than DG for this case. This is because for the same number of degrees of freedom, SUPG uses far more elements.

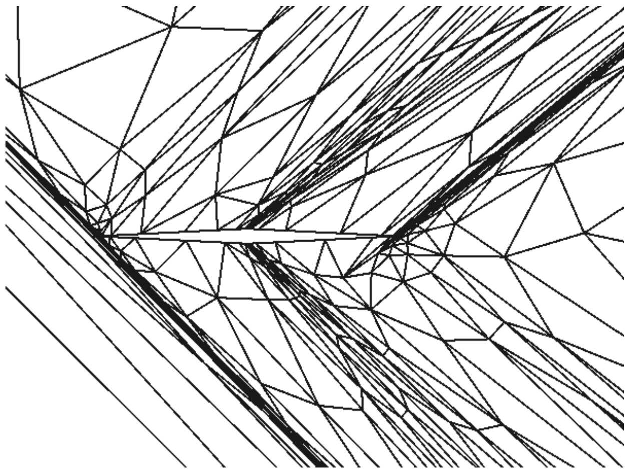
Sample meshes of around 1500 elements generated using SUPG at $p = 2$ and $p = 1$ are shown in Figure 6.32. For the entropy-variable approach, the refinement is targeted in the regions consumed by the discontinuities emanating from the nose and tip of the airfoil. This refinement extends unnecessarily to the far-field boundary because this approach has no way of discriminating regions of the domain that do not have much of an effect on the output of interest. This accounts for why the entropy-variable approach produces inferior results relative to the output-based approach. The output-based approach restricts the refinement to regions near the airfoil in the shape of a diamond. Whether all of the refinement that travels upstream from the airfoil is necessary to produce a good drag estimate is a worthwhile question, since the meshes generated from the combined approach do not show this feature. However, since the combined approach views refinement considerations from the output-based and entropy-variable approach equally, there is still a lot of unnecessary refinement far from the airfoil present in the combined approach. This leads to inferior performance compared to the output-based approach. This further suggests that an alternative way of combining the indicators to favor the output-based approach is likely necessary to improve performance.

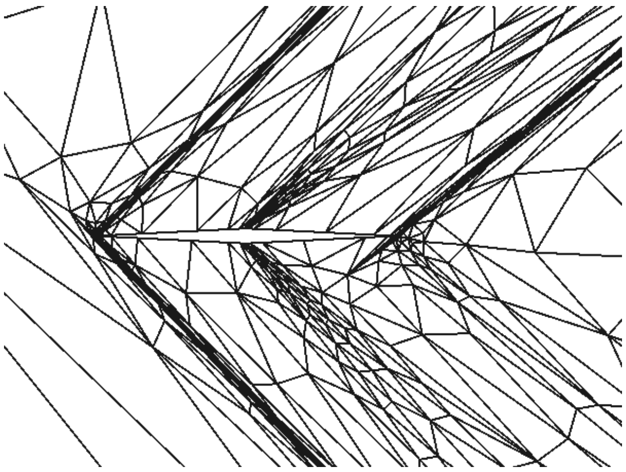
The effect the mesh adaptation strategy has on the various adaptive indicator approaches is shown in Figure 6.33. In this figure, drag coefficient error comparisons between cases that use MOESS versus cases that use the *a priori* mesh adaptation approach are shown. All of the cases have a solution approximation order of $p = 2$. The most important takeaway from this comparison is how much better the output-based approach performs with MOESS compared to *a priori*


(a) Drag Adaptation ($p = 1$, Elem≈1500)


(b) Drag Adaptation ($p = 1$, Elem≈1500)


(c) Entropy Adaptation ($p = 2$, Elem≈1500)


(d) Entropy Adaptation ($p = 1$, Elem≈1500)


(e) Drag-Entropy Adaptation ($p = 2$, Elem≈1500)

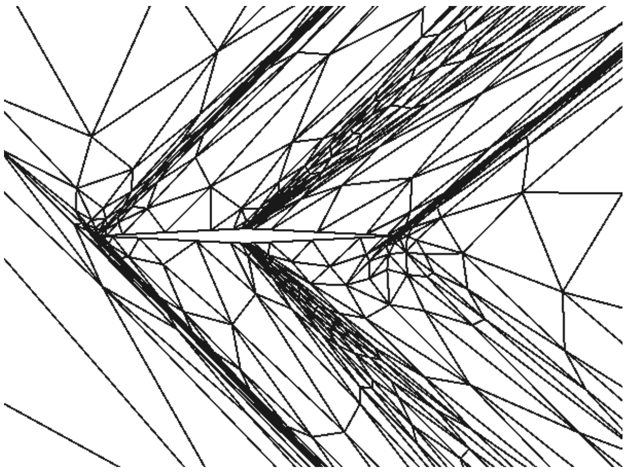

(f) Drag-Entropy Adaptation ($p = 1$, Elem≈1500)

Figure 6.32: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Meshes generated using SUPG for various error indicators.

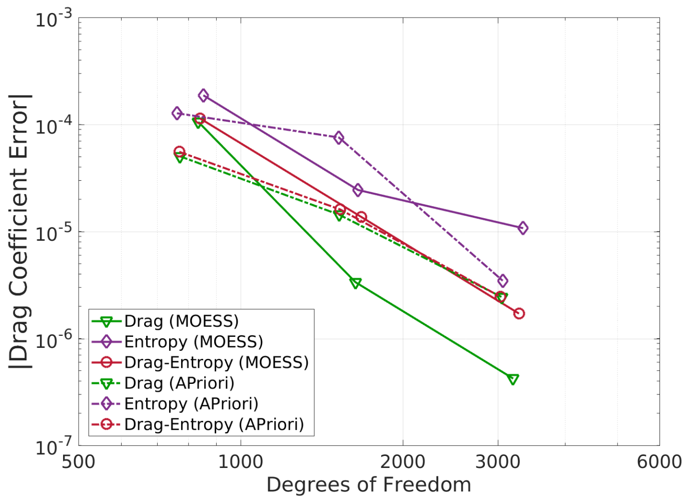
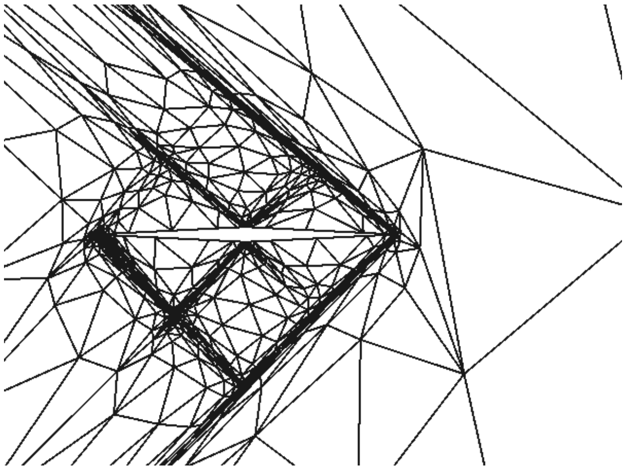


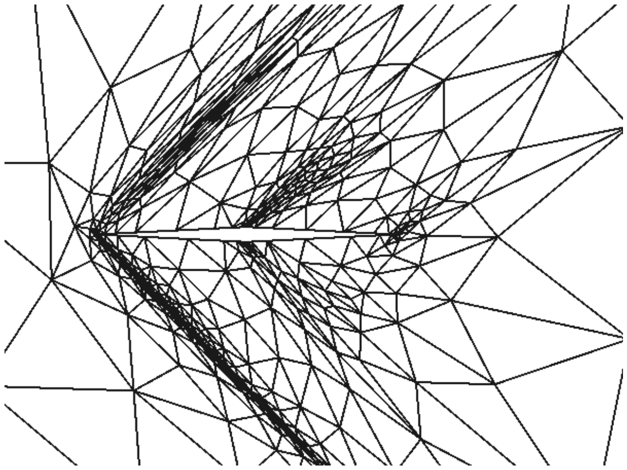
Figure 6.33: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Comparison of drag coefficient convergence histories for various mesh adaptation strategies using SUPG at $p = 2$.

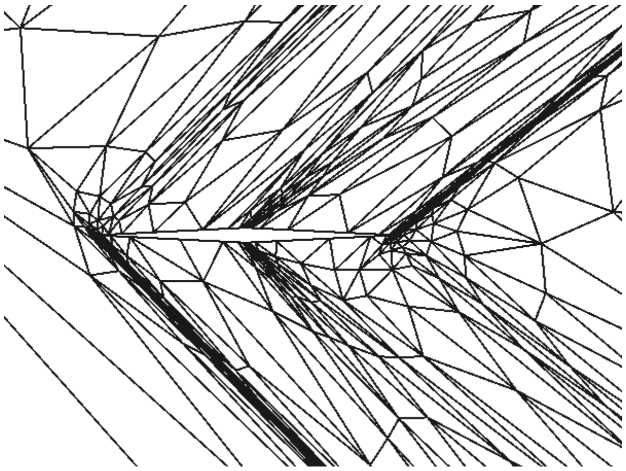
adaptation. In the subsonic airfoil case, highly anisotropic refinement was not necessary to predict an accurate drag coefficient. In this case, because of the flow discontinuities in the domain that must be captured, anisotropic refinement is much more beneficial and leads to a better drag error estimate for a given cost. The combined approach and entropy-variable approach do not show as significant of a difference between MOESS and *a priori* since both are degraded by all of the unnecessary refinement of the discontinuous regions of the flow domain far from the airfoil.

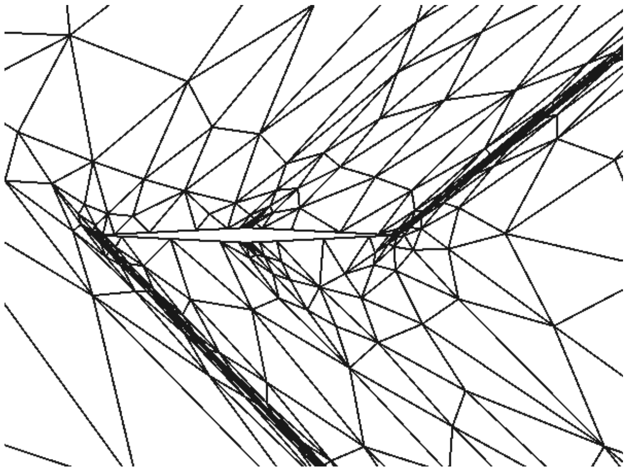
An examination of sample meshes made up of approximately 1500 elements in Figure 6.34 better illustrates the benefit of MOESS over *a priori* for this case. While the differences between the meshes for the entropy variables and the combined approaches are subtle, there is a significant difference between the meshes for the output-based adjoint approach. Using the *a priori* approach no longer produces the diamond pattern of mesh refinement shown with MOESS. Instead, the *a priori* approach focuses most of its refinement on the discontinuity emanating from the nose. Without the ability to capture the discontinuity with highly anisotropic elements, the *a priori* approach has to refine this region with far more elements. This consequently leads to less refinement in other regions of the domain, which negatively impacts the ability to produce an accurate drag error estimate.

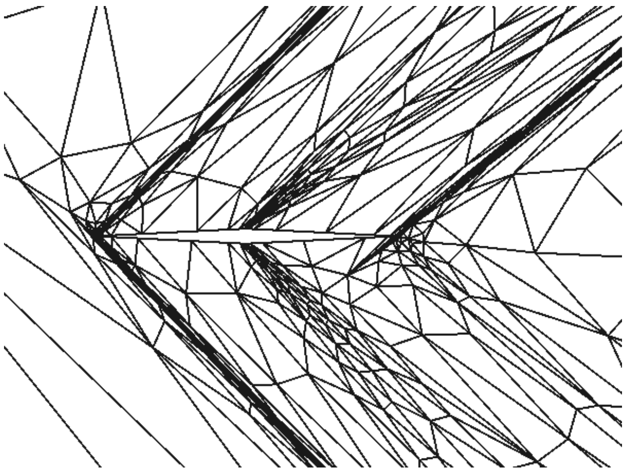
For all of the results shown thus far for this case, the output-based adjoint approach has generally produced lower drag coefficient errors compared to the entropy-variable approach and standard combined approach that uses the fine-space adjoint, regardless of the mesh adaptation strategy used. Error indicators generated from the entropy variables focus far too much on the shocks present in the domain, regardless of how far from the airfoil they are. This negatively impacts


(a) Drag Adaptation (MOESS, Elem≈1500)


(b) Drag Adaptation (*A Priori*, Elem≈1500)


(c) Entropy Adaptation (MOESS, Elem≈1500)


(d) Entropy Adaptation (*A Priori*, Elem≈1500)


(e) Drag-Entropy Adaptation (MOESS, Elem≈1500)

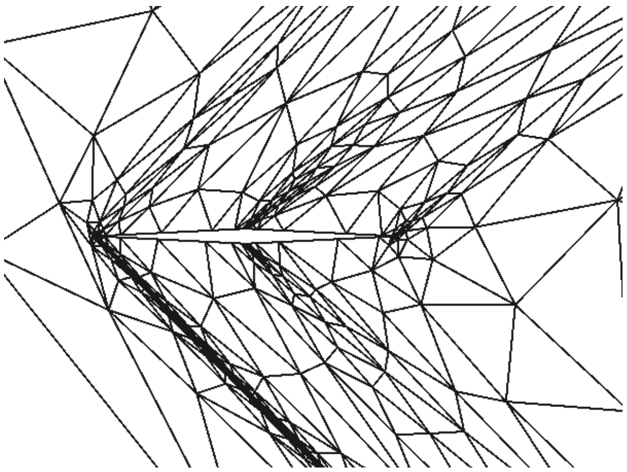
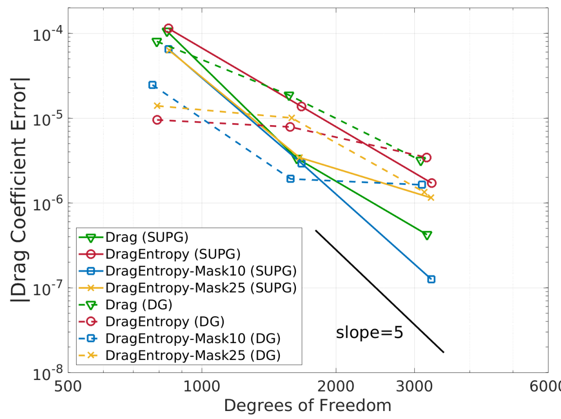

(f) Drag-Entropy Adaptation (*A Priori*, Elem≈1500)

Figure 6.34: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Comparing mesh adaptation strategies using SUPG at $p = 2$.

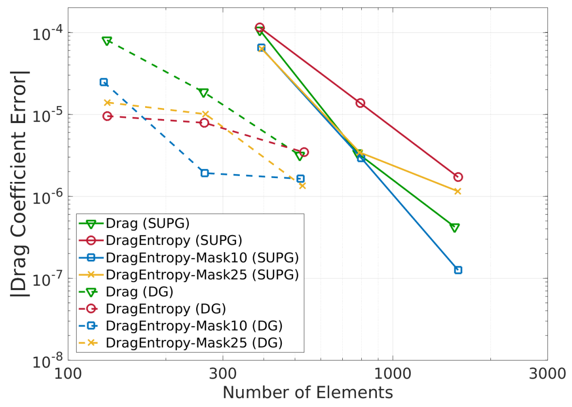
the combined approach when the indicators obtained using the fine-space output-based adjoint are equally weighted with those from the entropy variables. To potentially remedy this problem, a version of the combined approach with a mask, outlined in Subsection 3.4.3, was implemented for this case.

The version of the combined approach implemented for this case is very similar to the version shown in Subsection 6.2.2. However, there are a few differences. In this version, the error indicator obtained from the output-based adjoint that is used to create the mask is from the fine space. Another difference is the mesh adaptation strategy used in Subsection 6.2.2 was the hanging node approach, while for this case we are using MOESS. The masked error indicator is used everywhere in the MOESS algorithm, except for the element-local error sampling portion used to determine the optimal local refinement of each element (see Subsection 5.4.2). For the sampling, we use the standard combined error indicator. This is done since after the mask is applied to create a new error indicator array over all of the elements, many of the values in that array are now zero. To prevent pollution of the local samples by using the masked error indicator, the error indicator using the combined approach is used for this portion of the algorithm.

Figure 6.35 shows the effect of the masking on the combined approach for the drag coefficient errors. We consider combined approaches that use the same masking percentages as before, 10% and 25%. All of the cases are at $p = 2$ and use MOESS for the mesh adaptation approach. Regardless of the discretization used, there is a noticeable benefit to using the mask compared



(a) Drag Error Versus Degrees of Freedom

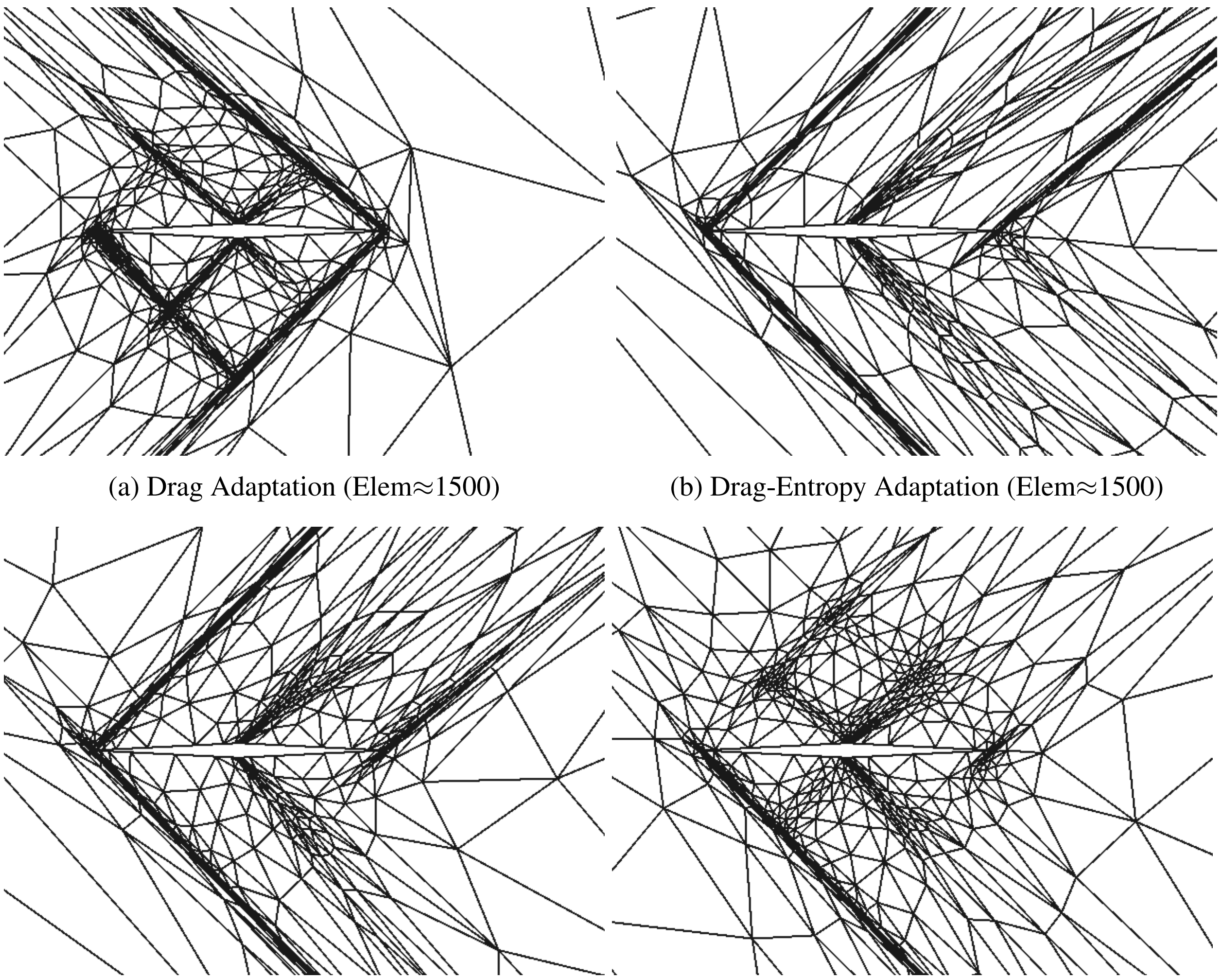


(b) Drag Error Versus Number of Elements

Figure 6.35: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Effect of the combined approach using a mask on drag coefficient convergence histories using SUPG and DG at $p = 2$.

to just the standard combined approach. This is not surprising given how much higher the drag coefficient error was in Figure 6.31 when using entropy variables alone. What is also promising is the combined approach with the 10% mask performs better than the output-based adjoint case for both SUPG and DG. For the 10% mask case with SUPG, we even observe a convergence rate of 5, which is equal to the super-convergent rate of $2p+1$. This shows that while the entropy variables alone are not a good error indicator for this case, they do provide useful information for error estimation that can be exploited to yield better results compared to just using the error indicators obtained from the fine-space output-based adjoint.

Sample meshes of the cases that use the combined approach that incorporates the mask, as well as the output-based adjoint approach and the standard combined approach, are shown in Figure 6.36. All of the meshes shown in this figure were generated using the SUPG discretization. When using the 25% mask, there is far less refinement in the area where the shock emanating from



(a) Drag Adaptation (Elem≈1500)

(b) Drag-Entropy Adaptation (Elem≈1500)

(c) Drag Adaptation with 25% Mask (Elem≈1500)

(d) Drag Adaptation with 10% Mask (Elem≈1500)

Figure 6.36: Diamond $M_\infty = 1.5$, $\alpha = 2°$: Effect of the combined approach using a mask on meshes generated using SUPG at $p = 2$.

the trailing edge is, especially far away from the airfoil. This leads to some improvement compared to the standard combined approach, but not nearly as much compared to using the 10% mask. For that case, most of the refinement is concentrated near the airfoil. There is still a lot of refinement far away from the airfoil in the region consumed by the leading edge shock. However, there is far less refinement in the "reverse wake" region emanating backward from the leading edge compared to the output-based adjoint approach. This explains why there is a benefit to using the combined approach with the 10% mask. Much of the refinement in this region, especially when it is far from the airfoil, is unnecessary to obtain an accurate drag coefficient.

Based on the results from this case, it is evident that the output-based adjoint approach and the entropy-variable approach refine regions of the domain may not lead to lower output error estimates. For the entropy-variable case, it was the shocks far away from the airfoil, while for the output-based adjoint it was the "reverse wake" region. By using a version of the combined approach that appropriately weights one approach over the other, it is possible to achieve lower error estimates.

6.3 NACA 0012 Airfoil With Unsteady Gust

A case that consists of an encounter between a NACA 0012 airfoil and a vertical gust is used in this work to test the combined output-based and entropy-based adjoint approach for unsteady simulations. All cases in this section use the DG discretization. The results in this section were previously published in [123].

6.3.1 Initial Conditions

The initial mesh, shown in Figure 6.37, is made up of 664 triangular elements with a farfield situated 100 chord lengths away from the airfoil. Curved elements are used to represent the airfoil geometry.

The initial condition is a converged steady-state solution at $Re = 1000$, $M_\infty = 0.4$, $\alpha = 2^\circ$, with adiabatic boundary conditions. The Prandtl number is set to $Pr = 0.72$ and viscosity is held constant. After the steady-state solution is obtained on the initial mesh, a vertical gust is superimposed on the solution by perturbing the vertical velocity ahead of the airfoil with a Gaussian function of amplitude $0.3U_\infty$, standard deviation of one chord length, centered five chords upstream of the leading edge, and offset to reach zero perturbation one chord length upstream of the leading edge. The location of the gust at $t = 0$ is shown in Figure 6.38a. The airfoil chord length is also set to $c = 1$. The simulation runs from $t = 0$ to $t = 15$, where one time unit is simply the unit chord divided by the freestream velocity. The initial number of time steps is $N_t = 15$.

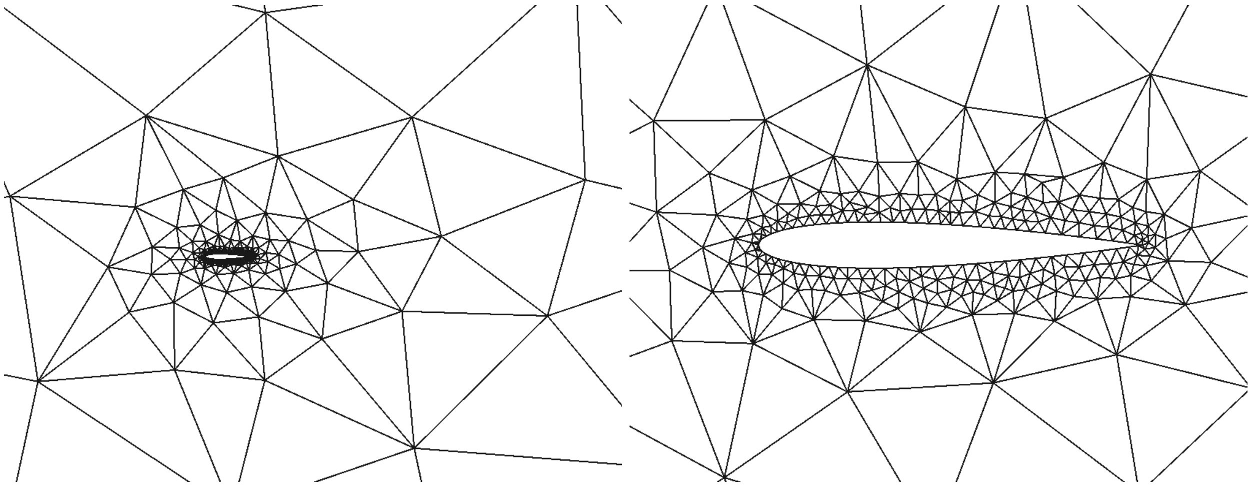

(a) Initial Mesh (b) Initial Mesh (Zoom)

Figure 6.37: Initial mesh for unsteady gust simulation.

DIRK3 is used to advance the primal solution in time, while DIRK4 is used to march the adjoint backward in time. Mach number contours showing the gust-inducing flow separation and vortex shedding are shown in Figure 6.38b.

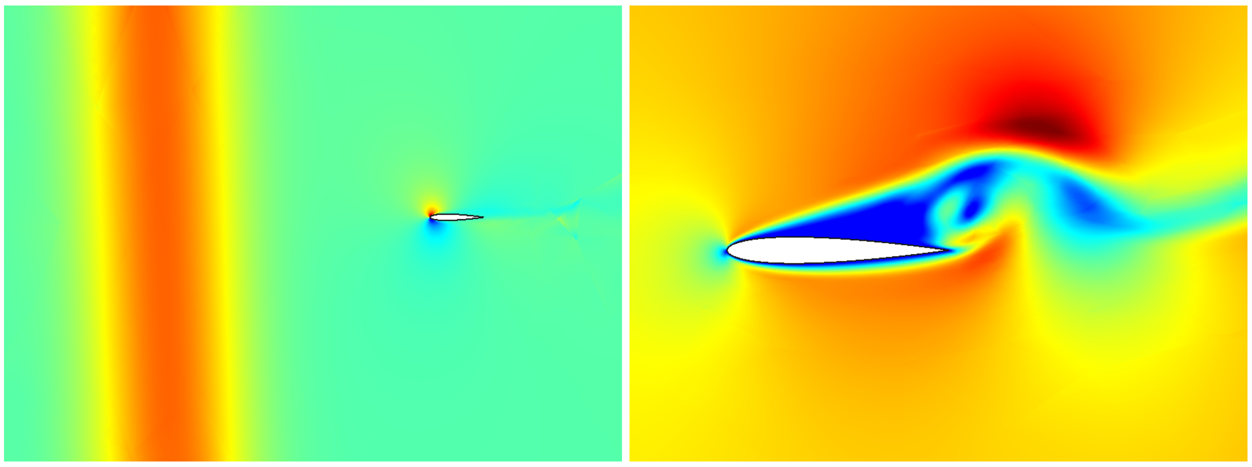

(a) Y-Momentum at $t = 0$ (b) Mach Number at $t = 7.5$ (0 to 0.6)

Figure 6.38: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Primal solution.

The mesh adaptation strategy used for this case was MOESS for unsteady problems, which was reviewed in Subsection 5.4.5. It is worth noting that all combined approaches used for this case rely on just the entropy variables for the sampling approach outlined in Subsection 5.4.2 and not the combined error indicators. This decision was made because of the complexity in the allocation of degrees of freedom for unsteady simulations (see Subsection 4.2.4).

As previously mentioned in Subsection 4.2.4, it is desirable, for comparison purposes, that the degrees of freedom are kept relatively consistent among the various adaptation approach used. Thus, fixed degrees of freedom targets were used to drive the mesh optimization instead of a fixed growth rate. For this case five different targets were considered, starting with 240,000, and increasing by a factor of 2 until 3,840,000 degrees of freedom. At each degrees of freedom target, 15 adaptive iterations were considered for the cases to settle on the specified target. Averages of the desired output were taken over the last seven iterations, with the maximum and minimum values discarded to reduce noise. The computational time to run the 15 adaptive iterations was also recorded for each case to compare relative computation times. All cases were run in parallel with four cores on the high-performance computing cluster, Flux, at the University of Michigan. The core hardware is 2.5 to 2.8 GHz Intel Xeon processors all connected with Infiniband networking. Each core was assigned 4GB of RAM per job. The computational times shown in this paper are not exact, as portions of the solver have not been fully optimized. The computational times are included in this work to show the relative performance of the various approaches. They are not meant to be taken as definitive.

The output of interest in this simulation is the weighted time-integral of the lift force,

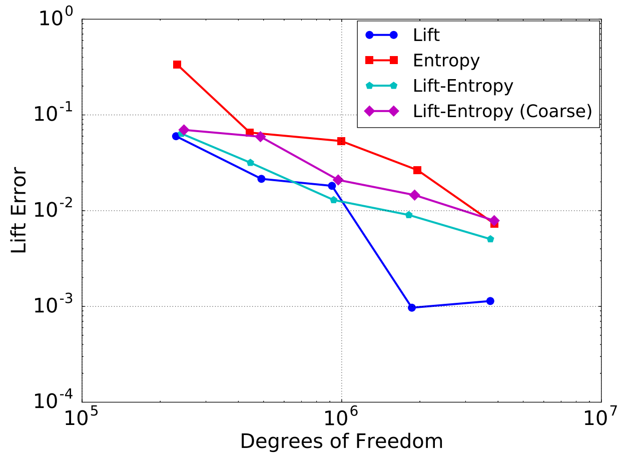
$$\bar{J} = \int_0^T w(t)L(t)dt, \tag{6.3}$$

where the temporal weighting function, $w(t) = \exp(-0.5(t-7.5)^2)$, is used to smooth the instantaneous lift force on the airfoil, $L(t)$, over time. The weighted time-integral of the lift force is the output of choice that guides the various adaptation strategies. The truth solution for this output is obtained by using an output-based refined mesh made up of 4,299 elements and running a simulation at $p = 3$ with 500 times steps of DIRK4.

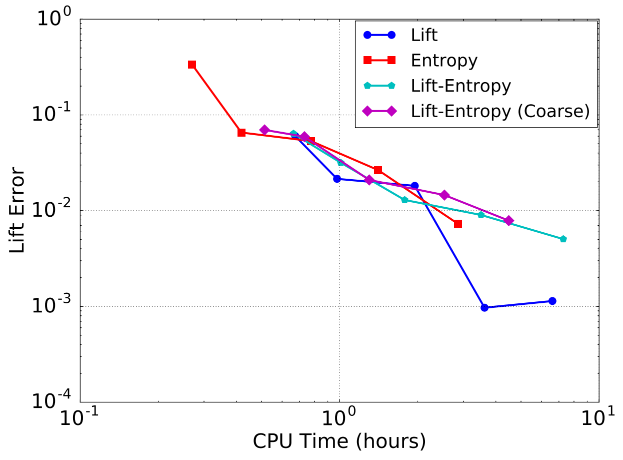
6.3.2 Results Using Non-Conservative Error Estimates

Figure 6.39a presents the convergence of the output error versus total degrees of freedom for various strategies using non-conservative error estimates. As previously mentioned, the outputs were averaged over the last seven iterations, with the maximum and minimum output values over that range excluded from the averaging. The corresponding degrees of freedom were averaged as well. The four strategies considered in this convergence study are the fine-space output-based adjoint approach, the entropy-variable approach, the standard combined approach with the fine-space output-based adjoint, and the combined approach that uses the coarser output-based adjoint. At the same degrees of freedom target, the output-based adjoint approach significantly outperforms the others. The lift coefficient error at the last degrees of freedom target is almost an entire magnitude lower.

The entropy-variable approach yields the highest magnitude, hence the rationale for exploring the combined approach. As expected, the combined approach that uses the fine-space output-based adjoint performs slightly better than the combined approach with the coarser-space output-based adjoint.



(a) Convergence versus total degrees of freedom



(b) Convergence versus CPU time

Figure 6.39: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Output error convergence using non-conservative error estimates.

However, the benefit of the coarser combined approach is evident in Figure 6.39b, where the same errors are plotted against the CPU time required to run the 15 iterations at each degrees of freedom target. The coarser combined approach is significantly cheaper than the fine-space combined approach. The reason is due to how much cheaper it is to solve for entropy variables on the fine-space. From these results, we observe that it takes more than double the amount of time to run the output-based adjoint approach, compared to the entropy-variable approach. Despite the fact the coarser combined approach is cheaper, it does not gain much lower error magnitudes relative to the fine-space combined approach. The issue is that the entropy variables do not provide nearly as accurate non-conservative error estimates.

One of the reasons the entropy-variable approach does not yield comparable output error results to the output-based approach is the emphasis on targeting spatial degrees of freedom over temporal. Figure 6.40 compares the number of spatial versus temporal degrees of freedom for the five targeted totals. For the highest degrees of freedom targets, the output-based adjoint approach has a much greater emphasis on temporal degrees of freedom compared to the other approaches. While the entropy-based approach initially targets more temporal refinement, by the final two adaptive targets it levels off and refines in space only. Based on comparing the relative performance in estimating the output, it is evident the entropy-variable approach does not yield an accurate temporal error estimate. This negatively impacts the two combined approaches where the preference of spatial over temporal refinement is even more drastic.

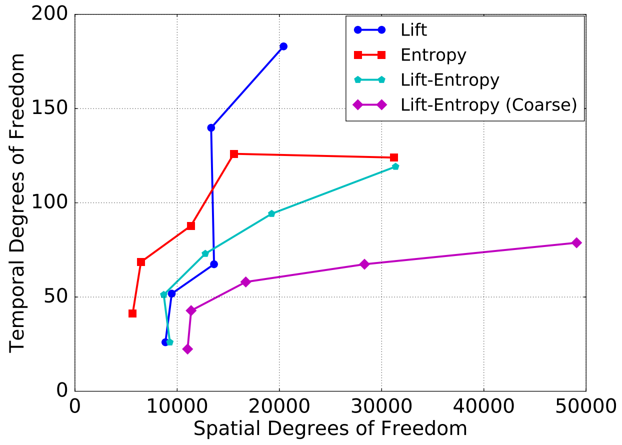
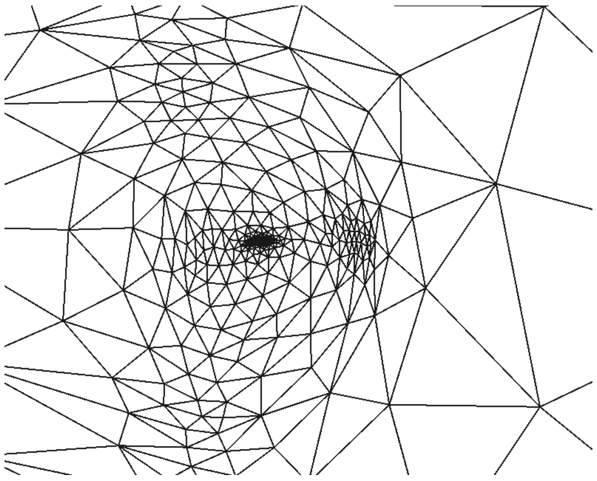


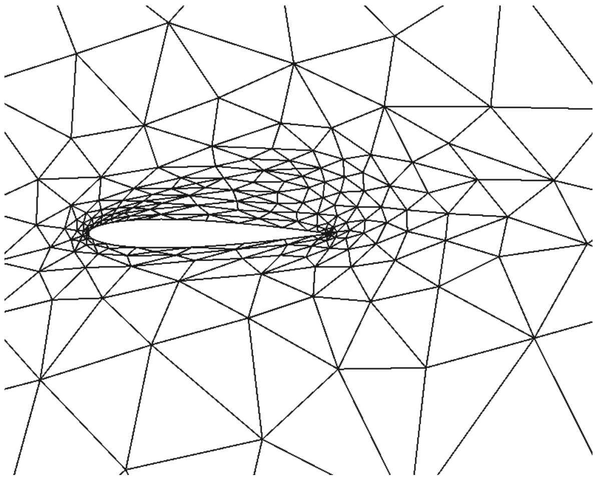
Figure 6.40: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Degrees of freedom using non-conservative error estimates.

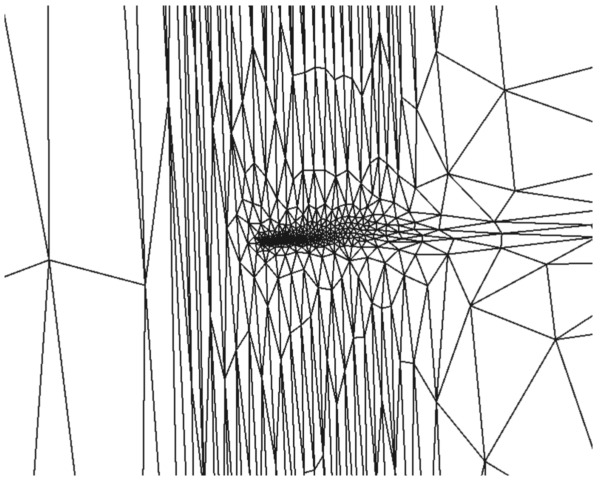
The final adapted mesh for the output-based, entropy-based, and coarse-space adjoint combined approaches are shown in Figure 6.41. From the previous figure, it was known that the output-based adjoint approach should have far fewer elements in the mesh. This conclusion can be observed in the zoomed-out view of the output-based mesh. There are fewer elements away from the airfoil compared to the other two approaches. The reason is that entropy is generated due to the gust in regions not near the airfoil. This is the common issue with the entropy-based approach for adaptation we observed for steady flow problems in Sections 6.1 and 6.2. The combined approach slightly mitigates this issue, as the mesh near the airfoil is more resolved than the mesh generated from the entropy-based approach. However, the combined approach still has far too much refinement above and below the airfoil, that extends unnecessarily to the outer boundaries of the mesh. In addition, the significant lack of temporal degrees of freedom for this approach negates the benefit of better spatial refinement near the mesh. These results show that using the output-based adjoint yields far superior conservative error estimates compared to using entropy variables and subsequently the combined approach.

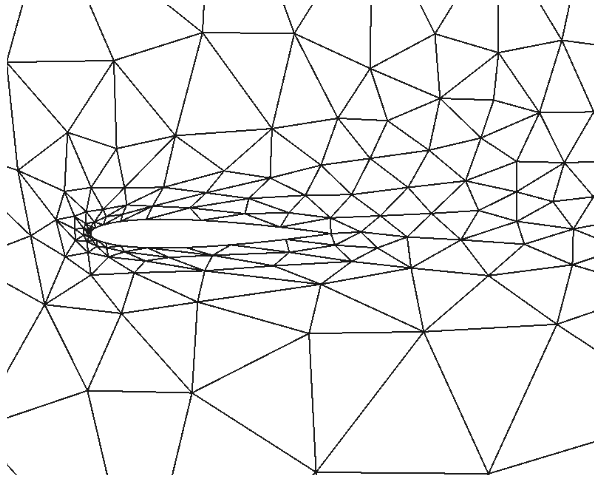
6.3.3 Results Using Conservative Error Estimates

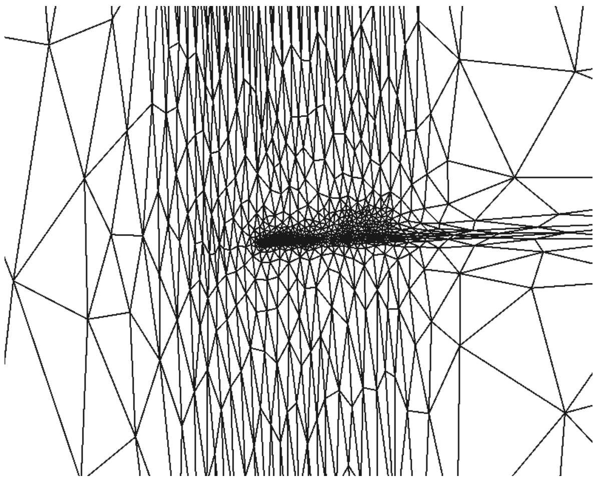
In this section, we will analyze the same case, with similar approaches, the only difference being the use of conservative error estimates, Equation 4.20, to govern the allocation of temporal and spatial degrees of freedom. The reason for using conservative error estimates is to eliminate the excess noise caused by the cancellation of errors.


(a) Lift (Zoom Out)


(b) Lift (Zoom In)


(c) Entropy (Zoom Out)


(d) Entropy (Zoom In)


(e) Lift-Entropy (Coarse) (Zoom Out)

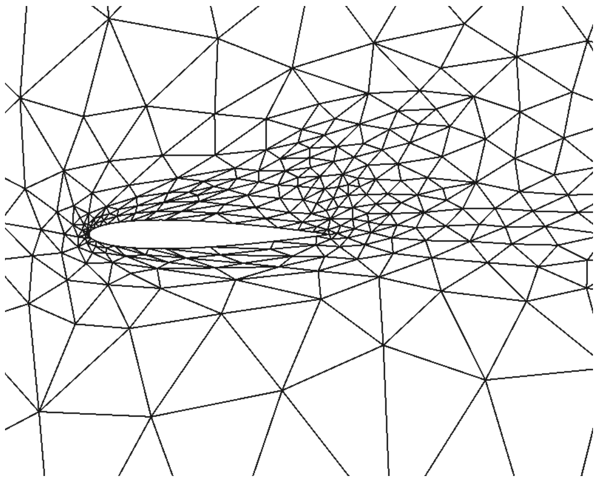
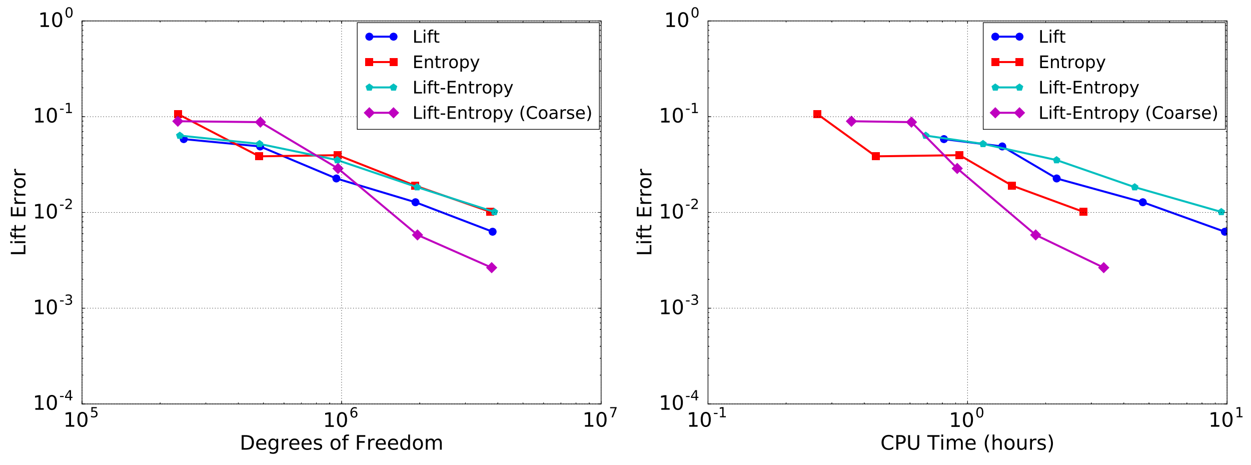

(f) Lift-Entropy (Coarse) (Zoom In)

Figure 6.41: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Final adapted meshes using non-conservative error estimates.

Figure 6.42a shows the integrated lift error convergence versus degrees of freedom at the same five degrees of freedom targets as before. The first major observation is how poor the output-based approach performs relative to when non-conservative error estimates were used. The entropy-based approach performs slightly better compared to before, but still worse than the output-based approach. Since the output-based approach output error magnitudes are higher than before, the combined approach suffers as well. Its relative performance to the combined approach using non-conservative error estimates is far less severe compared to the output-based approach. This is because the entropy-based approach remains relatively unchanged. However, the most interesting finding from this result is that not only does the coarser combined approach improve with conservative error estimates, but it also yields comparatively better performance than all of the other approaches, including the output-based approach. The effect of the improved performance in the coarse combined approach is even more apparent in Figure 6.42b, since the coarse combined approach is so much less computationally expensive to run.



(a) Convergence versus total degrees of freedom

(b) Convergence versus CPU time

Figure 6.42: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Output error convergence using conservative error estimates.

The reason the coarser combined approach outperforms the output-based approach is evident in Figure 6.43. Unlike when using the non-conservative error estimates, where the output-based approach preferred allocating more degrees of freedom to the temporal domain compared to the entropy-based approach, the opposite is true here. The average number of temporal degrees of freedom at the last degrees of freedom target for output-based cases decreased by almost half, while the spatial degrees of freedom subsequently doubled. This leads to the combined approach that uses the fine-space output-based adjoint also allocating a substantial decrease in the total amount of temporal degrees of freedom. For the entropy-based approach, the number of temporal degrees

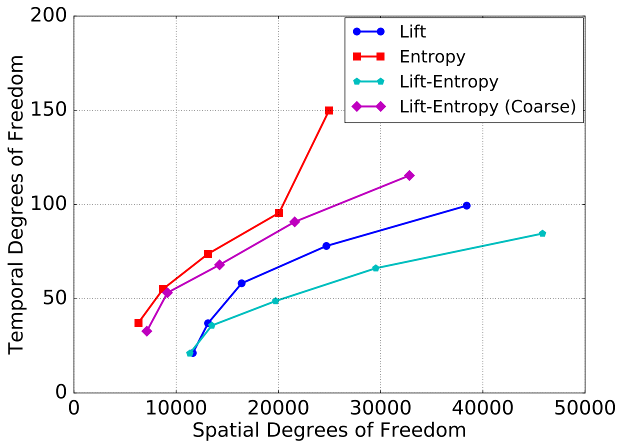
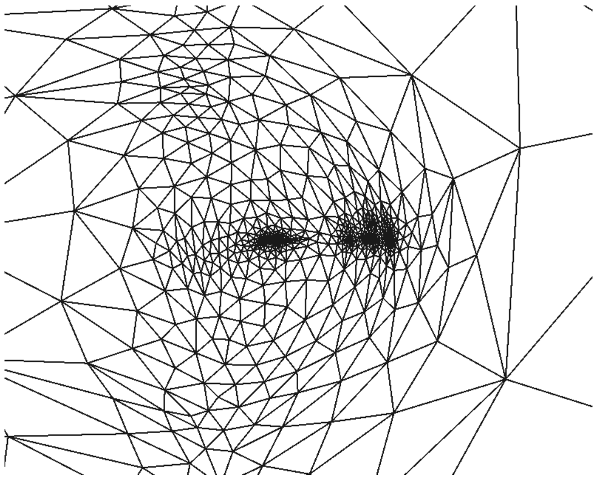
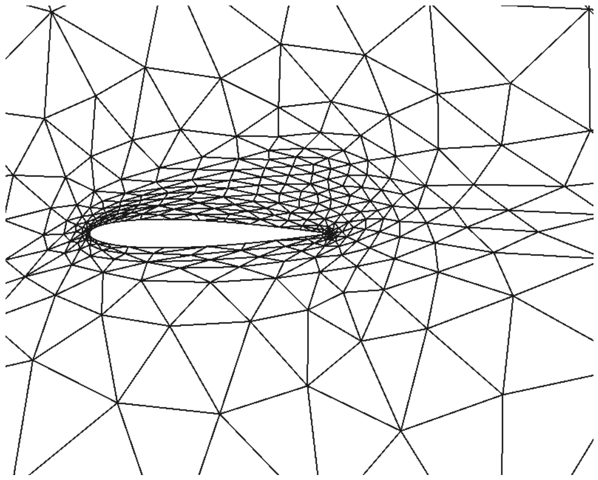


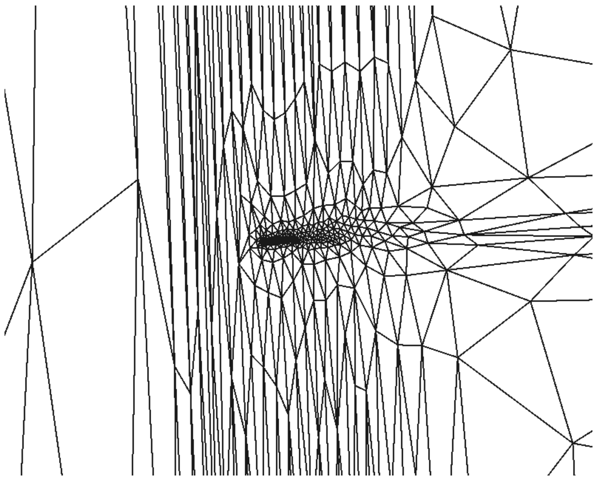
Figure 6.43: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Degrees of freedom using conservative error estimates.

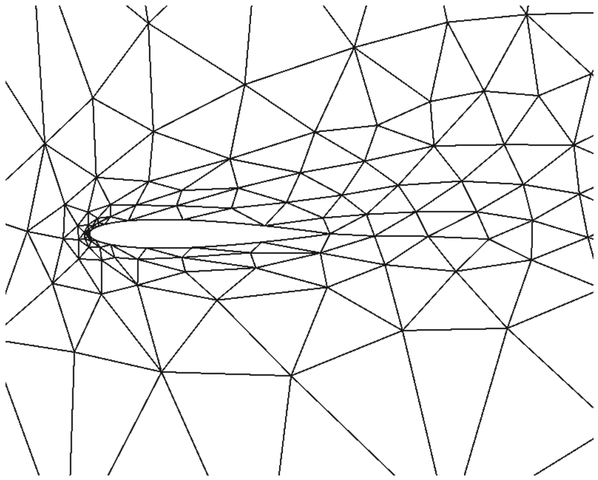
continues to increase with each target, as opposed to before where it leveled off. This leads to the coarser combined approach yielding far more temporal degrees of freedom than before. The reason the coarser combined approach sees an increase in temporal degrees of freedom while the fine-space combined approach does not is that the effect of the errors due to the output-based adjoint is far less significant. It is quite apparent that the output-based adjoint approach is significantly more inaccurate using conservative variables, while the entropy-variable approach is more accurate. This negatively impacts the fine-space combined approach, but positively impacts the coarser combined approach since the more accurate errors due to the entropy variables are now more dominant.

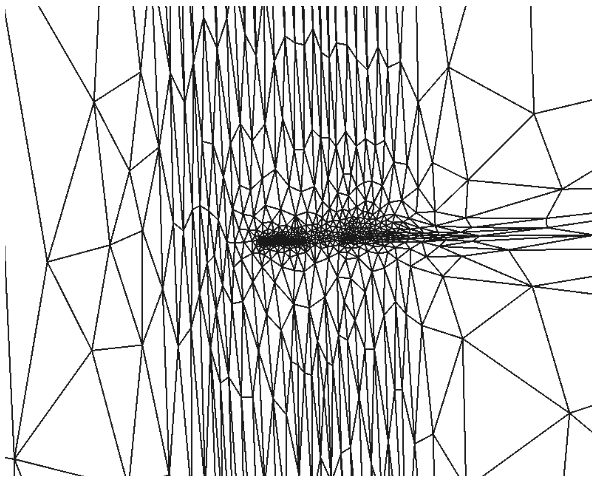
The final adapted mesh for the output-based approach in Figure 6.44 shows that the extra spatial degrees of freedom that were allocated to the temporal domain when using non-conservative variables are now used to refine the mesh downstream of the airfoil. Based on the poorer output error convergence, it is evident that the extra refinement there does not affect the integrated lift coefficient as much as more temporal degrees of freedom do. The entropy-based approach that yields more temporal degrees of freedom, which in theory should lead to more accuracy, produces a much coarser mesh near the airfoil. Ideally, the mesh would have been coarsened far away from the airfoil. This explains why the accuracy did not improve despite the necessary increase in temporal degrees of freedom. However, the coarse combined approach produced a relatively similar mesh near the airfoil compared to what was shown in Figure 6.41. The increase in temporal degrees of freedom and a similar mesh near the airfoil leads to a much more accurate output prediction.


(a) Lift (Zoom Out)


(b) Lift (Zoom In)


(c) Entropy (Zoom Out)


(d) Entropy (Zoom In)


(e) Lift-Entropy (Coarse) (Zoom Out)

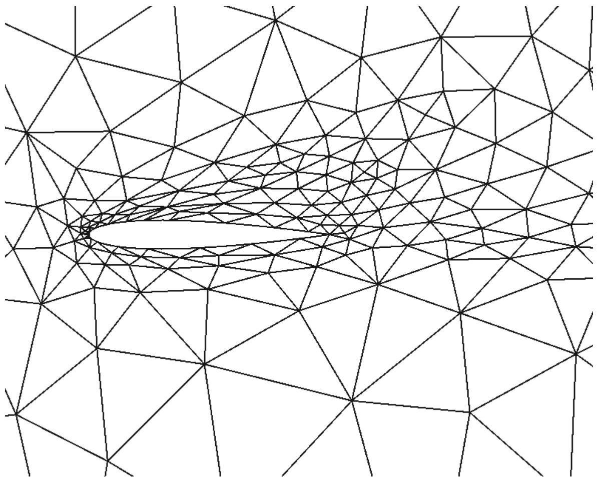
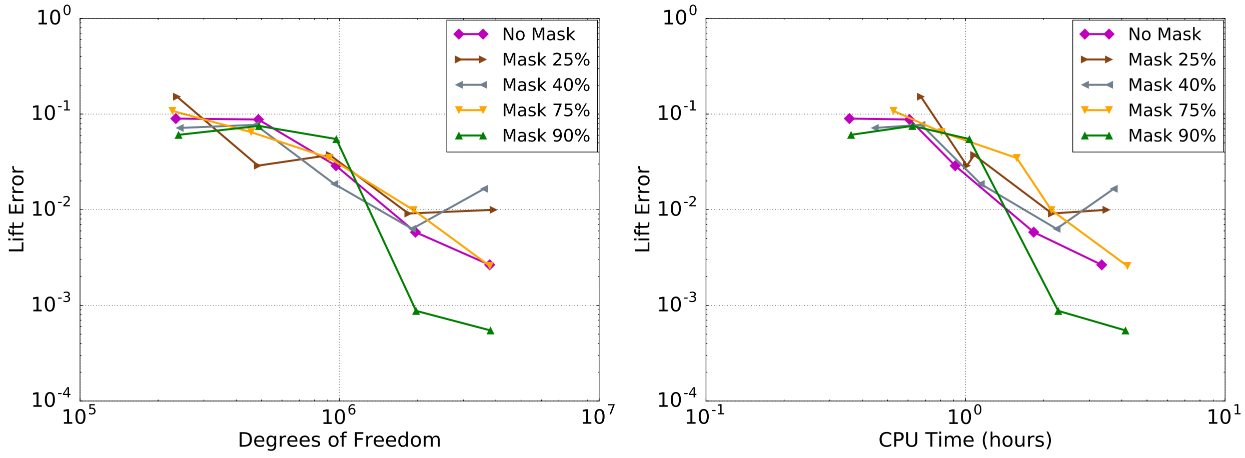

(f) Lift-Entropy (Coarse) (Zoom In)

Figure 6.44: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Final adapted meshes using conservative error estimates.

6.3.4 Effect of the Mask on the Combined Approach

In the previous subsection, we saw the potential of using the coarser combined approach with conservative error estimates. In this subsection, that method will be further analyzed with the incorporation of the mask outlined in Subsection 4.4.3. The purpose of the mask, generated from the elemental output-based spatial error indicator, is to improve the same indicator obtained using entropy variables by eliminating areas of the mesh far away from the airfoil that do not affect the desired output. For this study, we will use the following masking percentages: 25%, 40%, 75%, and 90%. All results shown in this section use conservative error estimates.

Figure 6.45 shows the output error convergence for the coarser combined approach, as well as the same approach with the four different masking values. For the finer degrees of freedom targets, the 90% masking case generates a significantly more accurate output than any of the other cases. The 75% masking case is almost directly comparable to not using a mask, while the 25% and 40% masks yield comparatively worse error estimates. Based on these results it appears that the lower the masking percentage, i.e. the more values in the output-based spatial error indicator that are set to zero, the less accurate the output is. Figure 6.45b shows an increase in CPU time due to the mask, which is to be expected since the masking algorithm is not free given the large size of the error indicators. However, the penalty is not sufficient enough to render this approach impractical. The benefit of using the 90% mask is still significant.



(a) Convergence versus total degrees of freedom (b) Convergence versus CPU time

Figure 6.45: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Effect of masking the coarser combined approach on output error convergence.

Figure 6.46 illustrates that the higher masking percentages yield more temporal degrees of freedom compared to not using a mask. This might explain why there was a benefit in using the 90% mask. As the masking percentages decrease, there generally is an increase in the allocation

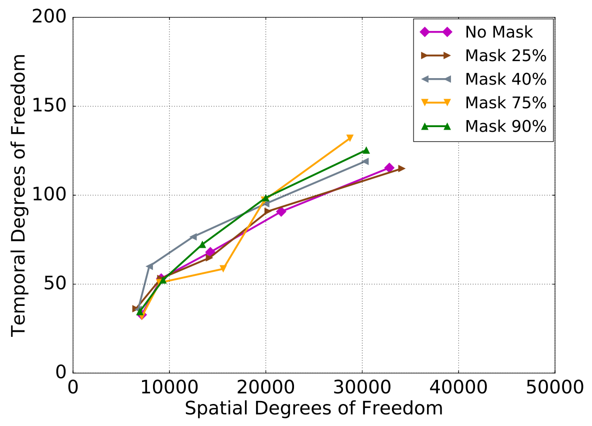
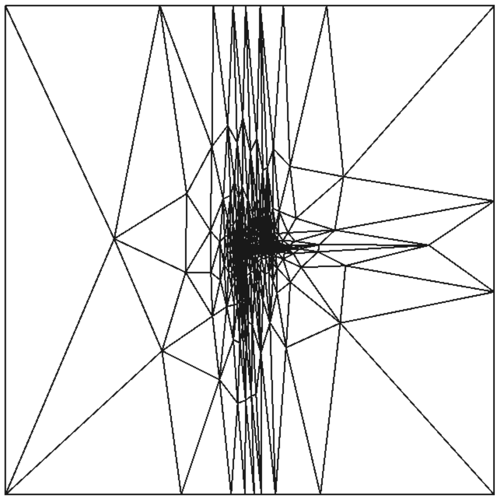


Figure 6.46: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Degrees of freedom using conservative error estimates for the combined approach using a mask.

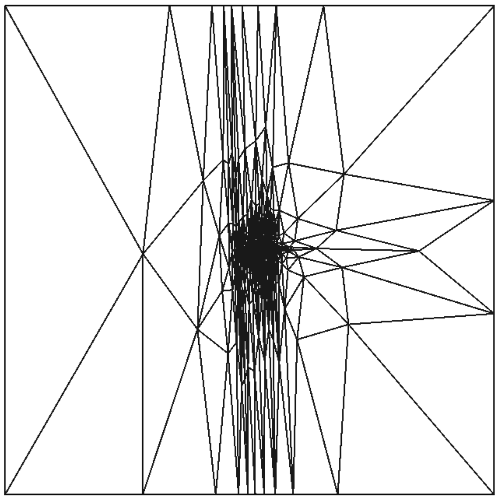
of spatial degrees of freedom, to the point where the 25% mask allocates more spatial degrees of freedom than it does without the mask.

In Subsection 4.4.3, we stated that the mask was applied before obtaining the aggregated spatial error estimates. Otherwise, there would be a disconnect between the spatial errors used to govern the allocation of degrees of freedom between the spatial and temporal domain, and the errors used to govern the relative mesh optimization with MOESS. The rationale for the mask is logical, as it effectively removes the influence of the spatial entropy-based error indicator in regions of the domain that do not have as strong of an influence on the output error estimate. This is apparent in Figure 6.47 where the lower the masking percentage, the less refinement far away from the airfoil. Unfortunately, these images might lead to the incorrect conclusion that the lower masking percentages should lead to more accurate output error estimates. Based on the output error convergence results, the opposite is true.

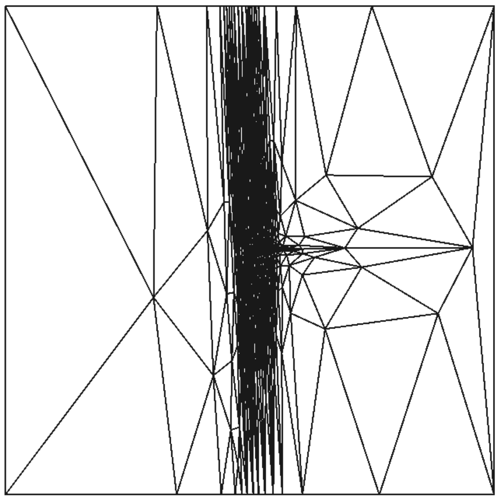
One possible reason why the lower percentage mask values lead to less accurate outputs is that the lower masking cases are not allocating enough degrees of freedom in the temporal domain. Perhaps using low masking percentages, which leads to a much strong effect in the spatial domain, leads to issues since no comparable mask is applied in the temporal domain. Another reason might be the lack of proper refinement near the nose of the airfoil for the lower masking percentage meshes, as evident in Figure 6.48. For the cases that use the 25% and 40% mask, the mesh ahead of the airfoil is quite coarse, with very anisotropic cells. This issue appears to be magnified as the masking percentage goes down. This might contribute to the fact that despite the mesh being less refined far away from the airfoil for these cases, the mesh near the airfoil does not improve. Overall, these results show that there can be a potential benefit to applying a mask on the entropy-based indicator. However, more work needs to be done regarding when the mask should be applied in the overall process, and should an additional mask in the temporal domain be considered as well.



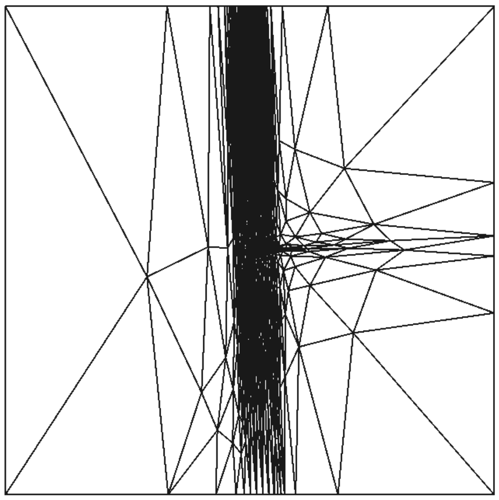
(a) 25% Mask



(b) 40% Mask

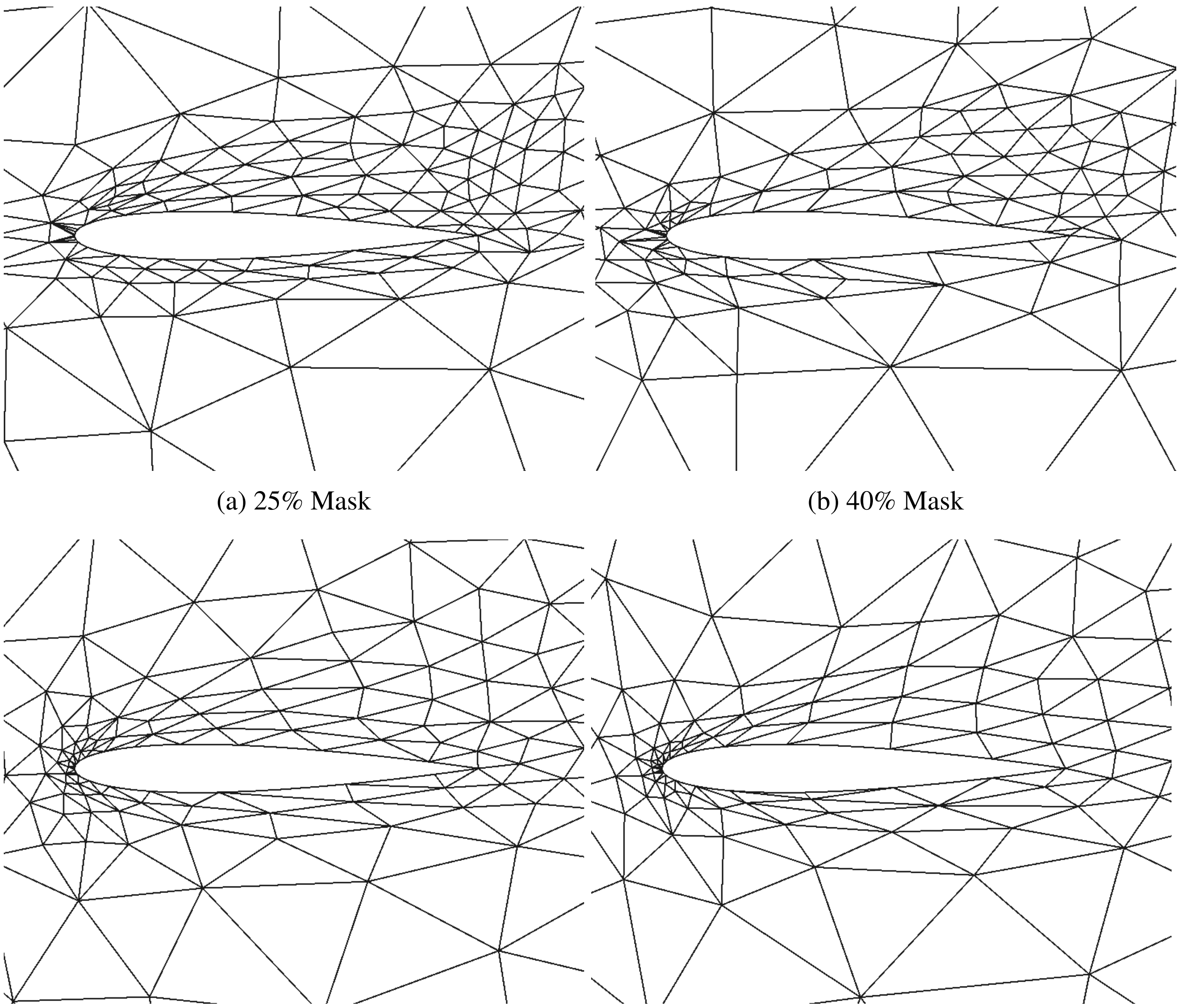


(c) 75% Mask



(d) 90% Mask

Figure 6.47: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Full view of adapted meshes for the combined approach using a mask.



(a) 25% Mask

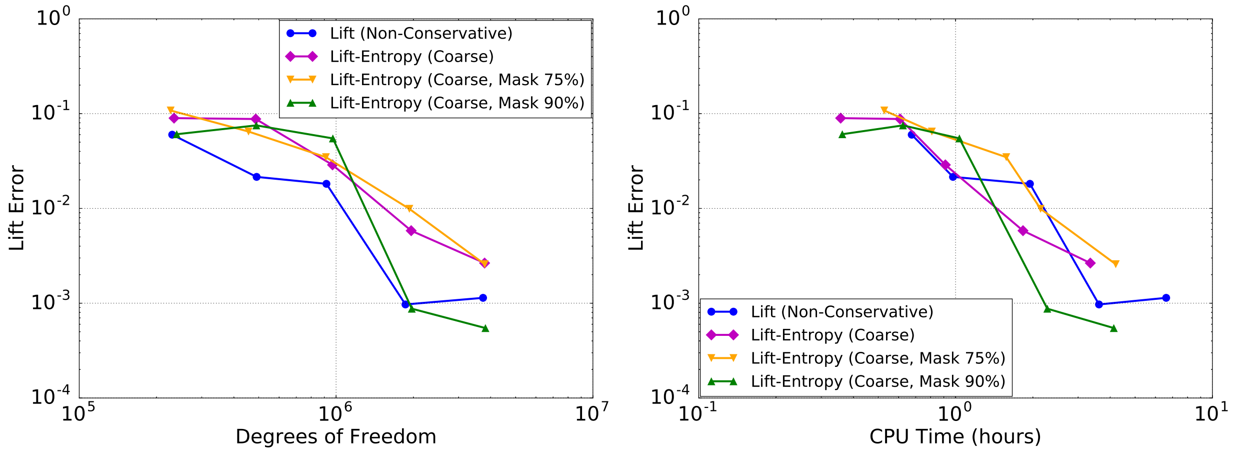
(b) 40% Mask

(c) 75% Mask

(d) 90% Mask

Figure 6.48: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Zoomed in view of final adapted meshes for the combined approach using a mask.

Figure 6.49 compares the output error convergence for the output-based adjoint approach using non-conservative error estimates to several coarser output adjoint combined approaches using conservative error estimates. The outputs obtained from these selected combined approaches compare favorably to the output from the fine-space output-based adjoint approach when the overall CPU time is taken into account. Since the combined approaches compute results faster than the output-based approach at the same degrees of freedom target, it can be argued that, especially for the combined approach with the 90% mask, the combined approaches produce more accurate results than the output-based approach. These results show the potential this new combined approach has for unsteady simulations in particular, due to its cheaper computational cost and sometimes more accurate error estimates.



(a) $f^{\text{adapt}} = 0.1$ (b) $f^{\text{adapt}} = 0.075$

Figure 6.49: NACA 0012 $M_\infty = 0.4$, $\alpha = 2°$: Output error convergence comparison for unsteady gust case.

CHAPTER 7

Conclusions and Future Work

7.1 Summary and Conclusions

The findings showcased in this dissertation were motivated by two major objectives. The first was to demonstrate that the combined approach for error estimation and mesh adaptation has strong potential for the high-fidelity CFD tools that will be necessary for future aerospace development programs. The second objective was to directly compare error estimates using the combined approach for multiple high-order discretizations. The purpose of this objective was not only to demonstrate robustness in the combined approach but also to provide useful findings that aid in the search for optimal high-order methods.

By combining the output-based adjoint indicator with the entropy-variable indicator, an appreciable benefit over the entropy-variable approach was found in several steady-state examples of aerodynamic interest. The entropy-variable approach targets areas of the mesh where spurious entropy is generated, even though those areas may not significantly affect the engineering output of interest. This sometimes prevents refinement in more critical regions of the mesh. By combining the entropy-variable indicator with the output-based adjoint indicator, this problem is marginalized, since output-based adjoint methods do not target these unnecessary areas of the mesh for refinement. In addition, the combined indicator approach sometimes yields improved performance compared to the standard output-based indicator approach due to the previously-discussed inefficiencies of the output-based approach. Since the output-based indicator is derived from approximate adjoint solutions, which may be under-resolved and noisy, these methods may end up targeting areas of the domain where additional refinement is not necessary. This is particularly the case with the flow ahead of the body of interest, near the stagnation streamline. Combining the indicators limits excessive refinement both upstream of and downstream from the body.

A significant effort of this work was the creation of a code that implements the SUPG discretization. Findings in this work demonstrated that the potential benefits in accuracy and computational cost that the combined approach offers are not exclusive to one type of finite-element discretization. The combined approach refined and coarsened the computational mesh in similar locations when using SUPG and DG. At similar mesh sizes, cases that utilized DG produced lower errors. This is not surprising given that DG uses more degrees of freedom per element compared to SUPG. When comparing SUPG directly to DG for cases that used similar degrees of freedom, SUPG generally produced lower output errors compared to DG for linear, $p = 1$, solution approximations. When the solution approximation order was increased to $p = 2$, the difference between SUPG and DG was not nearly as noticeable.

The benefits of using the combined indicator approach are not exclusive to one type of mesh adaptation approach, as benefits were demonstrated for cases using multiple types of mesh adaptation strategies. For cases that required highly anisotropic elements, using the combined approach with MOESS reduced the output error significantly compared to the other mesh adaptation strategies. While the combined approach shows an appreciable benefit for inviscid and viscous cases, the benefit is not as significant for RANS cases. The combined approaches produce comparable drag coefficient errors compared to the drag-coefficient-based approach. However, the combined approaches fail to produce reliable lift coefficient errors. Expanding the application scope of using the combined approach with masks to RANS cases is a potential topic for future research.

For the unsteady simulations investigated in this work, we observed that the combined approach often yields better integrated output error convergence than the entropy-based approach. In some cases, variations of the combined approach even produced better error convergence than the output-based approach. Using non-conservative error estimates, it was clear that the output-based approach yielded superior output error estimates compared to both the entropy-based and combined approaches. Non-conservative error estimates obtained using entropy variables focus the allocation of degrees of freedom excessively in the spatial domain and not nearly enough in the temporal domain. This negatively impacts the coarser combined approach since the error estimates from the entropy variables dominate. However, the exact opposite is true when conservative error estimates are used. In this case, the output-based approach excessively refines the mesh while sacrificing too much accuracy in the temporal domain. Conversely, the entropy-based approach produces a much better balance between the allocation of spatial and temporal degrees of freedom. This leads to the coarser combined approach yielding a much more accurate output since the entropy-based error estimates are far more accurate. The addition of the mask to the coarser combined approach shows potential for some of the cases. If the masking percentage is high, which means only a small portion of the entropy-based spatial error indicator is affected, the error estimates show substantial improvement. However, the mask can lead to comparatively worse performance if too much of the entropy-based spatial error indicator is affected. When directly comparing output error convergence generated using the output-based adjoint approach using non-conservative error estimates to several cheaper output-based adjoint combined approaches using conservative error estimates, we observe comparable results when the overall CPU time is taken into consideration. This demonstrates that the greatest potential benefits of the combined approach exist for unsteady simulations when the CPU time to complete the simulations is significant.

7.2 Future Work Directions

In this work, we have shown the potential of the combined approach for error estimation and mesh adaptation for two finite-element discretizations in a variety of CFD applications. However, there is more work that must be done to prove this as a realistic capability for the future. Ideas for directions in future work are summarized below:

- **Further Evaluate Cost-Effective Versions of the Combined Approach**
  This work provided examples where the combined approach that used indicators obtained from the coarse-space adjoint produced comparable, if not better, output error estimates compared to the fine-space output-based adjoint approach. However, those examples were limited to just one unsteady simulation. In addition, this approach was not tested using the SUPG discretization. While the combined approach that uses the fine-space adjoint is a good way to test the method, it is not a feasible option moving forward as it is more expensive than the fine-space output-based adjoint approach. To reduce the computational expense, the cheaper, coarse-space adjoint must be used in the combined approach; acting only as a guide for the entropy variable indicator approach to limit the unnecessary refinement inherent to the entropy variables approach for many non-smooth simulations.

- **More Unsteady Simulations**
  The benefit of leveraging the combined approach that uses a cheaper output-based adjoint error indicator is most apparent for unsteady simulations where the costs associated with the calculation of the fine-space output-based adjoint can be high. In this work, only one unsteady simulation was examined. While the results for this case showed promise, significantly more unsteady simulations over a large variety of aerospace applications must be studied to show the combined approach is robust. Testing the combined approach for unsteady simulations with SUPG or other high-order finite-element discretizations would also be worthwhile and give further confidence in the results presented in this work. In particular, unsteady RANS cases should be the focus of additional unsteady cases to vet the combined approach. When factoring in the cost difference of calculating the entropy-variable and output-based adjoints, we expect an improvement in the performance of the combined approach relative to steady-state RANS simulations.

- **Further Investigate the Use of Masks**
  For multiple applications presented in this work, a combined approach that utilized masking showed tremendous benefit compared to just the standard combined approach. By masking a portion of the entropy-variable error indicator using a mask obtained from the output-based adjoint error indicator, we saw a sizable reduction in output error estimates for both steady and unsteady simulations. The main problem with the masking technique is determining what the ideal masking percentage should be. Vetting the masking approach over a larger array of relevant aerospace applications would help identify if the masking approach is robust and what the ideal masking percentage should be.

- **More Industry-Relevant Applications**
  To further demonstrate the potential of the combined approach, more industry-relevant applications must be tested with this approach. In this dissertation, only one three-dimensional case and only one case that used RANS were thoroughly presented. In addition, we only considered one turbulence model for the RANS equations. The combined approach needs to be vetted with more complex applications that not only further validate the approach, but also demonstrate its potential to possibly replace the more common fine-space output-based adjoint approach for error estimation used in many CFD solvers capable of mesh adaptation. The combined approach should also be tested for problems that use different physics, e.g. icing. As long as it is possible to formulate an adjoint based on a scalar output and one based on entropy variables for a particular set of physics or governing equations, the combined approach can be used for that application. For more complex problems where the computational costs of the fine-space adjoint problem are even greater, the combined approach that uses a cheaper output-based adjoint should demonstrate even more potential.

- **Implement the Combined Approach in Commercial Codes**
  In this dissertation, the combined approach was implemented into an existing DG code used by the computational fluid dynamics research group at the University of Michigan and a SUPG code written specifically for this research. To further analyze the combined approach for more industry-relevant applications, it would be worthwhile to implement the combined approach in commercial CFD software that has adjoint-based error estimation and mesh adaptation capabilities. Even if the software did not have entropy-variable adjoint capabilities, it is not difficult to add this capability as long as the adjoint-based error estimation framework exists in the mesh adaptation process. From there, it is not challenging to implement a version of the combined approach. Adding the combined approach capability to commercial CFD software would provide excellent opportunities for validating the accuracy and robustness of the approach for more industry-relevant problems.

- **Extend the Combined Approach to Large Eddy Simulations**
  While RANS remains the dominant approach to approximating fluid flow in industry-relevant applications, large eddy simulation (LES) is gaining traction as computational resources become more powerful and plentiful. Unlike in direct numerical simulation (DNS) where all spatial and temporal scales of the turbulence are resolved, in LES the smallest length scales are spatially filtered out and only the scales that contain the most energy are resolved. This leads to a higher fidelity solution compared to RANS, at the expense of a large increase in computation cost. This makes the calculation of the fine-space output-based adjoint incredibly expensive for complex problems. However, the benefits of the entropy-variable adjoint approach have already been extended for adaptation with LES [139]. While a fine-space output-based adjoint approach would not be practical for LES due to the excessive computational cost, an approach that uses a cheaper, coarse-space output-based adjoint could be more practical. Combining this entropy-variable approach for LES with a coarse-space output-based adjoint would be a valuable direction for expanding the scope of the combined approach.

APPENDIX A

Boundary Conditions

This appendix details the implementation of boundary conditions for the compressible Navier-Stokes equations and RANS equations. The details are based on [24, 140]. Figure A.1 following key quantities to describe the boundary conditions are used in this appendix:

$$\begin{aligned}
\mathbf{u}^+ &= \text{state just inside the computational domain} \\
\mathbf{u}^b &= \text{state on the boundary, just outside computational domain} \\
n_i &= \text{normal vector pointing out of domain} \\
\hat{\mathbf{F}}^b &= \text{numerical boundary flux in the direction of } n_i \text{ on the boundary}
\end{aligned}$$

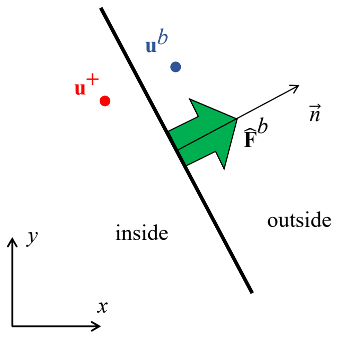


Figure A.1: Boundary flux and state.

A.1 Full-State

For the full-state boundary condition, the entire boundary state, $\mathbf{u}^b$, is known. This does not mean that the entire state is necessary for the boundary condition. In the case of a subsonic inflow, only one characteristic (acoustic wave) will physically leave the domain even though the entire boundary state may be specified. To account for the fact that the full-state will typically be an over-specification of the boundary condition, a numerical flux is used to compute the boundary condition. The inviscid flux is calculated using the Roe approximate Riemann solver:

$$\hat{\mathbf{F}}^b = \hat{\mathbf{F}}\left(\mathbf{u}^+, \mathbf{u}^b, n_i\right). \tag{A.1}$$

The viscous fluxes at the boundary are computed based on the Dirichlet boundary flux indicated in Table 2.1:

$$\widehat{\mathbf{G}}^b = -\mathbf{K}_{ij}^b \nabla \mathbf{u}_h^+ n_i + \eta \boldsymbol{\delta}_i n_i. \tag{A.2}$$

A.2 Inviscid Wall

The inviscid wall boundary condition represents a solid object that flow cannot pass through. This boundary condition is suitable for the Euler equations. It is different from a no-slip boundary condition as that requirement is only necessary for viscous flows. The inviscid boundary flux is given by

$$\hat{\mathbf{F}}^b = \begin{bmatrix} 0 \\ p^b n_i \\ 0 \end{bmatrix}, \tag{A.3}$$

where

$$p^b = (\gamma - 1)\left[\rho^+ E^+ - \frac{1}{2}\rho^+ |v_i^b|^2\right]. \tag{A.4}$$

The boundary velocity, v_i^b, is tangential. This means that it is equal to the interior velocity with the wall-normal component taken out,

$$v_i^b = v_i^+ - \left(v_i^+ \cdot n_i\right) n_i. \tag{A.5}$$

The other quantities in the boundary pressure equation, ρ^+, and E^+, are calculated based on interior density and energy, respectively.

A.3 Adiabatic, No-Slip Wall

For the viscous Navier-Stokes equations and RANS, a no-slip wall boundary condition is required. For an adiabatic, no-slip wall, velocity, eddy viscosity, and heat flux are all set to zero. The interior boundary state, $\mathbf{u}_h^b$, is created using density and energy from the interior state with the velocity component terms all set to zero. For the three-dimensional RANS equations, this can be expressed mathematically as

$$\mathbf{u}_h^b = \begin{bmatrix} \rho^+ \\ 0 \\ 0 \\ 0 \\ \rho^+ E^+ \\ 0 \end{bmatrix} \tag{A.6}$$

The numerical inviscid flux is computed using the analytical flux based on $\mathbf{u}_h^b$:

$$\widehat{\mathbf{F}}^b = \mathbf{F}\left(\mathbf{u}_h^b\right). \tag{A.7}$$

There is no need to calculate the Roe-averaged flux for this case. The viscous fluxes are evaluated based on the Dirichlet flux in Table 2.1. The exception is the viscous energy flux is set to zero to satisfy the adiabatic wall condition, as long as the wall is stationary. For the RANS equations, this can be expressed as

$$\widehat{\mathbf{G}}^b = \begin{cases} -\mathbf{K}_{ij}^b \nabla \mathbf{u}_h^+ n_i + \eta \boldsymbol{\delta}_i n_i & i \neq d+2 \\ 0 & i = d+2. \end{cases} \tag{A.8}$$

APPENDIX B

Curved Elements

For finite-element methods such as DG and SUPG, the geometry of an object is represented by the shape of the neighboring elements. If the elements are linear, straight lines represent the geometry. However, for objects such as airfoils that are not piecewise linear and instead curved, using "panel" representation of the geometry may lead to computational errors when a high-order solution approximation is used. For these situations, the elements near the geometry must be curved, as shown in Figure B.1. Linear geometry especially causes problems for high-order discretizations since the

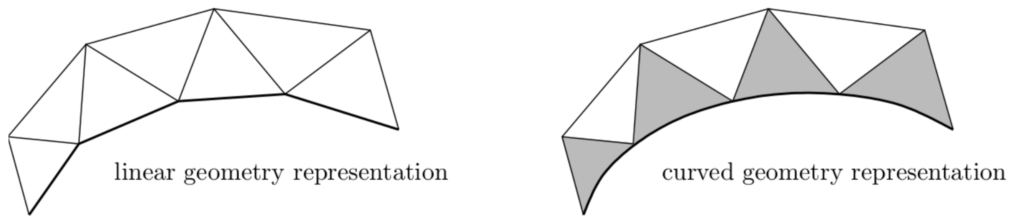

linear geometry representation
curved geometry representation

Figure B.1: Linear versus curved representation of geometry (Figure reproduced from [141]).

fidelity of the geometry is inferior compared to the solution approximation. When trying to represent a highly curved airfoil with linear elements, flow over a smooth surface will be replaced instead by flow over a surface with a series of sharp corners. Unless the mesh is incredibly fine, those sharp corners will have an impact on the solution and negate the benefits of the high-order solution approximation. By using curved elements to represent the geometry, the benefits of a high-order solution approximation are not only more attainable, but the solution is more accurate. The details found in the appendix are based on [141].

B.1 Geometry Mapping

In this work, we generate curved elements using a nonlinear map from a reference triangle. Specifically, a high-order polynomial mapping is used to represent the element in physical space to take advantage of the existing infrastructure in codes that incorporate high-order solvers. The polynomial mapping is shown in Figure B.2, where q is the order of the polynomial, $\vec{\xi} = [\xi, \eta]^T$ are the reference space coordinates, and $\vec{x} = [x, y]^T$ are the coordinates in global space. The total number of degrees of freedom in the mapping is $N_q = (q+1)(q+2)/2$.

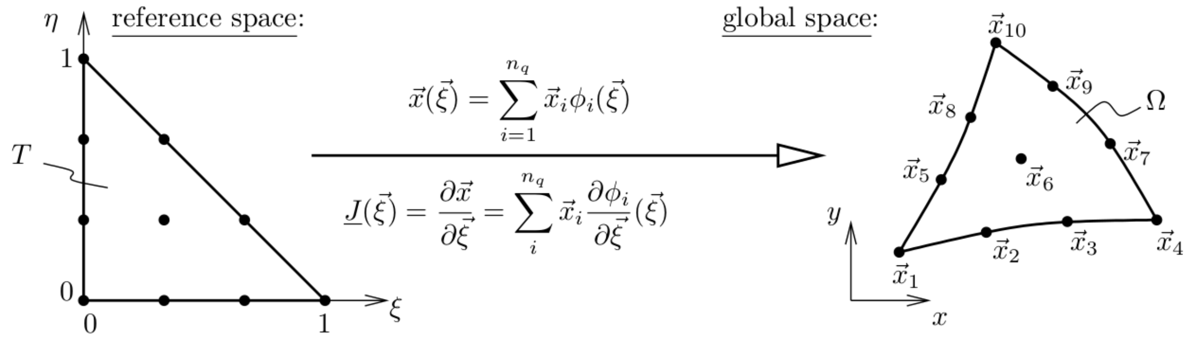

reference space:
global space:
η
1
0
0
1
ξ
T
$\vec{x}(\vec{\xi}) = \sum_{i=1}^{n_q} \vec{x}_i \phi_i(\vec{\xi})$
$\underline{J}(\vec{\xi}) = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \sum_i^{n_q} \vec{x}_i \frac{\partial \phi_i}{\partial \vec{\xi}}(\vec{\xi})$
$\vec{x}_{10}$
$\vec{x}_9$
Ω
$\vec{x}_8$
$\vec{x}_7$
$\vec{x}_5$
$\vec{x}_6$
y
$\vec{x}_1$
$\vec{x}_2$
$\vec{x}_3$
$\vec{x}_4$
x

Figure B.2: Elemental notation for a DG discretization on a two-dimensional triangle (Figure reproduced from [141]).

In this work Lagrange basis functions, denoted as $\phi_i(\vec{\xi})$ in Figure B.2, are used for the polynomials, just as they are for the solution approximation. They are also advantageous since they provide a convenient way to represent the geometry since Lagrange interpolation inherently interpolates the geometry between points.

B.2 Normal Calculation

The calculation of the normal vector for a curved edge is more challenging than calculating a normal on a linear edge, since the normal does not remain constant along the entire edge. For a curved edge, the normal direction may not be consistent. To begin the computation of the normal vector, $\vec{n}$, along a curved edge, we note the geometric order, q of an element adjacent to the edge. In the case where q is different between two elements, the lower q is chosen since the normal will be the same between the two elements. As a one-dimensional structure, we represent the edge with $q+1$ nodes and define the edge interpolant as

$$\vec{x}_{\text{edge}}(\sigma) = \sum_{j=1}^{q+1} \vec{x}_{\text{edge},j} \phi_j^{1d}(\sigma), \tag{B.1}$$

where

$$\sigma = \text{reference position along the edge, } \sigma \in (0,1)$$
$$\vec{x}_{\text{edge},j} = q+1 \text{ global nodes on edge, } 1 \le j \le (q+1)$$
$$\phi_j^{1d}(\sigma) = \text{1D Lagrange basis on } q+1 \text{ equally-spaced nodes in } [0,1]$$

Along the edge, the tangent vector is written as

$$\vec{t}\frac{ds}{d\sigma} = \frac{d\vec{x}_{\text{edge}}}{d\sigma} = \sum_{j=1}^{q+1} \vec{x}_{\text{edge},j} \frac{d\sigma^{1d}}{d\sigma}(\sigma), \tag{B.2}$$

where s is the arc-length position along the edge. The normal associated with the edge is

$$\vec{n} = \vec{t} \times \vec{k} = [t_y, -t_x]. \tag{B.3}$$

In order to get the tangent vector, $\vec{t}$, we can normalize the direction vector $\frac{d\vec{x}_{\text{edge}}}{d\sigma}$ since

$$\frac{ds}{d\sigma} = \left|\frac{d\vec{x}_{\text{edge}}}{d\sigma}\right|. \tag{B.4}$$

B.3 Curved Edge Integration

What makes integrating on a curved edge different compared to a linear edge is that the arc length and normal are not constant along the edge. Integration of a scalar becomes

$$\int_{\text{edge}} f(\vec{x})ds = \int_0^1 f(\vec{x}(\sigma))\frac{ds}{d\sigma}d\sigma, \tag{B.5}$$

where $ds/d\sigma$ is key for providing information about the length of the edge in global-space. Integration of a vector dotted with a normal becomes

$$\int_{\text{edge}} \vec{f} \cdot \vec{n} ds = \int_0^1 \vec{f} \cdot \vec{n}\frac{ds}{d\sigma}d\sigma. \tag{B.6}$$

Note the product $\vec{n}(\sigma_q)\frac{ds}{d\sigma}(\sigma_q)$ directly as $\frac{d\vec{x}_{\text{edge}}}{d\sigma} \times \hat{k}$.

APPENDIX C

Time Schemes

This appendix presents details of various multi-step and multi-stage methods for advancing the following system of ordinary differential equations:

$$\bar{\mathbf{R}}(\mathbf{U}) \equiv \mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{C.1}$$

where $\mathbf{U} \in \mathbb{R}^N$ is state vector of basis function coefficients, $\mathbf{R}$ is the discrete spatial residual vector, $\mathbf{M}$ is the mass matrix, and $\bar{\mathbf{R}}$ is the unsteady residual. Specifically, we focus on the general backward-difference multi-step method and implicit Runge-Kutta methods. The details found in the appendix can be found in [106].

C.1 Backwards Difference Formula

The general backward difference formula (BDF) takes the form of

$$\frac{\mathbf{M}}{\Delta t}\sum_{i=0}^{n_{\text{step}}} a_i \mathbf{U}^{n+1-i} + \mathbf{R}\left(\mathbf{U}^{n+1}, t^{n+1}\right) = \mathbf{0}, \tag{C.2}$$

where n_{step} is the number of steps in the method and a_i are coefficients presented in Table C.1.

Name	a_0	a_1	a_2	a_3
BDF1	1	-1	0	0
BDF2	$\frac{3}{2}$	-2	$\frac{1}{2}$	0
BDF3	-$\frac{1}{3}$	$\frac{3}{2}$	-3	$\frac{11}{6}$

Table C.1: Coefficients of various BDF methods.

C.2 Diagonally-Implicit Runge-Kutta

A diagonally-implicit Runge-Kutta (DIRK) method for advancing the state $\mathbf{U}$ over a time step of size Δt takes the form of

$$\begin{aligned}
\text{for} \qquad & i = 1 \ : \ n_{\text{stage}} \\
& \mathbf{S}_i = -\frac{\mathbf{M}}{\Delta t}\mathbf{W}^0 + \sum_{j=1}^{i-1} a_{ij}\mathbf{R}\left(\mathbf{W}^j, t_j\right) \\
& \text{solve: } \frac{\mathbf{M}}{\Delta t}\mathbf{W}^i + a_{ij}\mathbf{R}\left(\mathbf{W}^i, t_i\right) + \mathbf{S}_i = \mathbf{0} \\
\text{end} &
\end{aligned} \tag{C.3}$$

where $\mathbf{W}^0 = \mathbf{U}^n$ is that initial state at $t = 0$, $\mathbf{U}^{n+1} = \mathbf{W}^{n_{\text{stage}}}$ is the state at the final time, and $t_i = t^n + b_i\Delta t$ are the stage times. The coefficients a_{ij} and b_i define the method and can be found in the aforementioned reference [106].

BIBLIOGRAPHY

[1] Bengelink, R. L., and Rubbert, P. E., "The impact of CFD on the airplane design process: today and tomorrow," *SAE Transactions*, Vol. 100, 1991, pp. 2059–2068.

[2] Jameson, A., and Vassberg, J., "Computational fluid dynamics for aerodynamic design - its current and future impact," 2001.

[3] Desai, S., "Relative roles of computational fluid dynamics and wind tunnel testing in the development-of aircraft," *Current Science*, Vol. 84, 2003.

[4] Tinoco, E. N., Bogue, D. R., Kao, T.-J., Yu, N. J., Li, P., and Ball, D. N., "Progress toward CFD for full flight envelope," *The Aeronautical Journal (1968)*, Vol. 109, No. 1100, 2005, p. 451460.

[5] Malik, M. R., and Bushnell, D. M., "Role of computational fluid dynamics and wind tunnels in aeronautics R and D," 2012.

[6] van Leer, B., and Powell, K. G., *Introduction to computational fluid dynamics*, 2010.

[7] Hussaini, M., van Leer, B., and Van Rosendale, J., *Upwind and High-Resolution Schemes*, Springer Berlin Heidelberg, 2015.

[8] Tinoco, E. N., "CFD codes and applications at Boeing," *Sadhana*, Vol. 16, 1991, pp. 141–163.

[9] Jameson, A., "Computational fluid dynamics: past, present, and future," Stanford University, 2012.

[10] Velmurugarajan, K., "Computational fluid dynamics in aerospace and defence," Alliance College of Engineering and Design, 2021.

[11] Tinoco, E. N., "CFD uncertainty and validation for commercial aircraft applications," 2007.

[12] Slotnick, J. P., Khodadoust, A., Alonso, J. J., Darmofal, D. L., Gropp, W. D., Lurie, E. A., and Mavriplis, D. J., "CFD Vision 2030 study: a path to revolutionary computational aerosciences," , 2014.

[13] Park, M., Krakos, J., Michal, T., Loseille, A., and Alonso, J., "Unstructured grid adaptation: status, potential impacts, and recommended investments toward CFD Vision 2030," 2016.

[14] Tinoco, E. N., Brodersen, O. P., Keye, S., Laflin, K. R., Feltrop, E., Vassberg, J. C., Mani, M., Rider, B., Wahls, R. A., Morrison, J. H., Hue, D., Roy, C. J., Mavriplis, D. J., and Murayama, M., "Summary data from the Sixth AIAA CFD Drag Prediction Workshop: CRM Cases," *Journal of Aircraft*, Vol. 55, No. 4, 2018, pp. 1352–1379.

[15] Rumsey, C. L., Slotnick, J. P., and Sclafani, A. J., "Overview and summary of the Third AIAA High Lift Prediction Workshop," *Journal of Aircraft*, Vol. 56, No. 2, 2019, pp. 621–644.

[16] Mavriplis, D. J., "Unstructured grid techniques," *Annual Review of Fluid Mechanics*, Vol. 29, No. 1, 1997, pp. 473–514.

[17] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, 2013.

[18] Bassi, F., and Rebay, S., "High–order accurate discontinuous finite element solution of the 2-D Euler equations," *Journal of Computational Physics*, Vol. 138, 1997, pp. 251–285.

[19] Hartmann, R., and Houston, P., "Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations," *Journal of Computational Physics*, Vol. 183, No. 2, 2002, pp. 508–532.

[20] Fidkowski, K. J., and Darmofal, D. L., "An adaptive simplex cut-cell method for discontinuous Galerkin discretizations of the Navier-Stokes Equations," AIAA Paper 2007-3941, 2007.

[21] Reza Ahrabi, B., "An hp-adaptive Petrov-Galerkin Method for steady-state and unsteady flow problems," Ph.D. thesis, 08 2015.

[22] Reza Ahrabi, B., Anderson, K., and Newman, J., "An adjoint-based hp-adaptive Petrov-Galerkin method for turbulent flows," 2015.

[23] Fidkowski, K. J., and Chen, G., "Output-based mesh optimization for hybridized and embedded discontinuous Galerkin methods," *International Journal for Numerical Methods in Engineering*, Vol. 121, No. 5, 2019, pp. 867–887.

[24] Fidkowski, K. J., "A simplex cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007.

[25] Anderson, K., Reza Ahrabi, B., and Newman, J., "Finite-element solutions for turbulent flow over the NACA 0012 airfoil," 2015.

[26] Reed, W., and Hill, T., "Triangular mesh methods for the neutront transport equation," Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.

[27] Cockburn, B., Lin, S.-Y., and Shu, C.-W., "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems," *Journal of Computational Physics*, Vol. 84, No. 1, 1989, pp. 90–113.

[28] Bey, K. S., and Oden, J. T., "hp-Version discontinuous Galerkin methods for hyperbolic conservation laws," *Computer Methods in Applied Mechanics and Engineering*, Vol. 133, 1996, pp. 259–286.

[29] Oliver, T. A., Fidkowski, K. J., and Darmofal, D. L., "Multigrid solution for high–order discontinuous Galerkin discretization of the compressible Navier-Stokes equations," *Third International Conference on Computational Fluid Dynamics, Toronto, Canada*, 2004.

[30] Persson, P.-O., and Peraire., J., "An efficient low memory implicit DG algorithm for time dependent problems," AIAA Paper 2006-113, 2006.

[31] Hartmann, R., "Adjoint consistency analysis of discontinuous Galerkin discretizations," *SIAM Journal on Numerical Analysis*, Vol. 45, No. 6, 2007, pp. 2671–2696.

[32] Diosady, L., and Darmofal, D., "Discontinuous Galerkin solutions of the Navier-Stokes equations using linear multigrid preconditioning." AIAA Paper 2007-3942, 2007.

[33] Wang, L., and Mavriplis, D., "Adjoint-based $h-p$ adaptive discontinuous Galerkin methods for the 2D compressible Euler equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 7643–7661.

[34] Burgess, N. K., and Mavriplis, D. J., "An *hp*-Adaptive Discontinuous Galerkin Solver for Aerodynamic Flows on Mixed-Element Meshes," AIAA Paper 2011-490, 2011.

[35] Cockburn, B., and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261.

[36] Fidkowski, K. J., "Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flows," *International Journal for Numerical Methods in Engineering*, Vol. 88, No. 12, 2011, pp. 1297–1322.

[37] Bassi, F., and Rebay, S., "A High–order discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 131, 1997, pp. 267–279.

[38] Bassi, F., and Rebay, S., "An implicit high-order discontinuous Galerkin method for the steady state compressible Navier-Stokes equations," *Computational Fluid Dynamics 98, Proceedings from the Fourth European Computational Fluid Dynamics Conference*, Vol. 2, 1998, pp. 1227–1233.

[39] Bassi, F., and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.

[40] Nguyen, N., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 3232–3254.

[41] Cockburn, B., Gopalakrishnan, J., and Lazarov, R., "Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems," *SIAM Journal on Numerical Analysis*, Vol. 47, No. 2, 2009, pp. 1319–1365.

[42] Nguyen, N. C., Peraire, J., and Cockburn, B., "Hybridizable discontinuous Galerkin methods," *Spectral and High Order Methods for Partial Differential Equations*, Lecture Notes in Computational Science and Engineering, Vol. 76, edited by J. S. Hesthaven, E. M. Rnquist, T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, Springer Berlin Heidelberg, 2011, pp. 63–84. 10.1007/978-3-642-15337-2_4.

[43] Peraire, J., Nguyen, N. C., and Cockburn, B., "An embedded discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations," AIAA Paper 2011-3228, 2011.

[44] Dahm, J. P., and Fidkowski, K. J., "Error estimation and adaptation in hybridized discontinous Galerkin methods," AIAA Paper 2014–0078, 2014.

[45] Fidkowski, K., and Chen, G., "Output-based mesh optimization for hybridized and embedded discontinuous Galerkin Methods," *International Journal for Numerical Methods in Engineering*, Vol. 121, 2019.

[46] Baker, T. J., "Mesh adaptation strategies for problems in fluid dynamics," *Finite Elements in Analysis and Design*, Vol. 25, 1997, pp. 243–273.

[47] Anderson, W. K., Wang, L., Kapadia, S., Tanis, C., and Hilbert, B., "Petrov-Galerkin and discontinuous-Galerkin methods for time-domain and frequency-domain electromagnetic simulations," *Journal of Computational Physics*, Vol. 230, No. 23, 2011, pp. 8360–8385.

[48] Anderson, K., Newman, J., and Karman, S., "Stabilized finite elements in FUN3D," *Journal of Aircraft*, Vol. 55, 2017, pp. 1–19.

[49] Brooks, A., and Hughes, T., "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, 1982, pp. 199–259.

[50] Tezduyar, T., and Hughes, T., "Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations," 1983.

[51] Hughes, T. J., and Mallet, M., "A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 58, No. 3, 1986, pp. 305–328.

[52] Shakib, F., Hughes, T. J., and Johan, Z., "A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 89, No. 1, 1991, pp. 141–219. Second World Congress on Computational Mechanics.

[53] Franca, L. P., and Frey, S. L., "Stabilized finite element methods: II. The incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 99, No. 2, 1992, pp. 209–233.

[54] Whiting, C., and Jansen, K., "A stabilized finite element method for the incompressible Navier-Stokes equations using a hierarchical basis," *International Journal for Numerical Methods in Fluids - INT J NUMER METHOD FLUID*, Vol. 35, 2001, pp. 93–116.

[55] Venkatakrishnan, V., Allmaras, S., Kamenetskiy, D., and Johnson, F., "Higher order schemes for the compressible Navier-Stokes equations," 2003.

[56] Kirk, B. S., "Adaptive finite element simulation of flow and transport applications on parallel computers," Ph.D. thesis, The University of Texas, Austin, 2005.

[57] Hughes, T., Scovazzi, G., and Tezduyar, T., "Stabilized methods for compressible flows," *Journal of Scientific Computing*, Vol. 43, 2010, pp. 343–368.

[58] Glasby, R., Burgess, N., Anderson, K., Wang, L., Allmaras, S., and Mavriplis, D., "Comparison of SU/PG and DG finite-element techniques for the compressible Navier-Stokes equations on anisotropic unstructured meshes," 2013.

[59] Reza Ahrabi, B., Anderson, K., and III, J., "An adjoint-based hp-adaptive stabilized finite-element method with shock capturing for turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 318, 2017, pp. 1030–1065.

[60] Rivire, B., Wheeler, M. F., and Girault, V., "A Priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems," *SIAM Journal on Numerical Analysis*, Vol. 39, No. 3, 2002, pp. 902–931.

[61] Eriksson, K., Johnson, C., and Thomee, V., "Adaptive finite element methods for parabolic problems I: A linear model problem," *Mathematical Modelling and Numerical Analysis*, Vol. 19, No. 4, 1985, pp. 611–643.

[62] Castro-Diaz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic unstructured mesh adaptation for flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 25, 1997, pp. 475–491.

[63] Dompierre, J., Vallet, M.-G., Bourgault, Y., Fortin, M., and Habashi, W. G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III: unstructured meshes," *International Journal for Numerical Methods in Fluids*, Vol. 39, 2002, pp. 675–702.

[64] Venditti, D. A., and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46.

[65] Fidkowski, K. J., and Darmofal, D. L., "A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 225, 2007, pp. 1653–1672.

[66] Mavriplis, D. J., Vassberg, J. C., Tinoco, E. N., Mani, M., Brodersen, O. P., Eisfeld, B., Wahls, R. A., Morrison, J. H., Zickuhr, T., Levy, D., and Murayama, M., "Grid quality and resolution issues from the Drag Prediction Workshop series," AIAA Paper 2008-930, 2008.

[67] Nemec, M., Aftosmis, M. J., and Wintzer, M., "Adjoint-based adaptive mesh refinement for complex geometries," AIAA Paper 2008-0725, 2008.

[68] Warren, G. P., Anderson, W. K., Thomas, J. L., and Krist, S. L., "Grid convergence for adaptive methods," AIAA Paper 1991-1592, 1991.

[69] Barth, T. J., "Numerical methods for gas dynamic systems on unstructured meshes," *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws, Proceedings of the International School on Theory and Numerics for Conservation Laws, Berlin, Lecture Notes in Computational Science and Engineering*, edited by D. Kröner, M. Ohlberger, and C. Rhode, Springer-Verlag, 1999.

[70] Pirzadeh, S. Z., "An adaptive unstructured grid method by grid subdivision, local remeshing, and grid movement," , No. 99-3255, 1999.

[71] Zhang, X. D., Vallet, M.-G., Dompierre, J., Labbe, P., Pelletier, D., Trepanier, J.-Y., Camarero, R., Lassaline, J. V., Manzano, L. M., and Zingg, D. W., "Mesh adaptation using different error indicators for the Euler equations," AIAA Paper 2001-2549, 2001.

[72] Venditti, D. A., and Darmofal, D. L., "Grid adaptation for functional outputs: application to two-dimensional inviscid flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40–69.

[73] Fidkowski, K. J., and Darmofal, D. L., "Output-based error estimation and mesh adaptation: overview and recent results," AIAA Paper 2009-1303, 2009.

[74] Fidkowski, K. J., and Darmofal, D. L., "Review of output-based error estimation and mesh adaptation in computational fluid dynamics," *American Institute of Aeronautics and Astronautics Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[75] Becker, R., and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

[76] Giles, M., and Pierce, N., "Adjoint error correction for integral outputs," *Lecture Notes in Computational Science and Engineering: Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*, Vol. 25, Springer, Berlin, 2002.

[77] Houston, P., and Süli, E., "Error estimation and adaptive discretization methods in computational fluid dynamics," Springer-Verlag, Heidelberg, Lecture Notes in Computational Science and Engineering Vol 25, 2002, pp. 269–344.

[78] Giles, M. B., and Pierce, N. A., "Adjoint equations in CFD: duality, boundary conditions and solution behavior," AIAA Paper 97-1850, 1997.

[79] Barth, T. J., "Space-time error representation and estimation in Navier-Stokes calculations," *Complex Effects in Large Eddy Simulations*, edited by S. C. Kassinos, C. A. Langer, G. Iaccarino, and P. Moin, Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007, pp. 29–48.

[80] Besier, M., and Rannacher, R., "Goal-oriented space-time adaptivity in the finite element Galerkin method for the compution of nonstationary incompressible flow," *International Journal for Numerical Methods in Fluids*, Vol. 70, 2012, pp. 1139–1166.

[81] Schmich, M., and Vexler, B., "Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 1, 2008, pp. 369–393.

[82] Mani, K., and Mavriplis, D. J., "Error estimation and adaptation for functional outputs in time-dependent flow problems," *Journal of Computational Physics*, Vol. 229, 2010, pp. 415–440.

[83] Fidkowski, K. J., and Luo, Y., "Output-based space-time mesh adaptation for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773.

[84] Fidkowski, K. J., "An output-based dynamic order refinement strategy for unsteady aerodynamics," AIAA Paper 2012-77, 2012.

[85] Kast, S. M., and Fidkowski, K. J., "Output-based mesh adaptation for high order Navier-Stokes simulations on deformable domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494. doi:10.1016/j.jcp.2013.06.007.

[86] Fidkowski, K. J., and Roe, P. L., "Entropy-based mesh refinement, I: the entropy adjoint approach," AIAA Paper 2009-3790, 2009.

[87] Fidkowski, K. J., and Roe, P. L., "An entropy adjoint approach to mesh refinement," *SIAM Journal on Scientific Computing*, Vol. 32, No. 3, 2010, pp. 1261–1287.

[88] Houston, P., Hartmann, R., and Süli, E., "Adaptive discontinuous Galerkin finite element methods for compressible fluid flows," *Numerical Methods for Fluid Dynamics VII, ICFD*, edited by M. Baines, 2001, pp. 347–353.

[89] Houston, P., and Süli, E., "A note on the design of hp-adaptive finite element methods for elliptic partial differential equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, 2005, pp. 229–243.

[90] Ceze, M. A., and Fidkowski, K. J., "A robust adaptive solution strategy for high-order implicit CFD solvers," AIAA Paper 2011-3696, 2011.

[91] Yano, M., "An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

[92] Fidkowski, K. J., "A local sampling approach to anisotropic metric-based mesh optimization," AIAA Paper SciTech, 2016.

[93] Allmaras, S., Johnson, F., and Spalart, P., "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model," Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.

[94] Boussinesq, J., "Essai sur la thorie des eaux courantes," *Mmoires prsents par divers savants l'Acadmie des Sciences*, Vol. 23, 1877, pp. 1–680.

[95] Fidkowski, K. J., "Output-based error estimation and mesh adaptation for steady and unsteady flow problems," *38th Advanced CFD Lectures Series; Von Karman Institute for Fluid Dynamics (September 14–16 2015)*, edited by H. Deconinck and T. Horvath, von Karman Institute for Fluid Dynamics, 2015.

[96] Kast, S. M., "Methods for optimal output prediction in computational fluid dynamics," Ph.D. thesis, The University of Michigan, 2016.

[97] Dahm, J., "Toward accurate, efficient, and robust hybridized discontinuous galerkin methods," Ph.D. thesis, The University of Michigan, 2017.

[98] Ding, K., "Efficient output-based adaptation mechanics for high-order computational fluid dynamics methods," Ph.D. thesis, The University of Michigan, 2018.

[99] Shimizu, Y., "Output-based error estimation and model reduction for chaotic flows," Ph.D. thesis, The University of Michigan, 2019.

[100] Sanjaya, D. P., "Towards automated, metric-conforming, mesh optimization for high-order, finite-element methods," Ph.D. thesis, The University of Michigan, 2019.

[101] Chen, G., "Enabling automated, reliable and efficient aerodynamic shape optimization with output-based adapted meshes," Ph.D. thesis, The University of Michigan, 2020.

[102] Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[103] Lu, J., "An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[104] Saad, Y., and Schultz, M. H., "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[105] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high–order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 207, 2005, pp. 92–113.

[106] Fidkowski, K. J., "Output-based space-time mesh optimization for unsteady flows using continuous-in-time adjoints," *Journal of Computational Physics*, Vol. 341(15), 2017, pp. 258–277.

[107] Doetsch, K. T., and Fidkowski, K. J., "A combined entropy and output-based adaptive approach using a stabilized continuous finite element formulation," AIAA Paper 2022–0581, 2022.

[108] Brooks, A. N., and Hughes, T. J., "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 32, No. 1, 1982, pp. 199–259.

[109] Hughes, T. J. R., Franca, L. P., and Mallet, M., "A new finite element formulation for computational fluid dynamics: I. symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics." *Computer Methods in Applied Mechanics and Engineering*, Vol. 54, 1986, pp. 223–234.

[110] Wang, L., Anderson, W. K., Erwin, J. T., and Kapadia, S., "Discontinuous Galerkin and Petrov Galerkin methods for compressible viscous flows," *Computers and Fluids*, Vol. 100, 2014, pp. 13–29.

[111] Ahrabi, B. R., and Mavriplis, D. J., "A scalable solution strategy for high-order stabilized finite-element solvers using an implicit line preconditioner," *Computer Methods in Applied Mechanics and Engineering*, Vol. 341, 2018, pp. 956–984.

[112] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H., "PETSc web page," , 2021. `https://www.mcs.anl.gov/petsc`.

[113] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H., "PETSc users manual," Tech. Rep. ANL-95/11 - Revision 3.15, Argonne National Laboratory, 2021. `https://www.mcs.anl.gov/petsc`.

[114] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., "Efficient management of parallelism in object oriented numerical software libraries," *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, Birkhäuser Press, 1997, pp. 163–202.

[115] Dupont, T., Kendall, R. P., and Rachford, H. H., Jr., "An approximate factorization procedure for solving self-adjoint elliptic difference equations," *SIAM Journal on Numerical Analysis*, Vol. 5, No. 3, 1968, pp. 559–573.

[116] Oliphant, T. A., "An implicit, numerical method for solving two-dimensional time-dependent diffusion problems," *Quarterly of Applied Mathematics*, Vol. 19, No. 3, 1961, pp. 221–229.

[117] Chan, T., Van, H., and Van der Vorst, H., "Approximate and incomplete factorizations," 1970.

[118] Saad, Y., and Schultz, M. H., "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[119] Oliver, T. A., and Darmofal, D. L., "Impact of turbulence model irregularity on high–order discretizations," AIAA Paper 2009-953, 2009.

[120] Hartmann, R., "Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 51, No. 9–10, 2006, pp. 1131–1156.

[121] Doetsch, K., and Fidkowski, K. J., "Combined entropy and output-based adjoint approach for mesh refinement and error estimation," *AIAA Journal*, Vol. 57, No. 8, 2019.

[122] Tadmor, E., "The numerical viscosity of entropy stable schemes for systems of conservation laws. I," *Mathematics of Computation*, Vol. 49, No. 179, 1987, pp. 91–103.

[123] Doetsch, K. T., and Fidkowski, K. J., "Unsteady combined entropy and output-based adjoint approach for mesh refinement and error estimation," AIAA Paper 2019–2951, 2019.

[124] Kast, S. M., Fidkowski, K. J., and Roe, P. L., "An unsteady entropy adjoint approach for adaptive solution of the shallow-water equations," AIAA Paper 2011-3694, 2011.

[125] Ceze, M. A., and Fidkowski, K. J., "An anisotropic hp-adaptation framework for functional prediction," *American Institute of Aeronautics and Astronautics Journal*, Vol. 51, 2013, pp. 492–509.

[126] Habashi, W. G., Dompierre, J., Bourgault, Y., Ait-Ali-Yahia, D., Fortin, M., and Vallet, M.-G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. part I: general principles," *International Journal for Numerical Methods in Fluids*, Vol. 32, 2000, pp. 725–744.

[127] Loseille, A., and Alauzet, F., "Continuous mesh framework part I: well-posed continuous interpolation error," *SIAM J. Numerical Analysis*, Vol. 49, 2011, pp. 38–60.

[128] Loseille, A., and Alauzet, F., "Continuous mesh framework part II: validations and applications," *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, 2011, pp. 61–86.

[129] Fidkowski, K. J., and Chen, G., "Metric-based, goal-oriented mesh adaptation using machine learning," *Journal of Computational Physics*, Vol. 426, 2021.

[130] Pennec, X., Fillard, P., and Ayache, N., "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, Vol. 66, No. 1, 2006, pp. 41–66.

[131] Hecht, F., "BAMG: Bidimensional Anisotropic Mesh Generator," INRIA–Rocquencourt, France, 1998. `http://pauillac.inria.fr/cdrom/www/bamg/bamg/eng.htm`.

[132] Frey, P., and Alauzet, F., "Anisotropic mesh adaptation for CFD computations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, 2005, pp. 5068–5082.

[133] Doetsch, K. T., and Fidkowski, K. J., "A combined entropy and output-based adjoint approach for mesh refinement and error estimation," AIAA Paper 2018-0918, 2018.

[134] Persson, P.-O., and Peraire., J., "Sub-cell shock capturing for discontinuous Galerkin methods," AIAA Paper 2006-112, 2006.

[135] Spalart, P. R., "Airplane trailing vortices," *Annual Review of Fluid Mechanics*, Vol. 30, 1998, pp. 107–138.

[136] Ceze, M. A., and Fidkowski, K. J., "High-order output-based adaptive simulations of turbulent flow in two dimensions," *AIAA Journal*, Vol. 54, No. 9, 2016.

[137] Fidkowski, K. J., Ceze, M. A., and Roe, P. L., "Entropy-based drag error estimation and mesh adaptation in two dimensions," *AIAA Journal of Aircraft*, Vol. 49, No. 5, 2012, pp. 1485–1496.

[138] Oswatitsch, K., *Gas dynamics*, Academic, New York, 1956.

[139] Bassi, F., Colombo, A., Crivellini, A., Fidkowski, K. J., Franciolini, M., Ghidoni, A., and Noventa, G., "An entropy-adjoint p-adaptive discontinuous Galerkin method for the under-resolved simulation of turbulent flows," *AIAA Journal*, 2020.

[140] Oliver, T. A., "A high-order, adaptive, discontinuous Galerkin finite elemenet method for the Reynolds-Averaged Navier-Stokes equations," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[141] Fidkowski, K. J., "Lecture notes in advanced computational fluid dynamics," 2017.